

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. Capítulo 1. Introducción al Sistema Operativo2](#_Toc147790883)

[1. Explicar el propósito principal de un sistema operativo2](#_Toc147790884)

[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 ejemplos2](#_Toc147790885)

[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:2](#_Toc147790886)

[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 complejidad2](#_Toc147790887)

[5. Modelo conceptual del sistema operativo en una perspectiva de arriba hacia abajo. Ídem de abajo hacia arriba.2](#_Toc147790888)

[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.2](#_Toc147790889)

[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?2](#_Toc147790890)

[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”.2](#_Toc147790891)

[Rectifique adecuadamente ese falso enunciado2](#_Toc147790892)

[9. Explicar las diferencias claras entre driver de dispositivo y controlador de dispositivo.2](#_Toc147790893)

[10. Mediante el esquema que se acompaña, explicar el concepto IRQ en el SO2](#_Toc147790894)

[11. Diferencia entre trampa (TRAP) e interrupción utilizando la siguiente tabla:2](#_Toc147790895)

[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 terciario2](#_Toc147790896)

[13. Qué estructura de datos es la de la figura y ponga ejemplos donde el SO la emplea2](#_Toc147790897)

[14. Qué estructura de datos es la de la figura y ponga ejemplos donde el SO la emplea2](#_Toc147790898)

[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:2](#_Toc147790899)

[2. Capítulo 2: Llamadas y tipos de sistemas operativos.2](#_Toc147790900)

[16. ¿Cuál es el propósito del intérprete de comandos? ¿Por qué suele estar separado del núcleo?2](#_Toc147790901)

[17. Ventajas de CLI2](#_Toc147790902)

[18. Ventajas de GUI2](#_Toc147790903)

[19. ¿Cuál es el propósito de las llamadas al sistema?2](#_Toc147790904)

[3. Capítulo 32](#_Toc147790905)

[2. Preguntas2](#_Toc147790906)

[1. ¿Qué es un proceso del sistema operativo? ¿Qué recursos del sistema necesita un proceso para realizar satisfactoriamente su tarea?2](#_Toc147790907)

[2. ¿Cuales son las obligaciones del S.O. como gestor de procesos?2](#_Toc147790908)

[3. ¿Qué suele almacenarse en cada una de las secciones de la memoria ocupada por un proceso?2](#_Toc147790909)

[4. Se ha ejecutado el comando “size /bin/more”. Explicar la salida obtenida2](#_Toc147790910)

[5. Respecto a los estados de un proceso, rellene adecuadamente la tabla2](#_Toc147790911)

[6. ¿Qué es un PCB? ¿Qué información generalmente se suele guardar en un PCB?2](#_Toc147790912)

[3. Prácticas de comandos para procesos en Linux¡Error! Marcador no definido.](#_Toc147790913)

[NOTA. ¿Qué hay que hacer y registrar en porfolio?2](#_Toc147790914)

[1. Comando ps2](#_Toc147790915)

[2. Comando top2](#_Toc147790916)

[3. Comando pstree2](#_Toc147790917)

[4. Comando kill2](#_Toc147790918)

[4. Capitulo 42](#_Toc147790919)

[4. Preguntas2](#_Toc147790920)

[1. Se dice que existe una pseudosimultaneidad cuando:2](#_Toc147790921)

[2. Respecto a multithreading:2](#_Toc147790922)

[3. Se dispone de un ordenador monoprocesador que permite multiprogramación, para lo cual es preciso:2](#_Toc147790923)

[4. En relación con los Sistemas Operativos, ¿Qué se entiende por overhead?2](#_Toc147790924)

[5. Uno de los beneficios de la programación multiproceso (concurrente) pueden ser (marcar la respuesta FALSA):2](#_Toc147790925)

[6. Un kernel de un buen sistema operativo multinúcleo dispone de programas que deben resolver gestiones de multithreding como (SEÑALE LA FALSA):2](#_Toc147790926)

[7. La tecnología HyperThreading consiste en2](#_Toc147790927)

[8. Una tecnología que “engaña” al sistema operativo al hacerle creer que existen dos núcleos cuando en realidad solo existe uno2](#_Toc147790928)

[9. 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:2](#_Toc147790929)

[10. El soporte para subprocesos se puede proporcionar a nivel de usuario, o por el kernel2](#_Toc147790930)

[11. El modelo de gestión de subprocesos del SO de tipo muchos a uno:2](#_Toc147790931)

[12. El modelo de gestión de subprocesos del SO de tipo uno a uno:2](#_Toc147790932)

[5. Ejercicios2](#_Toc147790933)

[1. Comandos sobre información de CPU2](#_Toc147790934)

[2. ¿Cómo sabemos si tenemos activado el Hyper Threading?2](#_Toc147790935)

[3. ¿Cómo contar el número de Threads que está utilizando un proceso de Linux?2](#_Toc147790936)

[5. Capitulo 52](#_Toc147790937)

[6. Preguntas2](#_Toc147790938)

[1. 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 contexto2](#_Toc147790939)

[2. 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:2](#_Toc147790940)

[3. 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:2](#_Toc147790941)

[4. Explique brevemente la planificación apropiativa (preemptive)2](#_Toc147790942)

[5. Explique brevemente la planificación no apropiativa (no preemptive)2](#_Toc147790943)

[7. Ejercicios2](#_Toc147790944)

[1. Observe la utilización de la CPU mediante el comando *monitor* en los sistemas Linux, macOS (Figura 4) y UNIX2](#_Toc147790945)

[2. Comando vmstat2](#_Toc147790946)

[3. Contenido del fichero status de un proceso2](#_Toc147790947)

[4. Ejercicio sobre SJF2](#_Toc147790948)

[5. Ejercicio de algoritmo de planificación RR2](#_Toc147790949)

[6. Capitulo 62](#_Toc147790950)

[8. Preguntas2](#_Toc147790951)

[1. ¿Cuándo se establece la exclusión mutua?2](#_Toc147790952)

[2. ¿Qué técnicas se usan en la exclusión mutua?2](#_Toc147790953)

[3. ¿Qué es una red de Petri?2](#_Toc147790954)

[4. ¿Por qué es útil Red de Petri en el diseño y estudio de sistemas operativos?2](#_Toc147790955)

[5. Diferencias entre cerrojos y semáforos2](#_Toc147790956)

[9. Ejercicio2](#_Toc147790957)

[1. Diseñar el modelo Productor Consumidor utilizando la herramienta para Redes de Petri PIPE (https://github.com/sarahtattersall/PIPE/releases)2](#_Toc147790958)

[2. Crear un modelo de MUTEX (exclusión mútua) con Red de Petri2](#_Toc147790959)

[3. 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 resultados2](#_Toc147790960)

[7. Capitulo 72](#_Toc147790961)

[8. Capitulo 82](#_Toc147790962)

[10. Preguntas2](#_Toc147790963)

[1. ¿Qué función realiza un controlador de dispositivo en un sistema operativo?2](#_Toc147790964)

