

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.

[1 Capítulo 1 3](#_Toc148024836)

[1.1 Preguntas 3](#_Toc148024837)

[1.1.1 Explicar el propósito principal de un sistema operativo 3](#_Toc148024838)

[1.1.2 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 3](#_Toc148024839)

[1.1.3 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 3](#_Toc148024840)

[1.1.4 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 3](#_Toc148024841)

[1.1.5 Modelo conceptual del sistema operativo en una perspectiva de arriba hacia abajo. Ídem de abajo hacia arriba. 4](#_Toc148024842)

[1.1.6 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. 4](#_Toc148024843)

[1.1.7 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? 4](#_Toc148024844)

[1.1.8 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”. 5](#_Toc148024845)

[Rectifique adecuadamente ese falso enunciado 5](#_Toc148024846)

[1.1.9 Explicar las diferencias claras entre driver de dispositivo y controlador de dispositivo. 5](#_Toc148024847)

[1.1.10 Mediante el esquema que se acompaña, explicar el concepto IRQ en el SO 5](#_Toc148024848)

[1.1.11 Diferencia entre trampa (TRAP) e interrupción utilizando la siguiente tabla: 6](#_Toc148024849)

[1.1.12 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 6](#_Toc148024850)

[1.1.13 Qué estructura de datos es la de la figura y ponga ejemplos donde el SO la emplea 7](#_Toc148024851)

[1.1.14 Qué estructura de datos es la de la figura y ponga ejemplos donde el SO la emplea 7](#_Toc148024852)

[1.1.15 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: 7](#_Toc148024853)

# Capítulo 1. Introducción al Sistema Operativo

## Preguntas

### 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.

El concepto de que un sistema operativo proporciona principalmente abstracciones a los programas de aplicación es una **visión de arriba hacia abajo**.

Una visión alternativa, **de abajo hacia arriba**, sostiene que el sistema operativo está ahí para administrar todas las piezas de un sistema complejo.

### 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”.

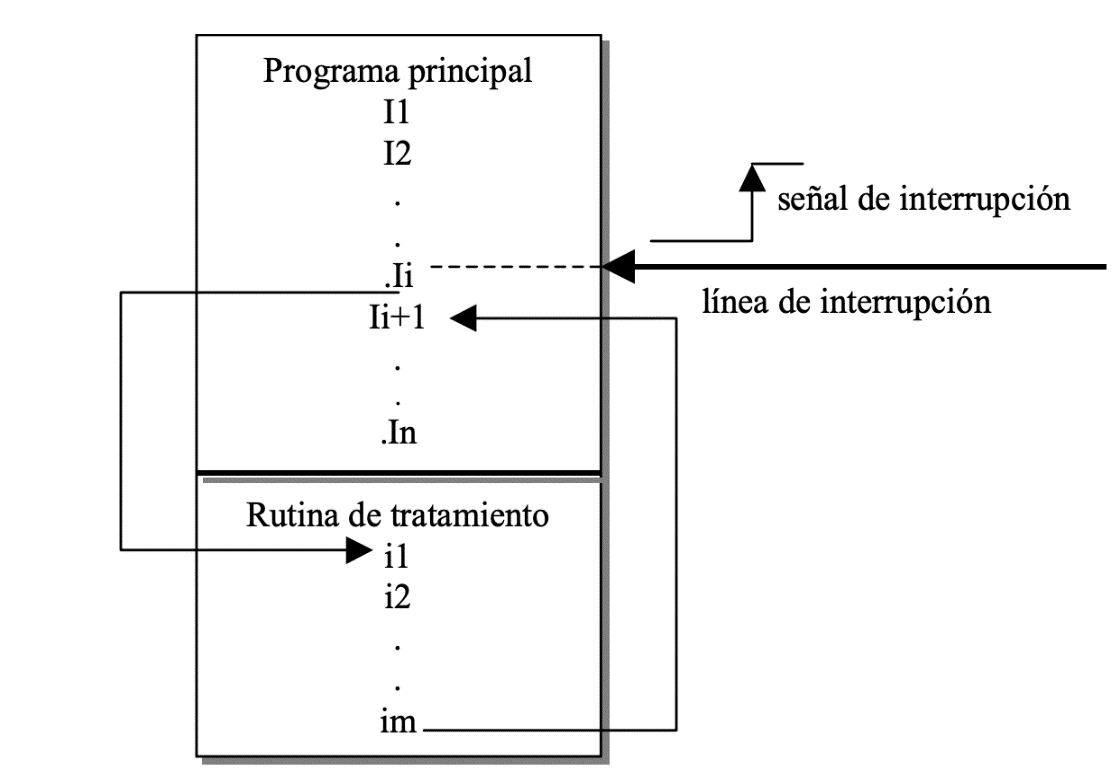
## Rectifique adecuadamente ese falso enunciado

“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.

