

Santo-Grial-SO-Parte1.pdf



MacarronCrudo



Sistemas Operativos



2º Grado en Ingeniería Informática



**Escuela Politécnica Superior (Jaén)
Universidad de Jaén**



[Accede al documento original](#)



Escuela de
Organización
Industrial

Contigo que evoluciones.
Contigo que lideras. Contigo que transformas.

**Esto es EOI.
Mismo propósito,
nueva energía.**



Descubre más aquí



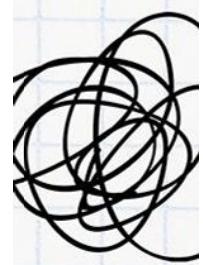
EOI Escuela de
Organización
Industrial

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

pierdo
espacio



Necesito
concentración

ali ali oooh
esto con 1 coin me
lo quito yo...

wuolah

Scribble Gal Sistemas Operativos



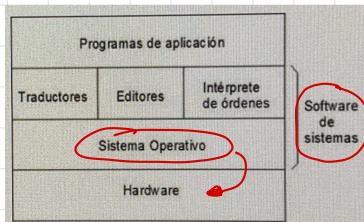
wuolah

Tema 1. Introducción

1-Conceptos Generales

•Definición de Sistema Operativo

Un sistema operativo es un conjunto de programas que manejan directamente los recursos hardware de un ordenador. Pertenece al software de sistemas y es el único que trabaja directamente con el hardware.



Como vemos en la imagen de la izquierda, el Sistema Operativo pertenece al Software de Sistemas junto a los Traductores, Editores y Interpretadores de órdenes. Además, al estar junto al Hardware del ordenador, es el único que trabaja directamente

•Funciones de un S.O.:

El sistema operativo tiene 2 objetivos:

- Sistema que facilita la compresión y programación del sistema subyacente (el que está por debajo):

- No debe de entenderse a nivel hardware como manejar y gestionar la E/S.
 - La memoria virtual de un SO difiere de la memoria de la máquina real.
 - Los ficheros son tratados como nombres y no como direcciones de memoria.
 - Hay un sistema de protección y tratamiento de errores.
 - Ofrece más interacción entre programas.
- El SO actúa como un administrador de recursos:
- Gestiona correctamente todos los recursos disponibles y maximiza el rendimiento.
 - Dirige al procesador cuando hace uso de las otras recursos o cuando se ejecutan otros programas.

Para todo esto el SO debe contabilizar, organizar y gestionar todos los recursos. Además debe elegir como conceder los recursos a los distintos programas que al solicitarlo entran en conflicto.

•SO de propósito general

Hoy en día existen 3 tipos de SO:

- Sistema Monoprogramado

Hay un único flujo de trabajo, es decir, la máquina se encarga de una única tarea/proceso.

- Sistema Multiprogramado

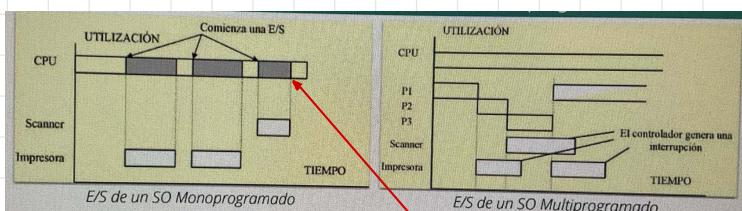
El uso de la CPU es incrementado, mantiene los programas en memoria principal y aprovecha los tiempos de E/S. Estos sistemas necesitan guardar los trabajos pendientes según una gestión de memoria y poder elegir un trabajo con un algoritmo de planificación.

-Sistema de tiempo compartido

El tiempo de la CPU es repartido entre los usuarios, esto permite sistemas más interactivos y sistemas multiusuario.

Genera dos tipos de sistemas: Sistemas distribuidos, sistemas de Tiempo Real.

Podemos observar las siguientes diferencias al realizar una comparación entre el Monoprogramado vs Multiprogramado:



En los sistemas Monoprogramados tenemos retrasos grandes, indicando que la CPU no se está utilizando, mientras que en el Multiprogramado la CPU siempre está ocupada por un proceso, por lo que siempre hay un programa en la CPU, siendo el procesador aprovechado al máximo posible.