[2. ¿Qué es la E/S por bloques en sistemas operativos?2](#_Toc147790965)

[3. ¿Qué es una interrupción de hardware en un sistema operativo?2](#_Toc147790966)

[4. ¿Qué significa DMA en el contexto de E/S en sistemas operativos?2](#_Toc147790967)

[5. ¿Cuál de las siguientes no es una función de un controlador de dispositivo?2](#_Toc147790968)

[6. ¿Qué es el pooling de E/S en sistemas operativos?2](#_Toc147790969)

[7. ¿Cuál es el propósito principal de las interrupciones de software en un sistema operativo?2](#_Toc147790970)

[8. ¿Qué tipo de interrupción es generada por dispositivos de hardware para notificar eventos como la finalización de una transferencia de datos?2](#_Toc147790971)

[9. ¿Cuál es el propósito de una tabla de vectores de interrupción en un sistema operativo?2](#_Toc147790972)

[10. ¿Qué beneficio proporcionan las interrupciones en un sistema operativo?2](#_Toc147790973)

[11. Ejercicios2](#_Toc147790974)

[1. Comando para ver interrupciones2](#_Toc147790975)

[2. Comando para ver una lista de interrupciones2](#_Toc147790976)

[3. Comando para ver la afinidad2](#_Toc147790977)

[4. Lista de los núcleos de CPU a los que se puede asignar una interrupción2](#_Toc147790978)

[5. Dispositivos PCI en el sistema2](#_Toc147790979)

[6. Módulos kernel cargados en el sistema2](#_Toc147790980)

[7. Lista de los dispositivos en el directorio /dev2](#_Toc147790981)

[8. Lista de dispositivos USB conectados al sistema2](#_Toc147790982)

[9. Detección y carga de controladores de dispositivos2](#_Toc147790983)

[10. Configuración de hardware del sistema, incluyendo la lista de dispositivos y controladores asociados2](#_Toc147790984)

[11. Información sobre un módulo del kernel, incluyendo detalles sobre el controlador de dispositivo2](#_Toc147790985)

[12. Información sobre los dispositivos de bloque2](#_Toc147790986)

[9. Capitulo 92](#_Toc147790987)

[1. Preguntas2](#_Toc147790988)

[1.1. ¿Qué es un inodo en un sistema de archivos Unix o Linux?2](#_Toc147790989)

[1.2. ¿Cuál es el propósito principal de la técnica de "journaling" en un sistema de archivos?2](#_Toc147790990)

[1.3. ¿Cuál es la principal diferencia entre un sistema de archivos FAT32 y NTFS en Windows?2](#_Toc147790991)

[1.4. ¿Cuál de los siguientes sistemas de archivos es más adecuado para sistemas de archivos grandes y aplicaciones de alto rendimiento en Linux?2](#_Toc147790992)

[1.5. ¿Qué hace el comando `tee` en Linux?2](#_Toc147790993)

[1.6. ¿Cuál es el propósito principal de un sistema de archivos en red (NFS) en Linux?2](#_Toc147790994)

[1.7. ¿Qué es el Sistema de Archivos Virtual (VFS) en Linux?2](#_Toc147790995)

[1.8. ¿Cuál es el sistema de archivos predeterminado en sistemas macOS más recientes?2](#_Toc147790996)

[1.9. ¿Qué tipo de sistema de archivos se utiliza comúnmente en discos ópticos como CD-ROMs y DVDs?2](#_Toc147790997)

[1.10. ¿Cuál es el propósito de un sistema de archivos comprimido?2](#_Toc147790998)

[1.11. ¿Qué hace el comando df en sistemas Unix o Linux?2](#_Toc147790999)

[1.12. En un sistema de archivos Unix o Linux, ¿cuál es el propósito principal del directorio /home?2](#_Toc147791000)

[1.13. Algunos sistemas operativos proporcionan un cambio de nombre de llamada al sistema para darle un nuevo nombre a un archivo. ¿Existe alguna diferencia entre usar esta llamada para cambiar el nombre de un archivo y simplemente copiar el archivo a un archivo nuevo con un nuevo nombre y luego eliminar el anterior?2](#_Toc147791001)

[2. Ejercicios2](#_Toc147791002)

[2.1. Pruebe mostrar las particiones de disco en Linux, con el comando Fdisk con "-1" (como se indica a continuación) como usuario sudo o root.2](#_Toc147791003)

[2.2. Mosttytrar el nombre del fichero de la terminal conectada a la salida estándar y hacer una prueba para mandarle caracteres.2](#_Toc147791004)

[2.3. Ejecutar el comando w (para obtener información de los terminales activos).2](#_Toc147791005)

[2.4. Probar el comando df y comentar la salida:2](#_Toc147791006)

[2.5. Listar información sistema de archivos Linux2](#_Toc147791007)

[2.6. Comprobar el espacio en disco en Linux usando el comando2](#_Toc147791008)

[10. Capitulo 102](#_Toc147791009)

Capítulo 1. Introducción al Sistema Operativo

Preguntas

* 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.

* 1. 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.

* 1. 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.

* 1. 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.

* 1. Modelo conceptual del sistema operativo en una perspectiva de arriba hacia abajo. Ídem de abajo hacia arriba.
  2. 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.

* 1. 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.

* 1. 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”.

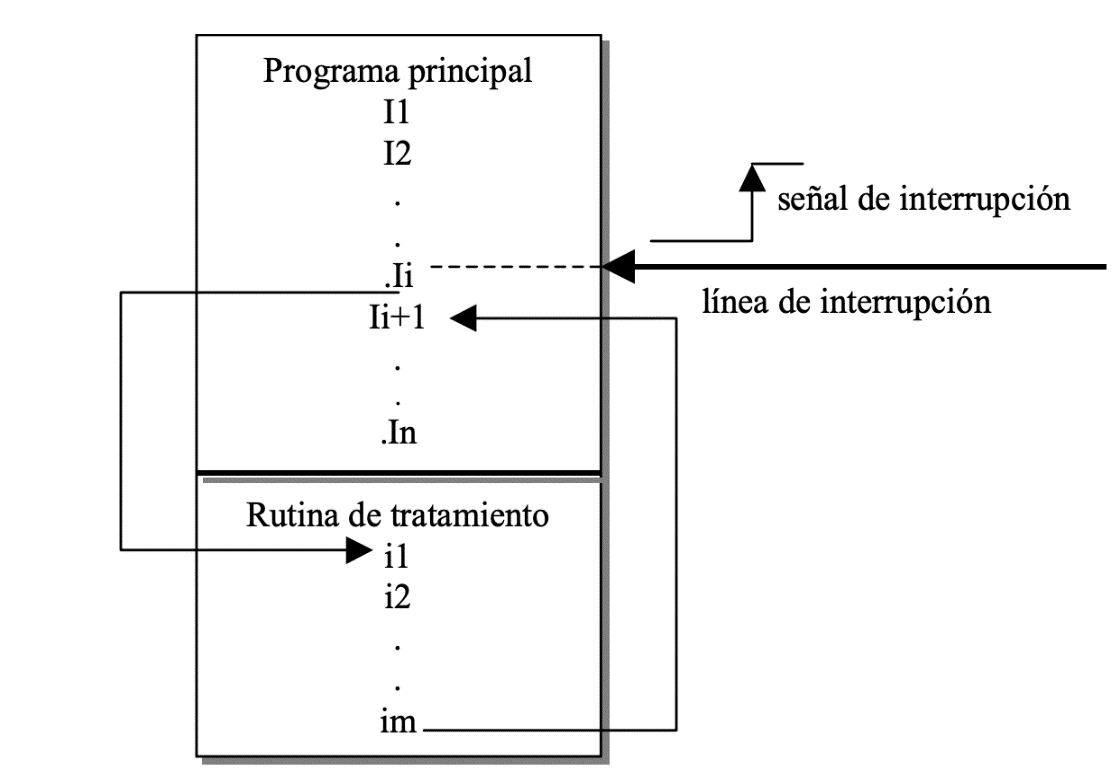
## Rectifique adecuadamente ese falso enunciado

“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”.