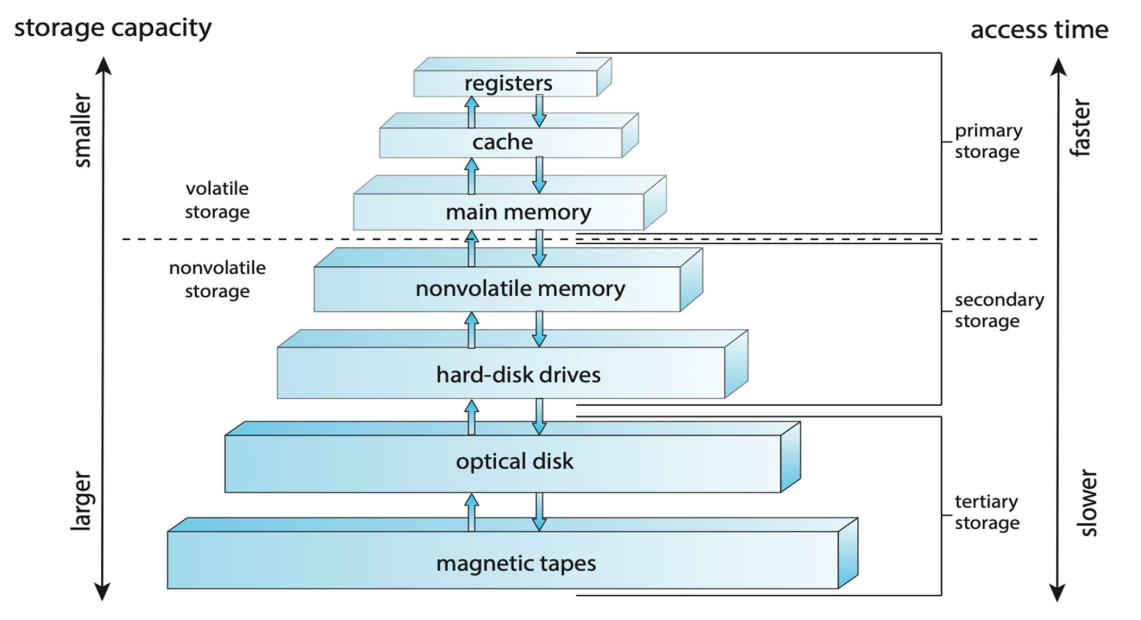
### Mediante el esquema que se acompaña, explicar el concepto IRQ en el SO

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.

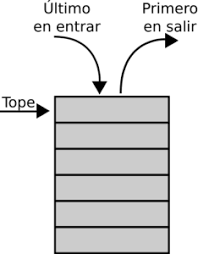
### Diferencia entre trampa (TRAP) e interrupción utilizando la siguiente tabla:

|  |  |  |
| --- | --- | --- |
|  | **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