Ahora vamos a ver las comparaciones al hacer Multiprogramado vs Tiempo Compartido:

- **Objetivos principales:** Del multiprogramado es maximizar el uso del procesador, mientras que de los SO de tiempo compartido es minimizar el tiempo de respuesta.
- En sistema multiprogramado no ofrece interacción con el usuario, solamente utiliza eficientemente los recursos del sistema.
- Los SO de tiempo Compartido son multiprogramados pero NO al revés, esto es porque los de Tiempo Compartido son una extensión del multiprogramado.
- Ambos utilizan una cola de trabajos

• Servicios ofrecidos de un SO

Un SO ofrece servicios a programas y usuarios:

- Interfaz de usuario
- Ejecución de programas
- Operaciones de E/S
- Manipulación del sistema de ficheros
- Comunicación entre procesos
- Detección de errores
- Asignación de recursos
- Registro/monitoreo y protección y seguridad

• Interfaz entre SO y Usuario

Una interfaz es un elemento intermedio que permite la comunicación entre diferentes niveles, un SO es una interfaz entre un usuario y la máquina.

- Mediante el intérprete de órdenes

Un intérprete de órdenes es un programa donde el usuario introduce por teclado órdenes para realizar tareas, una terminal básicamente. Muchas de estas órdenes hacen llamadas al sistema de forma indirecta.

Tipos

- Mediante un entorno de Escritorio

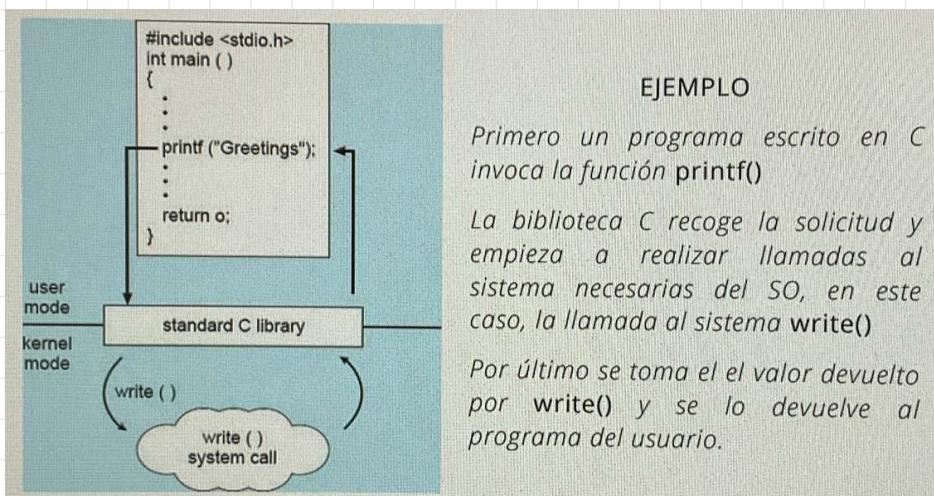
Muchos SO se han adaptado a una interfaz gráfica de usuario o GUI. Un escritorio es una GUI ya que emplea ventanas y menus para el uso del ratón principalmente. Al realizar clics con el ratón estamos haciendo llamadas al sistema.

- Mediante una pantalla táctil

En los teléfonos inteligentes nuestra interfaz gráfica que utilizamos es la pantalla táctil del propio dispositivo móvil.

• Llamadas al sistema

Las llamadas al sistema son un mecanismo con el que se puede invocar los distintos servicios que el SO ofrece a programas en modo usuario. Una llamada al sistema es necesaria cuando un proceso que es ejecutado en modo usuario intenta ejecutar una función exclusiva del modo núcleo.



En los sistemas multiprogramados se pueden ejecutar varios procesos de usuario a la vez, dándose una protección de las acciones al resto de programas, mediante un mecanismo hardware: el modo supervisor/kernel y el modo usuario.

Para realizar una llamada al sistema se siguen los siguientes pasos:

- 1- Se colocan unos parámetros en los registros
- 2- Se ejecuta una instrucción "trap" que guarda los registros PC, PSW en un lugar seguro en memoria principal
- 3- Se carga un nuevo parámetro en el registro PC
- 4- Cuando se acaba el servicio, se coloca en un registro si se realizó con éxito o no, después se ejecuta la instrucción "return from trap" para volver a tener los registros PC, PSW.