* 1. 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.

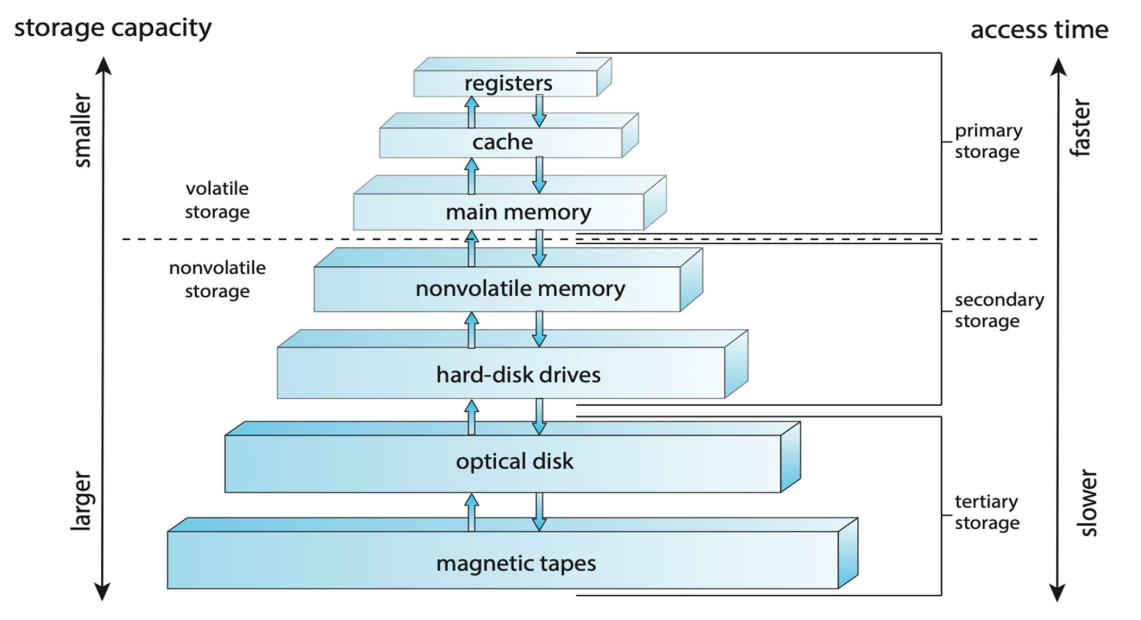
* 1. Mediante el esquema que se acompaña, explicar el concepto IRQ en el SO

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.

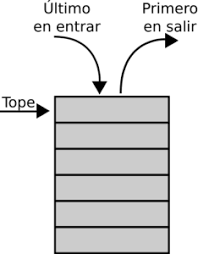
* 1. Diferencia entre trampa (TRAP) e interrupción utilizando la siguiente tabla:

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

* 1. 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

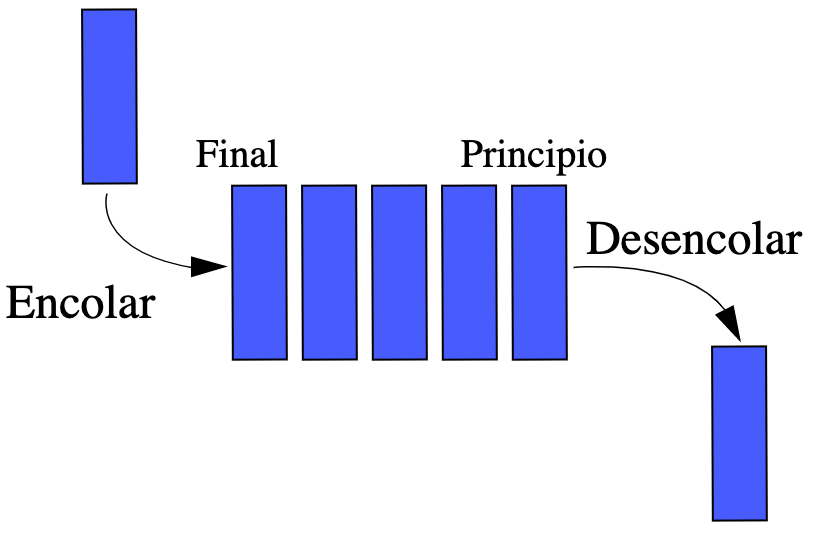


* 1. Qué estructura de datos es la de la figura y ponga ejemplos donde el SO la emplea



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.

* 1. 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.

* 1. 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:

* 1. ¿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.

* 1. 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.
  1. 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.

* 1. ¿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.

* 1. 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.

* 1. 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.

* 1. ¿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".

* 1. ¿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.
  1. 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.