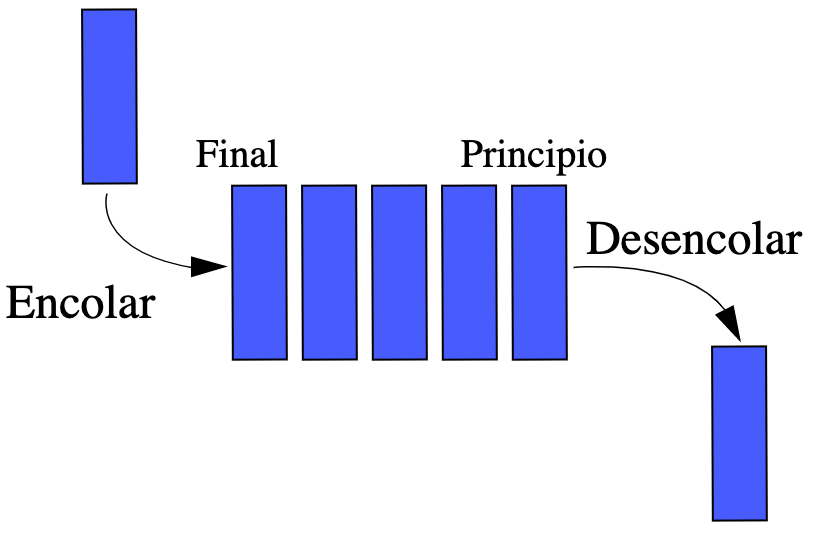


### Qué estructura de datos es la de la figura y ponga ejemplos donde el SO la emplea



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: Llamadas y tipos de sistemas operativos.

## Preguntas

### ¿Cuál es el propósito del intérprete de comandos? ¿Por qué suele estar separado del núcleo?

La función principal del intérprete de comandos es obtener y ejecutar el comando especificado por el usuario.Suele estar separado del núcleo porque la mayoría de los sistemas operativos, incluidos Linux, UNIX y Windows, tratan al intérprete de comandos como un programa especial que se ejecuta cuando se inicia un proceso o cuando un usuario inicia sesión por primera vez (en sistemas interactivos), existiendo sistemas operativos con múltiples intérpretes de comandos para elegir, que se conocen como shells.

### 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.

### Ventajas de GUI

Una interfaz gráfica de usuario o GUI es fácil de usar. En lugar de ingresar textos que expresan comandos directamente a través de una CLI, los usuarios emplean un sistema de ventanas y menús basado en ratón caracterizado por una metáfora de escritorio. El usuario mueve el dispositivo ratón para colocar su puntero sobre imágenes o iconos en la pantalla (el escritorio) que representan programas, archivos, directorios y funciones del sistema gráfico. Dependiendo de la ubicación del puntero, hacer clic en un botón del ratón puede invocar un programa, seleccionar un archivo o directorio, conocido como carpeta, o desplegar un menú que contiene comandos.

### ¿Cuál es el propósito de las llamadas al sistema?

Una llamada al sistema es una rutina que permite a una aplicación de usuario solicitar acciones que requieren privilegios especiales. La adición de llamadas al sistema es una de varias maneras de ampliar las funciones proporcionadas por el kernel.

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

Una llamada al sistema difiere de una función de usuario de varias formas:

* Una llamada al sistema tiene más privilegios que una subrutina normal.
* Una llamada al sistema se ejecuta con el privilegio modo de kernel en el dominio de protección de kernel.
* El código de llamada del sistema y los datos se encuentran en la memoria global del kernel.
* Las rutinas de llamada al sistema pueden crear y utilizar procesos de kernel para realizar el proceso asíncrono.
* Las llamadas del sistema no pueden utilizar bibliotecas compartidas ni ningún símbolo que no se encuentre en el dominio de protección del kernel.

### Indique los seis tipos de llamadas al sistema que generalmente tienen los SO

En general las llamadas al sistema se pueden agrupar aproximadamente en seis categorías principales: control de procesos, gestión de archivos, gestión de dispositivos, mantenimiento de la información, comunicaciones y protección.

### ¿Que es POSIX?

POSIX ([acrónimo](https://es.wikipedia.org/wiki/Acrónimo) 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/Intérprete_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".

### ¿Cuáles son los tipos de SO según estructura de kernel? De una explicación de una línea de cada uno de ellos

* **Sistemas operativos de kernel monolítico/monolítico con extensiones cargables**: Los sistemas operativos monolíticos se caracterizan por implementar en el núcleo los cuatro componentes fundamentales del sistema operativo.
* **Sistemas operativos de microkernel:** Se caracterizan por disponer de un núcleo que implementa únicamente Planificación de proceso, Mecanismo de comunicación entre procesos y la Gestión de interrupciones. Además, existen procesos servidores que, cuando un proceso cualquiera solicita un servicio a través de una llamada al sistema, el micronúcleo canaliza la petición al proceso servidor correspondiente.
* **Sistemas operativos de kernel híbrido**: El kernel de este tipo de Sistema Operativo el núcleo en cuestión usa conceptos de arquitectura tanto del diseño monolítico como del micronúcleo.