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

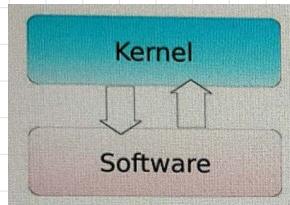
2. Estructura de los SO

Estructuras de los SO.

Tenemos diferentes estructuras:

- Sistema monolítico o de Estructura Simple

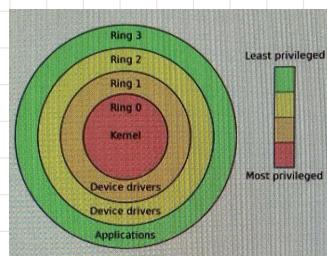
Se trabaja totalmente con el núcleo/kernel en modo supervisor. Estos SO no aplican el principio de ocultación de información y los servicios que ofrece son llamadas al sistema.



Estructura en niveles

La jerarquía del sistema se diferencia por niveles, siendo el nivel 0 el más privilegiado (hardware/kernel) y el nivel N el menor (interfaz de usuario). Cada nivel es implementado únicamente con las operaciones ofrecidas por niveles inferiores.

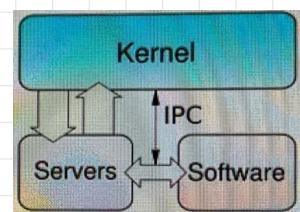
Debe de decidirse en qué nivel va cada funcionalidad y por cada nivel que se creará habrá más carga de trabajo a la llamada del sistema, por lo que tardará más en ejecutarse.



- Modelo cliente-servidor o Microkernel (Micronúcleo)

El objetivo es tratar el código necesario para tener un núcleo con las funcionalidades más básicas. Esto permite que cada parte del SO sea pequeña y manejable, evitando colapsos y fallos generales y adaptándose mejor a los sistemas operativos distribuidos.

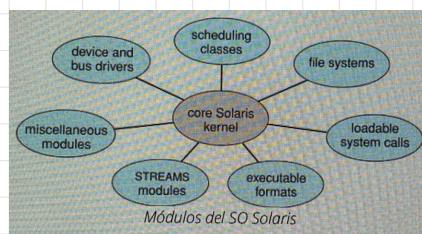
Los procesos de usuario (cliente) envían peticiones al servidor, por lo que el núcleo se encarga de permitir la comunicación entre clientes y servidores.



Módulos

Este tipo posee un kernel dispone de un conjunto de componentes fundamentales, enfatizando dinámicamente los servicios adicionales durante el arranque.

Similar al sistema de estructura por niveles ya que cada sección del núcleo está bien definida y protegida, excepto que un módulo puede invocar a cualquier otro, también es similar al microkernel ya que el módulo principal solo tiene funciones esenciales.



WUOLAH

pierdo espacio



Necesito concentración

ali ali oooh
esto con 1 coin me
lo quito yo...

wuolah

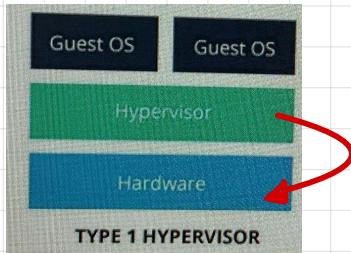
Máquina virtual

Las máquinas virtuales abstractan todos los componentes de un ordenador y mediante un programa se crea una máquina virtual. Esto permite emular un SO, todos aquellos SO que son emulados no permiten compartir información entre ellos.

La creación de máquinas virtuales / virtualización de un SO es gracias a un software especial llamado Hipervisor, esto permite abstractar el hardware físico. Tenemos 2 tipos de Hipervisor:

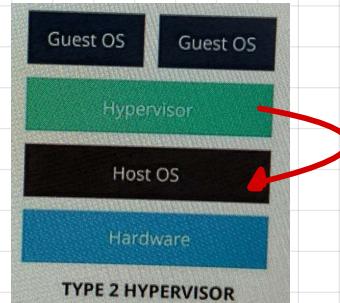
Hipervisor de tipo 1 o básico:

Ejecutado directamente en el Hardware físico, sin necesidad de un SO subyacente.



Hipervisor de tipo 2 o alojado:

Ejecutado en un SO, por lo que crea latencia al ser el intermediario entre hipervisor y hardware.