* 1. ¿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. |

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

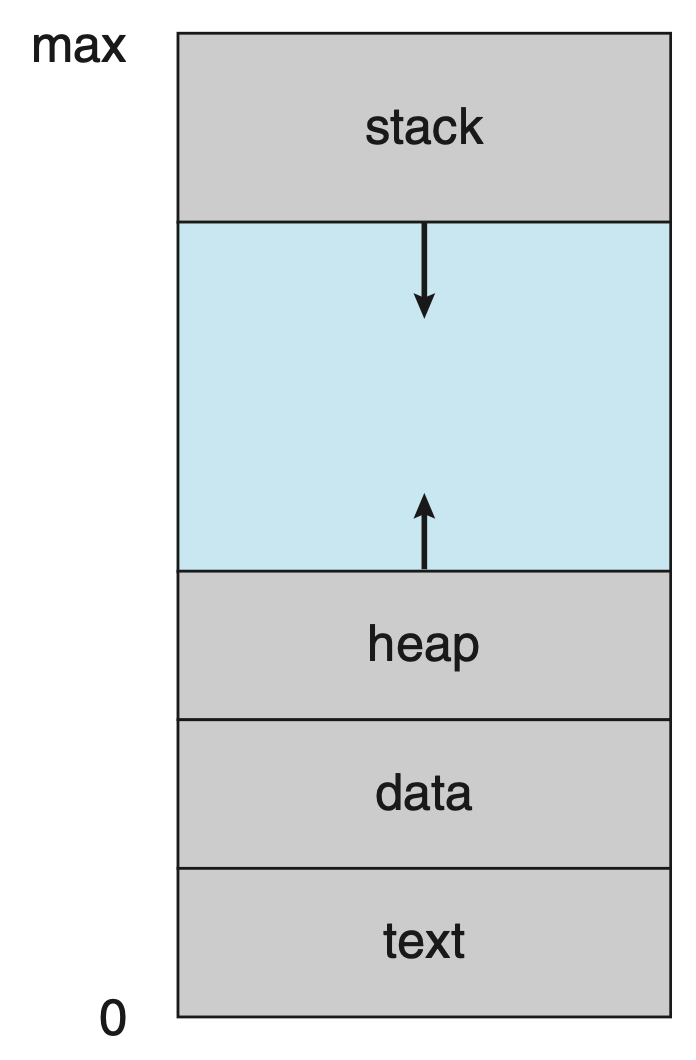
|  |  |  |
| --- | --- | --- |
|  | **Monolítico con Módulos Recargables** | **Híbrido** |
| Arquitectura |  |  |
| Espacio del S.O. en modo usuario |  |  |
| Espacio del S.O. en modo kernel |  |  |
| Modularidad |  |  |
| Modo de operación de los módulos |  |  |
| Riesgo |  |  |
| Coste de diseño |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

# Capítulo 3. Procesos

Preguntas

* 1. ¿Qué es un proceso del sistema operativo? ¿Qué recursos del sistema necesita un proceso para realizar satisfactoriamente su tarea?
  2. ¿Cuales son las obligaciones del S.O. como gestor de procesos?

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



* 1. 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

* 1. Respecto a los estados de un proceso, rellene adecuadamente la tabla

|  |  |
| --- | --- |
| **Estado** | **Descripción** |
| Nuevo |  |
| Preparado |  |
| Activo |  |
| Bloqueado |  |
| Terminado |  |

* 1. ¿Qué es un PCB? ¿Qué información generalmente se suele guardar en un PCB?

Ejercicios

## NOTA. ¿Qué hay que hacer y registrar en porfolio?

**Probar sobre Ubuntu los comandos que se describen en los párrafos siguientes. Incluir los resultados en el porfolio mediante el texto o volcado de pantalla. Añadir las observaciones necesarias.**

## Comando ps

‘ **ps**’, listará (de múltiples formas según las opciones que le pasemos) todos los procesos que se encuentran corriendo en nuestro equipo.

ps [opciones]

Podemos **revisar el manual de ps**dentro del sistema para conocer todas las opciones posibles:

man ps

Siendo las opciones más habituales :

###### ps aux (muestra todos los procesos del sistema)

###### ps axjf (que mostrará un árbol jerárquico con la ruta del programa al que pertenece el proceso)

Las opciones que podemos aplicar a **ps** no van más allá de mostrar la información de una u otra forma, más o menos extensa, o como es habitual, filtrar los resultados con **grep.** Sea cual sea el método de muestra que elijamos, siempre habrá dos constantes, el PID y el comando o nombre del programa. Aquí un ejemplo de filtrado sobre ps para obtener únicamente los procesos pertenecientes a bash.

ps aux | grep bash

El **PID**es el **número identificador de proceso**que le asigna el sistema a cada proceso que se inicia, mientras que el **command**es el programa al cual pertenece dicho proceso.

## Comando top

Otro gestor de procesos integrado en la mayoría de sistemas Linux es **top**. Mientras que ps nos muestra un listado de procesos estático, es decir, nos informa de los procesos, nombres, usuarios o recursos que se están usando en el momento de la petición; **top nos da un informe en tiempo real de los mismos.**

top

El comando Top ofrece un conjunto de estadísticas sobre el estado general del sistema, así como el "top" de los procesos más activos en el sistema. La lista se presenta y actualiza por defecto cada 2 segundos.

Los datos que presenta TOP son:

* Utilización de CPU (%);
* información sobre el “top” de los procesos;
* Uso de memoria RAM;
* Número de usuarios registrados en el sistema;
* Información de la memoria SWAP;
* Número de tareas;
* PID de tareas (número que se refiere a tareas).

El manual se pude ver con: man top

Aquí, como vemos en su manual, podemos controlar más aspectos, como los de los siguientes ejemplos entre otros:

top –d 5 (Donde 5 es el número de segundos a transcurrir entre cada muestreo)

top –o %CPU (Donde %CPU es el valor por el que vamos a **ordenar los procesos**)

top –u toushiro (Donde toushiro es el usuario del cual queremos mostrar los procesos)

Otro gestor de procesos parecido a top es ‘ **htop**’, El comando HTOP es una versión mejorada del comando TOP. La información es similar a la mostrada por el comando TOP, pero presentada en una interfaz más intuitiva.

htop

Cada una de las marcas `|` está codificada por colores para proporcionar un significado diferente. Para el uso de la CPU, la clave de color es:

Azul: procesos de baja prioridad (agradable > 0)

Verde: procesos normales (de usuario).

Rojo: procesos del kernel.

Ambos comandos top y htop verifican las mismas actividades, pero el uso de htop es mucho más apropiado, debido a su colorida e intuitiva “interfaz” (o salida).

Hay distribuciones de Linux que no traen htop, pero se puede encontrar con el comando:

apt-get install htop

## Comando pstree

El comando pstree, nos permitirá ver el **árbol de procesos activos**, en nuestros sistemas GNU Linux.

Difiere del popular ‘ps’ en varios aspectos. Entre los que destaca que este último muestra los procesos en una lista en lugar de un diagrama, pero proporciona información más detallada sobre estos.

El programa pstree facilita información sobre la finalización de una serie de procesos relacionados entre sí, esto es, todos los descendientes de un proceso particular. El programa deja claro desde un principio que proceso es el primario y cuales son los secundarios. Esto evita buscar las relaciones entre los diversos procesos de manera manual.

Su forma más común de uso es, desde la línea de comandos, la siguiente:

**pstree [opciones]**

Veamos una salida utilizando el comando sin parámetros:

Con el parámetro “**-a**” , nos muestra la línea de comandos utilizada. Por ejemplo, si el comando utiliza la ruta a un fichero de configuración.

Por defecto se inhabilita la visualización en el árbol de los nombres repetidos. Para evitar esto, podemos utilizar el parámetro “**-c**”

Si nos interesa podemos ver el árbol de un proceso específico, de la siguiente manera:

Un ejemplo sería:

Utilizando el parámetro “**-g**”, veremos los procesos agrupados por grupo.