### Comparación de un kernel monolítico de un microkernel

Los sistemas operativos monolíticos se caracterizan por implementar en el núcleo los cuatro componentes fundamentales del sistema operativo, que son la planificación de procesos, la administración de la memoria principal, la administración de ficheros y la gestión de los dispositivos de entrada/salida. Algunos de ellos admiten la inclusión de módulos compilados en tiempo de ejecución del kernel del sistema operativo sin necesidad de recompilar completamente el mismo.

Por otra parte, los microkernel se caracterizan por disponer de un núcleo que implementa únicamente las funciones básicas de Planificación de proceso, Mecanismo de comunicación entre procesos y la Gestión de interrupciones.

### ¿Qué tipo de núcleo llevan los siguientes SO?

|  |  |
| --- | --- |
| Ubuntu | Monolítico con extensiones. |
| Windows 10 | Híbrido. |
| MacOS | Monolítico con extensiones. |
| MacOSX | Híbrido. |
| Android | Monolítico con extensiones. |
| IOS | Híbrido. |
|  |  |

### Utilizando la tabla a continuación, establecer las diferencias entre el kernel de un S.O. monolítico con módulos recargable y un S.O. híbrido

|  |  |  |
| --- | --- | --- |
|  | **Monolítico con Módulos Recargables** | **Híbrido** |
| Arquitectura |  |  |
| Espacio del S.O. en modo usuario |  |  |
| Espacio del S.O. en modo kernel |  |  |
| Modularidad |  |  |
| Modo de operación de los módulos |  |  |
| Riesgo |  |  |
| Coste de diseño |  |  |
| Coste o facilidad de mantenimiento |  |  |
| Consumo de recursos computacionales |  |  |
| Ejemplos |  |  |

# Capítulo 3. Procesos

## Preguntas

### ¿Qué es un proceso del sistema operativo? ¿Qué recursos del sistema necesita un proceso para realizar satisfactoriamente su tarea?

El proceso es un programa en ejecución. El estado de la actividad actual de un proceso está representado por el valor del contador del programa y el contenido de los registros del procesador. Por último, los procesos necesitan recursos para ejecutarse como son:

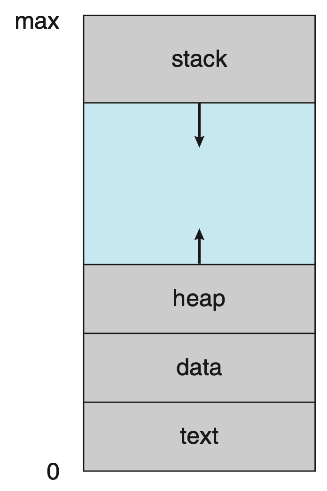
* Tiempo de CPU.
* Memoria.
* Archivos.
* Dispositivos E/S.

### ¿Cuales son las obligaciones del S.O. como gestor de procesos?

Las obligaciones del SO como gestor de procesos son:

* Creación y eliminación de procesos
* Planificación de procesos
* Establecimiento de mecanismos para la sincronización y comunicación de procesos.
* Manejo de bloqueos mutuos

### ¿Qué suele almacenarse en cada una de las secciones de la memoria ocupada por un proceso?

El diseño de la memoria que un proceso necesita generalmente se divide en varias secciones:

* Sección de texto: el código ejecutable
* Sección de datos: variables globales
* Sección Heap: memoria que se asigna dinámicamente durante el tiempo de ejecución del programa
* Sección stack (de pila): almacenamiento temporal de datos al invocar funciones (como parámetros de función, direcciones de retorno y variables locales.