Clasificación de los SO

Un SO se puede clasificar de diversas formas:

Modo de trabajo de los usuarios

- Online o interactivos
- Offline o batch jobs

Número de usuarios

- Monousuarios
- Multiusuarios

Propósito

- De propósito general
- De propósito específico

Número de procesadores

- Multiprocesador
- Sistemas distribuidos

Generación y arranque de un SO

Un SO es configurado/generado por un ordenador en un proceso conocido como generación del sistema o SYSGEN. Los SO suelen distribuirse en CD-ROM, DVD-ROM y con extensión .iso

El programa SYSGEN lee un archivo o pregunta sobre la información del hardware o prueba directamente el hardware para averiguar los componentes.

Para que el hardware reconozca donde está el Kernel y lo cargue emplea un proceso denominado booting:

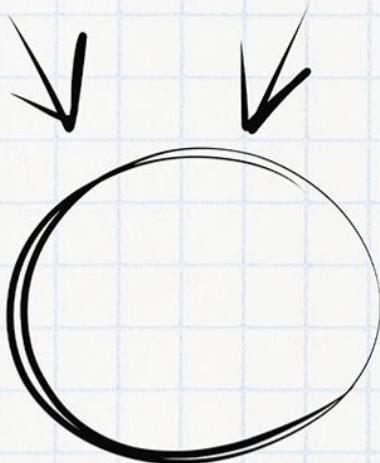
- 1- Un pequeño código (boot loader) localiza el Kernel
- 2- Se carga en memoria principal el Kernel y se ejecuta
- 3- El Kernel inicializa el hardware
- 4- Se monta el sistema de archivos raíz

Imagínate aprobando el examen

Necesitas tiempo y concentración

Planes	PLAN TURBO	PLAN PRO	PLAN PRO+
diamond Descargas sin publi al mes	10 🟡	40 🟡	80 🟡
clock Elimina el video entre descargas	✓	✓	✓
folder Descarga carpetas	✗	✓	✓
download Descarga archivos grandes	✗	✓	✓
circle Visualiza apuntes online sin publi	✗	✓	✓
glasses Elimina toda la publi web	✗	✗	✓
€ Precios	Anual <input type="checkbox"/>	0,99 € / mes	3,99 € / mes
			7,99 € / mes

Ahora que puedes conseguirlo,
¿Qué nota vas a sacar?



WUOLAH

Tema 2. Procesos

1 - Gestión de Procesos

• Concepto de proceso

Un **proceso** es la ejecución de un programa que incluye sus valores actuales en registros y variables. Mientras que **programa** es una secuencia de instrucciones escritas en un lenguaje dado. La existencia de los procesos permite que la multiprogramación sea posible utilizando pseudoparalelismo.

La velocidad de ejecución de un proceso no es siempre la misma, depende de:

- El número de procesos
- La complejidad de los programas
- Si está parcialmente alojado en memoria o no
- Del tiempo de ejecución disponible asignado al procesador

• Imagen de un proceso

El SO reconoce a los procesos ya que los agrupa en una estructura de datos llamado **Bloque de Control de Proceso (BCP)**, que es creado cuando un proceso aparece y eliminado cuando el proceso termina. El BCP contiene la siguiente información:

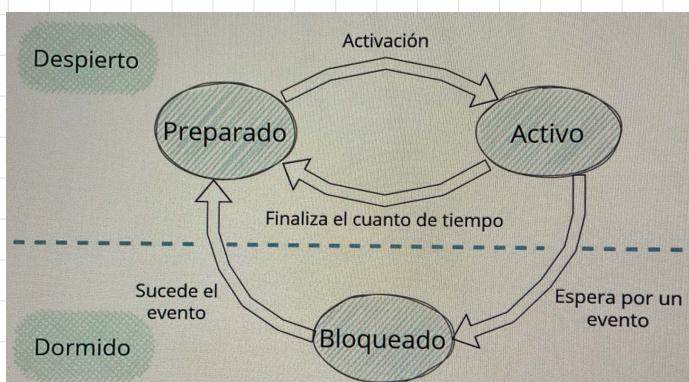
- Identificación del proceso
- Información de estado del procesador
- Información de control del proceso

} Contenido que almacena un BCP

• Estados de un proceso

Un proceso puede encontrarse en diferentes estados:

- Activo / En ejecución: El proceso está siendo ejecutado en la CPU
- Lista / Preparado: El proceso es un candidato para pasar al estado activo
- Bloqueado: El proceso está esperando algún evento para proseguir con la ejecución

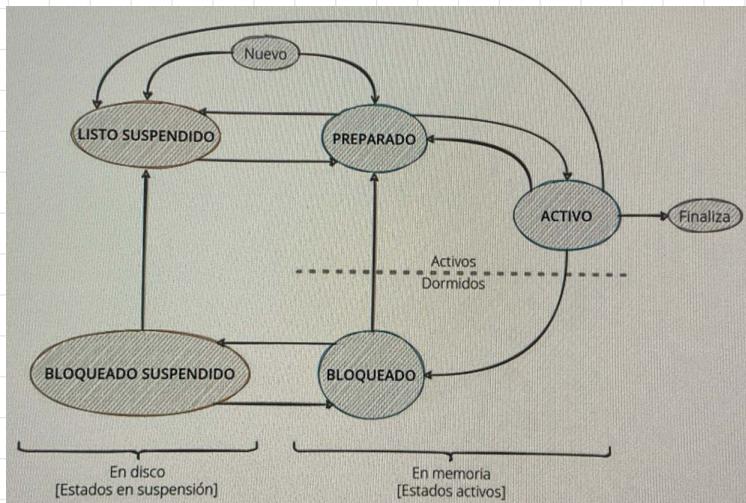


Se pasa de Preparado a Activo gracias al **despachador** que da la CPU al proceso elegido por el **planificador**. → Es el que elige el proceso a ejecutar. Solamente puede haber un único proceso en ejecución, mientras que pueden existir varios procesos preparados y varios procesos bloqueados.

Para organizar y dirigir los estados de todos los procesos se debe decidir que procesos deben ejecutarse y durante cuánto tiempo, esto es llevado a cabo gracias a una **planificación de procesos**.

↳ Decide qué procesos deben ejecutarse y durante cuánto tiempo

Pone a ejecutar un proceso



Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

- Control sobre procesos

- Cambio de proceso

pierdo espacio



Necesito concentración

ali ali ooooh
esto con 1 coin me
lo quito yo...

wuolah

Un proceso se cambia por otro cuando a un proceso se le interrumpe y el SO interviene para ejecutar y dar el control a un proceso diferente.

Hay 3 tipos de eventos los cuales pueden provocar un cambio de proceso:

- Interrupción de reloj.
 - Interrupción de E/S.
 - Fallo de memoria.
 - Excepción o "Trap".
 - Llamada al sistema.
- Tipos de Interrupción

Un cambio de proceso sigue los siguientes pasos:

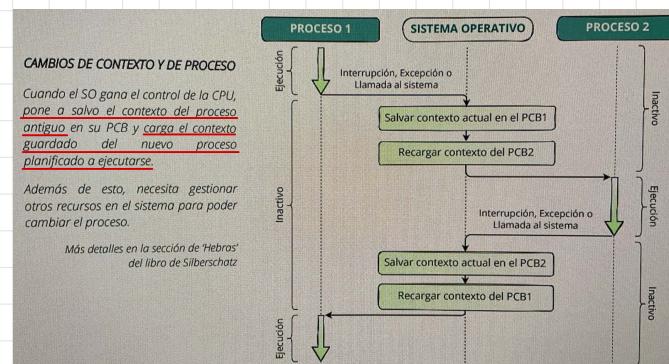
- 1 - Salvar el contexto del procesador
- 2 - Actualizar el PCB del proceso en ejecución
- 3 - Mover el PCB del proceso a la cola apropiada
- 4 - Seleccionar otro proceso
- 5 - Actualizar el PCB del proceso seleccionado
- 6 - Actualizar las estructuras de datos
- 7 - Restaurar el contexto del procesador

Proceso para un cambio de proceso

Por lo que un cambio de proceso implica 2 cambios de contexto

Mientras que un cambio de contexto no obliga a que se cambie de proceso

- Cambio de contexto



Con contexto nos referimos a la información principal que rodea a un proceso y no permite detallar el entorno de dicho proceso.

Realizar los cambios de contexto es una de las tareas del despachador.