Los datos de salida del programa están ordenados de manera alfabética. En cambio, podemos indicar que la salida sea basada en los números de PID, utilizando el parámetro “**-n**”

Veamos una combinación de varios de los parámetros:

De forma fácil podemos ver el árbol de procesos de un usuario único, de la siguiente manera, mediante la opción «-u» :

**pstree** -ucgn snmp

## Comando kill

Los sistemas Linux vienen con la herramienta **KILL**instalada, que usaremos para detener los procesos que necesitemos. Por defecto el comando kill envía una señal denominada TERM a un proceso que le pasaremos mediante su **PID**como argumento. Esta señal TERM pedirá a dicho proceso que termine, permitiéndole gestionar su función de cierre, completando las tareas necesarias y limpiando la información que ha cargado en memoria.

kill [PID del proceso]

En la captura de aquí arriba vemos cómo nos ha dado un error que aprovechamos para destacar un punto muy importante en la seguridad de los sistemas Linux, sistemas verdaderamente multiusuario y bien definidos, que no permite eliminar o cancelar procesos de otros usuarios.

En el caso de encontrarnos ante un proceso que “no quiere cerrarse” por la vía TERM, pasaremos a eliminar dicho proceso por la fuerza ejecutando el comando kill con el siguiente argumento, ejecutando como root:

kill –KILL [PID del proceso]

Con este último comando, no estamos mandado al proceso ninguna señal, directamente estamos diciéndole al kernel del sistema que descarte y cierre dicho proceso.

Estas señales también pueden ser identificadas con números. Por ejemplo, en los ejemplos anteriores **TERM**puede ser pasada al proceso mediante **“-15”**y **–KILL**es el equivalente a pasar **“-9”**. Es decir, el resultado de los siguientes comandos será el mismo:

kill -9 [PID del proceso]

kill –KILL [PID del proceso]

El comando kill además de para finalizar procesos, también podemos usarlo para reiniciar ciertos servicios. Uno de los que más necesita reiniciarse suele ser Apache (servidor web), sobre todo si aún estamos con la configuración base, para ir viendo que todo funciona correctamente.

Al igual que Apache, multitud de servicios necesitan ser reiniciados, y la mayoría de ellos responde al argumento **‘HUP’ (Hang up)**de kill. Mediante el siguiente comando, el servicio perteneciente a Apache, se reiniciará y volverá a cargar el fichero de configuración, permitiéndonos ver si los cambios han surtido efecto y volviendo a dar servicio a los usuarios.

kill –HUP [PID de Apache]

HUP también tiene su respectiva nomenclatura en numeración, siendo el equivalente al comando anterior, la siguiente línea:

kill -1 [PID de Apache]

Un dato importante es que además de por su PID, si conocemos el nombre exacto del proceso también podemos usarlo en el lugar en el que usaríamos el PID. Para esto usaremos ‘ **pkill**’ en lugar de kill, que funciona exactamente igual, pero preparado para trabajar con nombres de proceso en lugar de con PID. Es decir estos dos comandos harán exactamente lo mismo:

kill -9 3484

pkill -9 htop

' **killall**' es una variante del comando kill con el que enviaremos la misma señal a todos los procesos pertenecientes a un programa. Por ejemplo:

killall Firefox

Con estos comandos y herramientas ya podremos gestionar de forma correcta y eficiente los procesos de nuestro sistema, monitorizándolos para ver si hay algo que no debiese estar, o que se encuentre consumiendo recursos por encima de lo normal; optimizando así nuestra distribución y el aprovechamiento que hacemos de nuestro hardware.

# Capitulo 4

Preguntas

* 1. 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
   1. Respecto a multithreading:
5. La concurrencia o multithreading de una aplicación depende exclusivamente de la gestión del SO
6. El efecto de concurrencia o multithreading de una aplicación depende exclusivamente del programador que la crea.
7. La simultaneidad real de una aplicación creada con hilos o tareas concurrentes necesita que la CPU sea multinúcleo.
8. 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?

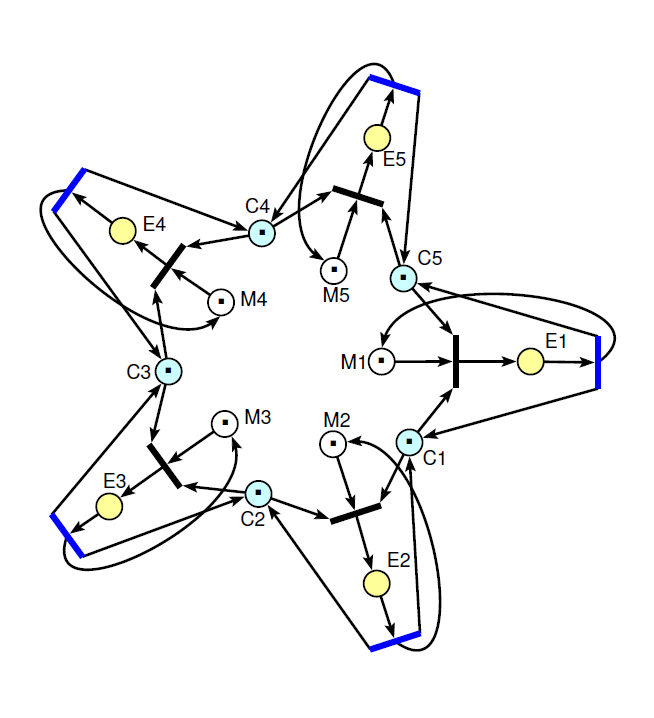
## Diferencias entre cerrojos y semáforos

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 mútua) 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



C1, C2, C3, C4 y C5 son los tenedores. Si están marcados están disponibles.

M1, M2, M3, M4 y M5 estado PENSAR de los filósofos 1, 2, 3, 4 y 5. Si están marcados están pensando

E1, E2, E3, E4 y E5 estado COMER de los filósofos 1, 2, 3, 4 y 5. Si están marcados están comiendo.

# Capitulo 7

**1 Mencione dos diferencias entre direcciones lógicas y físicas.**

**2 ¿Por qué los tamaños de página son siempre potencias de 2?**

**3 ¿Cuáles son las funciones de una MMU?**

**4 Considere un espacio de direcciones lógicas de 64 páginas de 1024 palabras cada una, mapeadas en una memoria física de 32 marcos.**

**a. ¿Cuántos bits hay en la dirección lógica?**

**b. ¿Cuántos bits hay en la dirección física?**

**5 ¿Cuál es el efecto de permitir que dos entradas en una tabla de páginas apunten al mismo marco de página en la memoria? Explique cómo podría usarse este efecto para disminuir la cantidad de tiempo necesario para copiar una gran cantidad de memoria de un lugar a otro. ¿Qué efecto tendría la actualización de algún byte en una página en la otra página?**

**6 Suponiendo un tamaño de página de 1 KB, ¿cuáles son los números de página y los desplazamientos para las siguientes referencias de direcciones (proporcionadas como números decimales):**

**a. 3085**

**b. 42095**

**C. 215201**

**d. 650000**

**e. 2000001**

**7 Considere un espacio de direcciones lógicas de 256 páginas con un tamaño de página de 4 KB, mapeado en una memoria física de 64 marcos.**

**a. ¿Cuántos bits se requieren en la dirección lógica?**

**b. ¿Cuántos bits se requieren en la dirección física?**

**8 Un ordenador emplea 32 bits para las direcciones virtuales de 32 bits. Dispone de una una memoria principal de 512 MB, que emplea páginas de 4 KB. Indique el formato de la dirección virtual (número de bits para páginas y bits para desplazamiento) y el número de marcos de la página de este ordenador.**

**9 ¿Qué es la memoria SWAP? ¿Dónde se encuentra la memoria swap? ¿cuando es necesaria? ¿Qué inconvenientes tiene?**

EJERCICIOS

1. ¿Por qué Windows consume tanta RAM? (superfetch)

(buena respuesta en ChatGPT)

1. ¿Qué es la presión de la memoria y cómo puedo aliviarla?