### Se ha ejecutado el comando “size /bin/more”. Explicar la salida obtenida

[root@localhost home]# size /bin/more

   text    data     bss     dec     hex filename

  29527    1824      24   31375    7a8f /bin/more

### Respecto a los estados de un proceso, rellene adecuadamente la tabla

| **Estado** | **Descripción** |
| --- | --- |
| Nuevo | El proceso se está creando. |
| Preparado | El proceso está esperando a ser asignado a un procesador. |
| Activo | Se están ejecutando instrucciones. |
| Bloqueado | El proceso está esperando que ocurra algún evento (como una finalización de E/S o la recepción de una señal). |
| Terminado | El proceso ha terminado la ejecución. |

### ¿Qué es un PCB? ¿Qué información generalmente se suele guardar en un PCB?

Cada proceso está representado en el sistema operativo por un bloque de control de procesos (PCB), también llamado bloque de control de tareas. Contiene muchos campos de información asociados a un proceso específico como:

* **Estado del proceso.** El estado puede ser nuevo, listo, en ejecución, en espera, detenido, etc.
* **Contador de programa** indica la dirección de la siguiente instrucción a ejecutar para este proceso
* **Registros de la CPU**. Los registros varían en número y tipo, dependiendo de la arquitectura de la computadora. Incluyen acumuladores, registros de índice, punteros de pila y registros de propósito general, además de cualquier información que condicione la ejecución del proceso. Junto con el contador del programa, esta información de estado debe guardarse cuando ocurre una interrupción, para permitir que el proceso continúe correctamente cuando se reanude la ejecución.
* **Información de planificación (scheduling) de CPU**: prioridad de proceso, punteros a colas de scheduling, etc.
* **Información de gestión de memoria:** valor de los registros base y límite, y las tablas de páginas o de segmentos, según el sistema de memoria utilizado por el sistema operativo.
* **Información de contabilidad de CPU**: la cantidad de CPU y el tiempo real utilizado, los límites de tiempo, la identificación de tareas o de proceso, etc.
* **Información de estado de E/S:** la lista de dispositivos de E/S asignados al proceso, una lista de archivos abiertos, etc.

## Ejercicios

# Capitulo 4

## Preguntas

### Se dice que existe una pseudosimultaneidad cuando:

1. Una CPU con un solo núcleo tiende que compartir su tiempo de proceso entre varias tareas, tal que en un instante de tiempo está ejecutando varias instrucciones de programa a la vez
2. El SO organiza y calcula los hilos de forma óptima tal que varios usuarios (al frente de diferentes procesos) tienen la impresión de estar atendidos a la vez por una CPU mononúcleo
3. Una CPU multinúcleo donde se procesan varios tareas de forma simultánea.
4. Una CPU con varios microprocesadores donde se procesan varios tareas de forma simultánea

### Respecto a multithreading:

1. La concurrencia o multithreading de una aplicación depende exclusivamente de la gestión del SO
2. El efecto de concurrencia o multithreading de una aplicación depende exclusivamente del programador que la crea.
3. La simultaneidad real de una aplicación creada con hilos o tareas concurrentes necesita que la CPU sea multinúcleo.
4. Una CPU multinúcleo siempre producirá mayor velocidad de proceso que una mononúcleo al ejecutar cualquier aplicación

### Se dispone de un ordenador monoprocesador que permite multiprogramación, para lo cual es preciso:

1. Utilizar técnicas de multiplexación en el tiempo virtual
2. Utilizar técnicas de ejecución de programas en tiempo real
3. Utilizar técnicas de ejecución en tiempo compartido
4. Fragmentar de forma lógica la memoria caché en tantas partes como programas en ejecución tengamos

## En relación con los Sistemas Operativos, ¿Qué se entiende por overhead?

1. Porcentaje de tiempo que transcurre entre la cabecera (header) del primer paquete y el siguiente
2. Porcentaje de tiempo en el que la CPU está atendiendo a tareas del propio sistema
3. Tiempo que tarda la cabeza del disco, una vez posicionada en la pista, en acceder al sector que le interesa
4. Porcentaje de tiempo que la CPU gasta en tratar los datos de entrada

### Uno de los beneficios de la programación multiproceso (concurrente) pueden ser (marcar la respuesta FALSA):

1. El uso de subprocesos múltiples en una aplicación interactiva puede permitir que un programa continúe ejecutándose incluso si parte de él está bloqueado o está realizando una operación prolongada, lo que aumenta la capacidad de respuesta para el usuario.
2. Supongamos que en un GUI de una aplicación un usuario, al hacer clic en un botón de una pantalla, se desencadena una operación que requiere mucho tiempo de proceso. Una aplicación de thread único no respondería al usuario que quedaría bloqueado hasta que se haya completado la operación. En una aplicación creada con multithreding también habría que esperar a que ese duradero subproceso termine para avisar al usuario con un Thread que puede seguir introduciendo texto.
3. El beneficio de compartir código y datos permite que una aplicación tenga varios subprocesos de actividad diferentes dentro del mismo espacio de direcciones.
4. la creación de subprocesos consume menos tiempo y memoria que la creación de procesos.