(buena respuesta en ChatGPT)

# Capitulo 8

Preguntas

## ¿Qué función realiza un controlador de dispositivo en un sistema operativo?

a) Administrar recursos de CPU

b) Gestionar la memoria principal

c) Interactuar con dispositivos de hardware

d) Controlar el planificador de tareas

## ¿Qué es la E/S por bloques en sistemas operativos?

a) Un método para E/S donde los datos se transfieren uno a uno.

b) Un método para E/S donde los datos se transfieren en bloques más grandes.

c) Un método para E/S que no se utiliza en sistemas modernos.

d) Un método para E/S que solo funciona con dispositivos de almacenamiento.

## ¿Qué es una interrupción de hardware en un sistema operativo?

a) Una solicitud de un programa en ejecución.

b) Una señal generada por un dispositivo de hardware para solicitar atención del sistema operativo.

c) Un error en el código del sistema operativo.

d) Un proceso de planificación de tareas.

## ¿Qué significa DMA en el contexto de E/S en sistemas operativos?

a) Acceso Directo a la Memoria

b) Data Management Algorithm

c) Device Monitoring Application

d) Data Movement Accelerator

## ¿Cuál de las siguientes no es una función de un controlador de dispositivo?

a) Traducir comandos del sistema operativo en operaciones de hardware.

b) Administrar la memoria principal del sistema.

c) Gestionar recursos compartidos.

d) Optimizar el rendimiento del dispositivo.

## ¿Qué es el pooling de E/S en sistemas operativos?

a) Una técnica para gestionar recursos de CPU.

b) Una técnica para compartir recursos de memoria principal.

c) Una técnica para administrar recursos de E/S compartidos de manera eficiente.

d) Una técnica para dividir tareas de E/S en bloques.

## ¿Cuál es el propósito principal de las interrupciones de software en un sistema operativo?

a) Solicitar la atención de un dispositivo de hardware.

b) Iniciar una rutina específica en el sistema operativo para realizar una tarea.

c) Notificar un error en el sistema operativo.

d) Detener la ejecución del programa actual.

## ¿Qué tipo de interrupción es generada por dispositivos de hardware para notificar eventos como la finalización de una transferencia de datos?

a) Interrupción de reloj

b) Interrupción de software

c) Interrupción de tiempo real

d) Interrupción de E/S

## ¿Cuál es el propósito de una tabla de vectores de interrupción en un sistema operativo?

a) Almacenar datos de entrada y salida.

b) Asignar números únicos a los dispositivos de hardware.

c) Gestionar la memoria principal.

d) Priorizar las tareas del sistema operativo.

## ¿Qué beneficio proporcionan las interrupciones en un sistema operativo?

a) Mejoran la eficiencia en la administración de memoria.

b) Aceleran la velocidad de la CPU.

c) Evitan la necesidad de controladores de dispositivo.

d) Permiten una gestión eficiente de recursos y eventos.

Ejercicios

Pruebe y comente la salida de los comandos dados a continuación relacionados con E/S.

Seguramente muchos de ellos necesiten privilegios SUDO

## Comando para ver interrupciones

En Linux, para ver información relacionada con las interrupciones, utilizar el comando **cat /proc/interrupts**

Presentar un volcado de la salida y comentar esa información.

## Comando para ver una lista de interrupciones

Deseamos obtener más información detallada sobre las interrupciones y su asignación en Linux, para ello utilizamos el comando:

**ls /proc/irq**

Comente la salida y observando cómo se puede navegar a un directorio específico para obtener detalles sobre una interrupción concreta.

## Comando para ver la afinidad

**cat /proc/irq/[Número\_IRQ]/smp\_affinity**

Use este comando para ver o cambiar la afinidad de una interrupción a un núcleo de CPU específico.

Por ejemplo, para ver la afinidad de la IRQ 16:

**cat /proc/irq/16/smp\_affinity**

## Lista de los núcleos de CPU a los que se puede asignar una interrupción

**cat /proc/irq/[Número\_IRQ]/irq\_affinity\_list**

Este comando muestra una lista de los núcleos de CPU a los que se puede asignar una interrupción en sistemas con múltiples núcleos.

Por ejemplo:

**cat /proc/irq/16/irq\_affinity\_list**

## Dispositivos PCI en el sistema

**lspci -v**

Este comando muestra información detallada sobre los dispositivos PCI en el sistema, incluyendo las interrupciones que utilizan. Se utiliza para ver qué IRQ está asignada a cada dispositivo.

Por ejemplo:

**lspci -v**

## Módulos kernel cargados en el sistema

**lsmod**

Muestra la lista de módulos del kernel cargados en el sistema. Los módulos del kernel a menudo son controladores de dispositivos.

Ejemplo:

**lsmod**

## Lista de los dispositivos en el directorio /dev

ls /dev: Lista los dispositivos en el directorio /dev. Los nombres de los archivos en este directorio representan dispositivos de hardware y controladores de dispositivos.

Ejemplo:

**ls /dev**

## Lista de dispositivos USB conectados al sistema

**lsusb**

Muestra una lista de dispositivos USB conectados al sistema. Esto es especialmente útil para identificar dispositivos USB y buscar información sobre ellos.

Ejemplo:

**lsusb**

## Detección y carga de controladores de dispositivos

dmesg

Muestra el registro del kernel, que incluye información sobre la detección y carga de controladores de dispositivos durante el arranque del sistema.

Ejemplo:

dmesg | less # Para desplazarse a través del registro

## Configuración de hardware del sistema, incluyendo la lista de dispositivos y controladores asociados

**lshw**

Muestra información detallada sobre la configuración de hardware del sistema, incluyendo la lista de dispositivos y controladores asociados.

**lshw**

## Información sobre un módulo del kernel, incluyendo detalles sobre el controlador de dispositivo

modinfo

Proporciona información sobre un módulo del kernel, incluyendo detalles sobre el controlador de dispositivo y los parámetros del módulo.

Ejemplo:

**modinfo nombre\_del\_modulo**

## Información sobre los dispositivos de bloque

lsblk

Muestra información sobre los dispositivos de bloque (por ejemplo, discos duros y particiones) en el sistema, incluyendo información sobre los controladores y las rutas de los dispositivos.

Ejemplo:

**lsblk**

# Capitulo 9

Preguntas

## ¿Qué es un inodo en un sistema de archivos Unix o Linux?

a) Un archivo de registro de transacciones.

b) Un sistema de archivos comprimido.

c) Una estructura de datos que almacena metadatos de archivos y directorios.

d) Un sistema de archivos de red distribuye metadatos de archivos y directorios.

## ¿Cuál es el propósito principal de la técnica de "journaling" en un sistema de archivos?

a) Reducir el tamaño de los archivos.

b) Mejorar la velocidad de lectura de archivos.

c) Mejorar la integridad de los datos y facilitar la recuperación después de fallos inesperados.

d) Comprimir archivos para ahorrar espacio en disco.

## ¿Cuál es la principal diferencia entre un sistema de archivos FAT32 y NTFS en Windows?

a) FAT32 admite journaling, mientras que NTFS no lo hace.

b) FAT32 es más rápido en la lectura de archivos grandes.

c) NTFS ofrece características avanzadas como journaling y permisos de acceso.

d) FAT32 es más eficiente en el uso del espacio en disco.

## ¿Cuál de los siguientes sistemas de archivos es más adecuado para sistemas de archivos grandes y aplicaciones de alto rendimiento en Linux?

a) Ext4

b) FAT32

c) XFS

d) HFS+

## ¿Qué hace el comando `tee` en Linux?

a) Calcula la cantidad total de espacio en disco utilizado por archivos.

b) Redirige la salida de un comando hacia un archivo y muestra la salida en la terminal.

c) Monta un sistema de archivos en el sistema.

d) Comprime archivos y directorios en un archivo ZIP.

## ¿Cuál es el propósito principal de un sistema de archivos en red (NFS) en Linux?

a) Mejorar la velocidad de acceso a archivos locales.

b) Proporcionar una interfaz gráfica para gestionar archivos y directorios.

c) Comprimir archivos para ahorrar ancho de banda en la red.

d) Compartir archivos y directorios entre sistemas en una red.

## ¿Qué es el Sistema de Archivos Virtual (VFS) en Linux?

a) Una capa de abstracción que proporciona una interfaz unificada para el acceso a sistemas de archivos de diferentes tipos.

b) Un sistema de archivos de red.

c) Un sistema de archivos comprimido.

d) Un sistema de archivos exclusivo de macOS.

## ¿Cuál es el sistema de archivos predeterminado en sistemas macOS más recientes?

a) FAT32

b) NTFS

c) HFS+

d) APFS

## ¿Qué tipo de sistema de archivos se utiliza comúnmente en discos ópticos como CD-ROMs y DVDs?

a) Ext4

b) ISO 9660

c) NTFS

d) XFS

## ¿Cuál es el propósito de un sistema de archivos comprimido?

a) Mejorar la velocidad de lectura de archivos.

b) Ahorrar espacio en disco mediante la reducción del tamaño de los archivos.

c) Facilitar la recuperación de datos después de un fallo del sistema.

d) Aumentar la seguridad de los archivos.

## ¿Qué hace el comando df en sistemas Unix o Linux?

a) Lista los archivos en un directorio específico.

b) Calcula el tamaño de un archivo.

c) Muestra información sobre el uso de espacio en disco y sistemas de archivos montados.

d) Desmonta un sistema de archivos.

## En un sistema de archivos Unix o Linux, ¿cuál es el propósito principal del directorio /home?

a) Almacenar copias de seguridad del sistema.

b) Almacenar archivos temporales del sistema.

c) Almacenar los directorios personales de los usuarios.

d) Almacenar archivos de sistema críticos.

## Algunos sistemas operativos proporcionan un cambio de nombre de llamada al sistema para darle un nuevo nombre a un archivo. ¿Existe alguna diferencia entre usar esta llamada para cambiar el nombre de un archivo y simplemente copiar el archivo a un archivo nuevo con un nuevo nombre y luego eliminar el anterior?

Copiar el archivo implica el duplicado de los datos en disco, con la consiguiente posibilidad de errores y tiempo invertido.

Ejercicios

Pruebe y comente la salida de los comandos dados a continuación relacionados con E/S.

Seguramente muchos de ellos necesiten privilegios SUDO

## Pruebe mostrar las particiones de disco en Linux, con el comando Fdisk con "-1" (como se indica a continuación) como usuario sudo o root.

**$ sudo fdisk -1**

a08@a08-HP-ProDesk-600-G5-SFF:~$ sudo fdisk -l

Disco /dev/sda: 447,13 GiB, 480103981056 bytes, 937703088 sectores

Disk model: CT480BX500SSD1

Unidades: sectores de 1 \* 512 = 512 bytes

Tamaño de sector (lógico/físico): 512 bytes / 512 bytes

Tamaño de E/S (mínimo/óptimo): 512 bytes / 512 bytes

Tipo de etiqueta de disco: gpt

Identificador del disco: FDCBEDED-6450-4811-834E-D648F23F15B9

Dispositivo Comienzo Final Sectores Tamaño Tipo

/dev/sda1 2048 1050623 1048576 512M Sistema EFI

/dev/sda2 1050624 937701375 936650752 446,6G Sistema de ficheros de Linux

Disco /dev/sdb: 931,51 GiB, 1000204886016 bytes, 1953525168 sectores

Disk model: WDC WD10EZEX-60W

Unidades: sectores de 1 \* 512 = 512 bytes

Tamaño de sector (lógico/físico): 512 bytes / 4096 bytes

Tamaño de E/S (mínimo/óptimo): 4096 bytes / 4096 bytes

Tipo de etiqueta de disco: gpt

Identificador del disco: 9BD5C14D-BEB7-4813-9816-A618FE19853E

Dispositivo Comienzo Final Sectores Tamaño Tipo

/dev/sdb1 2048 1953523711 1953521664 931,5G Sistema de ficheros de Linux

Disco /dev/sdc: 28,85 GiB, 30979129344 bytes, 60506112 sectores

Disk model: TransMemory

Unidades: sectores de 1 \* 512 = 512 bytes

Tamaño de sector (lógico/físico): 512 bytes / 512 bytes

Tamaño de E/S (mínimo/óptimo): 512 bytes / 512 bytes

Tipo de etiqueta de disco: dos

Identificador del disco: 0x94cb125e

Dispositivo Inicio Comienzo Final Sectores Tamaño Id Tipo

/dev/sdc1 8064 60506111 60498048 28,8G 7 HPFS/NTFS/exFAT

## Mosttytrar el nombre del fichero de la terminal conectada a la salida estándar y hacer una prueba para mandarle caracteres.

nombre del fichero de la terminal conectada a la salida estándar:

#  **tty**

/dev/ttys000

Como en Unix todo son ficheros, se puede usar esto para comunicarse entre los diferentes terminales, en este caso imprimirá la palabra hola por la terminal tty

# echo hola > /dev/ttys000

hola

a08@a08-HP-ProDesk-600-G5-SFF:~$ echo hola > /dev/pts/0

hola

## Ejecutar el comando [w](https://francisconi.org/linux/comandos/nmap) (para obtener información de los terminales activos).