### Un kernel de un buen sistema operativo multinúcleo dispone de programas que deben resolver gestiones de multithreding como (SEÑALE LA FALSA):

1. Identificar en las aplicaciones a ejecutar los hilos programados que podrían ser ejecutadas en paralelo en los núcleos individuales de la CPU
2. Resolver la dependencia de una tarea respecto a los datos de otra asegurándose de que las tareas están sincronizadas para adaptarse a la dependencia de los datos
3. Balancear la carga computacional de diferentes tareas concurrentes para optimizar el tiempo de ejecución de la aplicación.
4. Cuando un programa se ejecuta en paralelo en múltiples núcleos, son posibles muchas rutas de ejecución diferentes. Probar y depurar tales programas concurrentes es inherentemente más fácil que probar y depurar aplicaciones de un solo subproceso.

### La tecnología HyperThreading consiste en

1. ‘simular’ dos procesadores (o núcleos físicos) en uno solo lógico, dividiendo la carga de trabajo entre ambos y por tanto mejorando la velocidad de procesamiento.
2. Es una lógica que Apple ya había implementado desde hace tiempo que consiste en duplicar algunos elementos dentro del procesador como los registros o las memorias caché de tercer nivel
3. Podemos ver procesadores de 2, 4 , 6 u 8 núcleos físicos de Intel, que gracias a la tecnología HyperThreading, pueden simular el doble de núcleos de procesamiento.
4. Los procesadores Intel Core i3 poseen solo dos núcleos lógicos pero se comporta como si tuviese 4 núcleos físicos gracias a esta tecnología.

### Una tecnología que “engaña” al sistema operativo al hacerle creer que existen dos núcleos cuando en realidad solo existe uno

1. Es el HyperThreading de las máquinas Macintosh
2. Intel fue pionera con su tecnología HyperThreading,
3. Es la llamada CMT (Cluster Based Multithreading) de los procesadores AMD
4. Es la llamada SMT (Simultaneous Multi-Threading) de Intel

## Aplicando la Ley de Amdahl, supongamos que tenemos una aplicación que es 50 por ciento paralela y 50 por ciento serie. Si ejecutamos esta aplicación en un sistema con dos núcleos de procesamiento, podemos obtener una aceleración de:

1. 1,6 veces
2. 1,3 veces
3. 1,2 veces
4. 2,6 veces

### El soporte para subprocesos se puede proporcionar a nivel de usuario, o por el kernel

1. Subprocesos (hilos) a nivel kernel son administrados directamente por la CPU
2. Subprocesos (hilos) a nivel usuario son administrados directamente por el kernel,
3. Linux y macOS, admiten subprocesos del kernel.
4. Prácticamente todos los sistemas operativos contemporáneos admiten subprocesos del kernel, excepto Windows

### El modelo de gestión de subprocesos del SO de tipo muchos a uno:

1. Asigna muchos subprocesos a nivel de usuario (compilador) a un subproceso del kernel y la gestión de subprocesos la realiza la biblioteca de subprocesos en el espacio del SO.
2. Todo el proceso se bloqueará si un subproceso realiza una llamada al sistema de bloqueo.
3. Como cualquier subproceso puede acceder al kernel, múltiples subprocesos pueden ejecutarse en paralelo en un sistemas multinúcleo.
4. Es el mejor modelo para aprovechar los múltiples núcleos de una CPU multicore.

### El modelo de gestión de subprocesos del SO de tipo uno a uno:

1. Asigna cada subproceso de usuario a un subproceso de la CPU
2. Permite que varios subprocesos se ejecuten en paralelo en multiprocesadores siempre que la creación de un subproceso de usuario se corresponda con la creación del subproceso del kernel correspondiente
3. Linux, junto con la familia de sistemas operativos Windows, no implementan el modelo uno a uno.
4. Multiplexa muchos threads a nivel de usuario en un número mayor de threads del kernel.