# w

19:55 up 10 days, 1:20, 2 users, load averages: 1,70 1,67 1,70

USER TTY FROM LOGIN@ IDLE WHAT

david console - 25sep23 10days -

david s000 - 6:52 - w

También con el comando who:

# who

david console Sep 25 17:45

david ttys000 Oct 5 06:52

a08@a08-HP-ProDesk-600-G5-SFF:~$ w

13:11:45 up 1:25, 1 user, load average: 0,36, 0,46, 0,52

USUARIO TTY DESDE LOGIN@ IDLE JCPU PCPU WHAT

a08 tty2 tty2 11:46 1:25m 0.01s 0.00s /usr/libexec/gdm-wayland-session env GNOME\_SHELL\_SESSION\_MODE=ubuntu /usr/bin/gnome-session --session=ubuntu

a08@a08-HP-ProDesk-600-G5-SFF:~$ who

a08 tty2 2023-10-09 11:46 (tty2)

## Probar el comando df y comentar la salida:

DF (disk filesystem – Sistema de archivos de disco) nos proporciona información detallada del espacio usado en el disco. Si deseamos que dentro de la información desplegada este el tipo de sistema de archivos usaremos el parámetro –T y para que sea legible añadiremos h, quedando la sintáis de la siguiente manera:

# df

#df -Th

a08@a08-HP-ProDesk-600-G5-SFF:~$ df

S.ficheros bloques de 1K Usados Disponibles Uso% Montado en

tmpfs 1613936 2368 1611568 1% /run

/dev/sda2 459850824 110566192 325851980 26% /

tmpfs 8069664 672 8068992 1% /dev/shm

tmpfs 5120 4 5116 1% /run/lock

tmpfs 8069664 0 8069664 0% /run/qemu

/dev/sda1 523244 6216 517028 2% /boot/efi

tmpfs 1613932 2472 1611460 1% /run/user/1000

/dev/sdc1 30249020 127848 30121172 1% /media/a08/KIOXIA

a08@a08-HP-ProDesk-600-G5-SFF:~$ df -Th

S.ficheros Tipo Tamaño Usados Disp Uso% Montado en

tmpfs tmpfs 1,6G 2,4M 1,6G 1% /run

/dev/sda2 ext4 439G 106G 311G 26% /

tmpfs tmpfs 7,7G 672K 7,7G 1% /dev/shm

tmpfs tmpfs 5,0M 4,0K 5,0M 1% /run/lock

tmpfs tmpfs 7,7G 0 7,7G 0% /run/qemu

/dev/sda1 vfat 511M 6,1M 505M 2% /boot/efi

tmpfs tmpfs 1,6G 2,5M 1,6G 1% /run/user/1000

/dev/sdc1 ntfs3 29G 125M 29G 1% /media/a08/KIOXIA

## Listar información sistema de archivos Linux

LSBLK (list block devices) nos permite desplegar en pantalla toda la información del bloque de dispositivos que se encuentren disponibles. Agregamos el parámetro –f para listar el tipo de sistema de archivos ejecutando el siguiente comando. De esta forma conoceremos en detalle los diversos sistemas de archivos.

#lsblk -f

a08@a08-HP-ProDesk-600-G5-SFF:~$ lsblk -f

NAME FSTYPE FSVER LABEL UUID FSAVAIL FSUSE% MOUNTPOINTS

loop0 squashfs 4.0 0 100% /snap/android-studio/128

loop1 squashfs 4.0 0 100% /snap/bare/5

loop2 squashfs 4.0 0 100% /snap/android-studio/127

loop3 squashfs 4.0 0 100% /snap/brave/285

loop4 squashfs 4.0 0 100% /snap/brave/291

loop5 squashfs 4.0 0 100% /snap/code/141

loop6 squashfs 4.0 0 100% /snap/code/142

loop7 squashfs 4.0 0 100% /snap/core/16091

loop8 squashfs 4.0 0 100% /snap/core/16202

loop9 squashfs 4.0 0 100% /snap/core18/2790

loop10 squashfs 4.0 0 100% /snap/core20/1974

loop11 squashfs 4.0 0 100% /snap/core20/2015

loop12 squashfs 4.0 0 100% /snap/core22/858

loop13 squashfs 4.0 0 100% /snap/core22/864

loop14 squashfs 4.0 0 100% /snap/cups/980

loop15 squashfs 4.0 0 100% /snap/dbeaver-ce/260

loop16 squashfs 4.0 0 100% /snap/dbeaver-ce/262

loop17 squashfs 4.0 0 100% /snap/drawio/186

loop18 squashfs 4.0 0 100% /snap/drawio/190

loop19 squashfs 4.0 0 100% /snap/eclipse/70

loop20 squashfs 4.0 0 100% /snap/eclipse/71

loop21 squashfs 4.0 0 100% /snap/firefox/3166

loop22 squashfs 4.0 0 100% /snap/firefox/3206

loop23 squashfs 4.0 0 100% /snap/gnome-3-28-1804/198

loop24 squashfs 4.0 0 100% /snap/gnome-3-38-2004/143

loop25 squashfs 4.0 0 100% /snap/gnome-42-2204/132

loop26 squashfs 4.0 0 100% /snap/gnome-42-2204/141

loop27 squashfs 4.0 0 100% /snap/gradle/134

loop28 squashfs 4.0 0 100% /snap/gtk-common-themes/1535

loop29 squashfs 4.0 0 100% /snap/nmap/3152

loop30 squashfs 4.0 0 100% /snap/onlyoffice-desktopeditors/164

loop31 squashfs 4.0 0 100% /snap/pac-vs/1

loop32 squashfs 4.0 0 100% /snap/remmina/6117

loop33 squashfs 4.0 0 100% /snap/snap-store/959

loop34 squashfs 4.0 0 100% /snap/snapd/19457

loop35 squashfs 4.0 0 100% /snap/snapd/20092

loop36 squashfs 4.0 0 100% /snap/snapd-desktop-integration/83

loop37 squashfs 4.0 0 100% /snap/vlc/3078

sda

├─sda1 vfat FAT32 0F02-33A6 504,9M 1% /boot/efi

└─sda2 ext4 1.0 cfcf4d74-d093-4d25-a419-3e4ebb05e857 310,8G 24% /var/snap/firefox/common/host-hunspell

/

sdb

└─sdb1 ext4 1.0 ec3f58be-a7b2-4c82-91f4-0194ef5ecb5e

sdc

└─sdc1 ntfs KIOXIA C07C9AA17C9A9230 28,7G 0% /media/a08/KIOXIA

sr0

## Comprobar el espacio en disco en Linux usando el comando

# du -h

a08@a08-HP-ProDesk-600-G5-SFF:~$ du -h

4,0K ./.gnupg/private-keys-v1.d

28K ./.gnupg

4,0K ./.mozilla/extensions

8,0K ./.mozilla

104K ./.dotnet/corefx/cryptography/crls

[…]

[…]

[…]

44K ./Escritorio/Nueva carpeta

646M ./Escritorio

80G .

# Capitulo 10

Etc.