## Ejercicios

### Comandos sobre información de CPU

* Pruebe a ejecutar los comandos de Linux que se describen al final del capítulo relacionados con la información de la CPU.
* Haga una breve explicación e interpretación de los principales resultados obtenidos
* Añada otros comandos con variaciones de los anteriores (utilizando otros parámetros)
* Añada otros comandos con relación a la información de procesos y cores, como:

sudo apt-get install cpufrequtils

cpufreq-info

sudo apt-get install cpuid

cpuid

Si se tiene un Intel core i3 i5 i7

sudo apt-get install i7z

sudo i7z

sudo apt-get install inxi

inxi -C

sudo apt-get install hardinfo

hardinfo

### ¿Cómo sabemos si tenemos activado el Hyper Threading?

El Hyper Threading es una característica que hay que activar desde la BIOS pero para entrar, hay que reiniciar el sistema y eso es algo que no siempre podemos hacer cuando queramos.

Con comandos de Linux también podemos saber si está activo el Hyper Threading sin necesidad de rebotar el sistema. Hay varios métodos para ello:

* Con el comando *lscpu*:

[lscpu |grep -i thread

* Con el comando *dmidecode*:

dmidecode -t processor | grep Count

* Revisando el archivo /sys/devices/system/cpu/smt/active:

cat /sys/devices/system/cpu/smt/active

### ¿Cómo contar el número de Threads que está utilizando un proceso de Linux?

Una vez que tenemos el Hyperthreading activado, podemos comprobar si los procesos están utilizando multithreading, si están programados para ello.

En Linux, encontraremos esta información en /proc/PID/status:

cat /proc/13248/status |grep -i thread

# Capitulo 5

## Preguntas:

### Además de elegir el proceso idóneo a ejecutar, el scheduler también tiene que hacer un uso eficiente de la CPU, ya que el cambio de contexto es costoso. Indique cúal de las siguientes acciones pone en evidencia el coste computacional de cambio de contexto

1. En el cambio de contexto el proceso permanece siempre en modo kernel.
2. Se debe guardar el PCB del proceso actual para que puedan recargarse más tarde.
3. En algunos sistemas, también se debe guardar el mapa de memoria (por ejemplo, los bits de referencia de la memoria de disco duro en la tabla de páginas).
4. En el cambio de proceso la memoria caché y las tablas relacionadas permanecen inalterables, lo que obliga a borrarlas dos veces (al ingresar al kernel y al salir de él).

### Respecto a que casi todos los procesos alternan ráfagas de computación (instrucciones aritmético lógicas o transferencias con la memoria) con solicitudes de E/S (de disco, red, etc), indique la respuesta verdadera:

1. Procesos vinculados a la computación (o vinculados a la CPU) pasan la mayor parte computando instrucciones. Tienen ráfagas de CPU cortas y, por lo tanto, esperas de E/S poco frecuentes
2. Procesos vinculados a E/S, pasan la mayor parte del tiempo esperando a que concluya su trabajo en un dispositivo E/S. Tienen ráfagas de CPU largas y, por lo tanto, esperas de E/S poco frecuentes
3. Procesos vinculados a la computación (o vinculados a la CPU) pasan la mayor parte computando instrucciones. Suelen tener ráfagas de CPU largas y, por lo tanto, esperas de E/S poco frecuentes
4. Procesos vinculados a E/S, pasan la mayor parte del tiempo esperando a que concluya su trabajo en la memoria. Tienen ráfagas de CPU largas y, por lo tanto, esperas de E/S poco frecuentes

### Respecto a la alternancia en los procesos de ráfagas de computación (instrucciones aritmético lógicas o transferencias con la memoria) con solicitudes de E/S (de disco, red, etc), indique la respuesta verdadera:

1. Algunas actividades de E/S son de computación. Por ejemplo, cuando la CPU copia bits a una RAM de video para actualizar la pantalla, está computando, no realizando E/S, aunque la CPU está implicada en ello.
2. Si las CPU se vuelven más rápidas, los procesos tienden a vincularse más a las E/S.
3. El factor clave para considerar que un proceso es de CPU o de E/S es la duración de la ráfaga (o ráfagas) de la CPU, no la duración de la ráfaga de E/S.
4. Los procesos vinculados a E/S necesitan mucho tiempo de CPU, y tienen solicitudes de E/S especialmente cortas.

### Explique brevemente la planificación apropiativa (preemptive)

### Explique brevemente la planificación no apropiativa (no preemptive)

## Ejercicios:

### Observe la utilización de la CPU mediante el comando *monitor* en los sistemas Linux, macOS (¡Error! No se encuentra el origen de la referencia.) y UNIX

Parámetros más relevantes:

* + Tiempo de tránsito (Throughput).
  + Tiempo de proceso completo.
  + Tiempo de espera.
  + Tiempo de respuesta.

### Comando vmstat

Utilizando el comando **vmstat** que está disponible en los sistemas Linux, investigue la frecuencia en la que ocurren los cambios de contexto

Observe el comando y explique su salida

**vmstat 1 3**

Este comando proporciona 3 líneas de salida con un retraso de 1 segundo:

POR EJEMPLO:

24

225

339

Explicación:

* Desde que esta máquina arrancó, ha promediado 24 cambios de contexto por segundo. Y en el último segundo
* En el último segundo, se realizaron 225 cambios de contexto
* En l penúltimo segundo se realizaron 339 cambios de contexto

### Contenido del fichero status de un proceso

Utilice el sistema de archivos /proc para determinar la cantidad de cambios de contexto para un proceso determinado. Por ejemplo, el contenido del archivo /proc/2166/status enumerará varias estadísticas para el proceso con pid = 2166.

**cat /proc/2166/status**

Esta salida muestra el número de cambios de contexto durante la vida útil del proceso.

* Un cambio de contexto voluntario ocurre cuando un proceso ha cedido el control de la CPU porque requiere un recurso que actualmente no está disponible (como el bloqueo de E/S).
* Un cambio de contexto no voluntario ocurre cuando se le quita la CPU a un proceso, como cuando su intervalo de tiempo ha expirado o ha sido reemplazado por un proceso de mayor prioridad.

### Ejercicio sobre SJF

Considere tres procesos, todos llegando a la vez a la cola de “listo”, con un tiempo total de ejecución de 10, 20 y 30 unidades, respectivamente.

Cada proceso dedica el primer 20% del tiempo de ejecución a realizar E/S, el siguiente 70% del tiempo a realizar cálculos y el último 10% del tiempo a realizar E/S nuevamente. El sistema operativo utiliza un algoritmo de programación del tiempo de cálculo restante más corto(SJF) y despacha un nuevo proceso cuando el proceso en ejecución se bloquea en E/S o cuando el proceso en ejecución finaliza su ráfaga de cálculo. Suponga que todas las operaciones de E/S se pueden superponer tanto como sea posible. ¿Durante qué porcentaje de tiempo permanece inactiva la CPU?

### Ejercicio de algoritmo de planificación RR

Considere la siguiente tabla con el tiempo de llegada y el tiempo de ejecución de varios procesos en estado “listo”; un cuanto de 100 ms y un algoritmo de scheduling por turnos (RR)

|  |  |  |
| --- | --- | --- |
| **Process name** | **Arrival time** | **Execute time** |
| P0 | 0 | 250 |
| P1 | 50 | 170 |
| P2 | 130 | 75 |
| P3 | 190 | 100 |
| P4 | 210 | 130 |
| P5 | 350 | 50 |

# Capitulo 6

## Preguntas

### ¿Cuándo se establece la exclusión mutua?

### ¿Qué técnicas se usan en la exclusión mutua?

### ¿Qué es una red de Petri?

### **¿Por qué es útil Red de Petri en el diseño y estudio de sistemas** operativos?

### Diferencias entre cerrojos y semáforos

## Ejercicio

### Diseñar el modelo Productor Consumidor utilizando la herramienta para Redes de Petri PIPE (<https://github.com/sarahtattersall/PIPE/releases>)

### Crear un modelo de MUTEX (exclusión mutua) con Red de Petri

### Diseñar una Red de Petri que emule el problema de los filósofos cenando como modelo de 5 procesos concurrente que se disputan recursos comunes (tenedores). Explicar los resultados