- Modo de ejecución

Un programa se puede ejecutar en modo usuario o en modo Kernel, dependiendo del modo se cambiarán las instrucciones que se pueden ejecutar.

↳ Ejecutado por el Hardware

- Operaciones con procesos

Se pueden realizar diversas operaciones con los procesos:

- Despachar un proceso

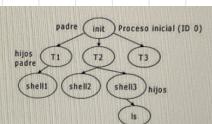
- Permitir que un proceso se comunique con otro

- Crear un proceso. Esto conlleva el seguimiento de los siguientes pasos:

- 1 - Al proceso se le asigna un PID (identificador)
- 2 - Se inserta en la lista de procesos conocidos del sistema
- 3 - Se determina su prioridad inicial del proceso
- 4 - Se crea su BCP
- 5 - Se le asignan recursos iniciales

Representación de procesos en sistema UNIX

Un proceso puede crear un nuevo proceso hijo. En los sistemas UNIX, la llamada a la función fork() (Wikipedia) crea procesos de tal manera que todos forman parte de un mismo árbol de procesos.



- Destruir un proceso. Un proceso finaliza cuando se termina su última instrucción, tras ello el SO lo elimina mediante una llamada al sistema, estableciendo comunicación con el proceso padre y liberando los recursos. También puede acabar por exceder el uso de los recursos que tenía asignados o por un proceso padre que acaba (al acabar el padre acaba el hijo).

- Clavar y desbloquear un proceso. Para ello sigue los siguientes pasos:

- 1 - Se borra de las listas y tablas que aparecen en el sistema
- 2 - Se devuelven los recursos que estaba utilizando
- 3 - Se borra su BCP y se devuelve el espacio

-Suspender un proceso. Para ello se desaloja de memoria principal y se transfiere a una memoria de almacenamiento, la suspensión de un proceso suele ser durante un breve periodo de tiempo, en caso contrario el SO puede liberar los recursos del proceso bloqueado.

Razones para la suspensión de un proceso	
Swapping	El SO necesita liberar suficiente memoria principal para traer un proceso en estado Preparado
Otras razones del SO	El SO puede suspender un proceso en segundo plano, útil o sospechoso que pueda causar problemas
Solicitud del usuario	Un usuario puede suspender la ejecución de un programa para poder depurarlo o porque está utilizando un recurso
Temporización	Un proceso puede ejecutarse periódicamente (por ejemplo, para estadísticas del sistema) y se suspende hasta el siguiente turno
Solicitud del proceso padre	Un proceso padre puede querer suspender la ejecución de su hijo para examinarlo, modificarlo o para coordinar la actividad de varios procesos hijo

-Reanudar un proceso. Consiste en activar un proceso que estaba suspendido

2 - Planificación de Procesos

Con el Scheduler conocemos a qué proceso y cuando le debemos asignar los recursos

La clave en SO multiprogramados o derivados es la planificación o scheduling, con lo que sabemos cuando y a qué proceso se le asignan recursos. Para que el Scheduler tome estas decisiones emplea un Algoritmo de planificación que permite la suspensión y reactivación de procesos. Además, existe más planificadores, como lo puede ser para la CPU.

•Objetivos y criterios

Para la correcta evaluación de estrategias debemos de fijarnos en los siguientes criterios:

- Criterios orientados a usuarios
- Criterios orientados al sistema
- Criterios relativos al rendimiento

Generalmente, debe ser equitativa, eficiente, debe maximizar la productividad, lograr buenos tiempos de respuesta y conseguir un tiempo de proceso global predecible.

•Niveles de planificación



! CONOCER MUY BIEN, TÍPICA DE TIPO TEST

Tenemos 3 niveles/tipos:

- A corto plazo (CPU scheduler)

A la CPU se le asigna un proceso mediante un algoritmo concreto. El dispatcher deberá transferirlo a la CPU.

Normalmente suele ocurrir un cambio de contexto cuando el SO necesita interactuar con el scheduler, esto puede darse por ejemplo al acabar un cuento de tiempo en un Round Robin (RR).

- A medio plazo

Encargada de elegir qué procesos pasan a suspendidos, es decir, moviliza los procesos de memoria principal a memoria secundaria y viceversa. Si se da el caso y en la memoria principal los procesos estén bloqueados, entonces, para no ocupar la CPU, el planificador a medio plazo los move a memoria secundaria para así aprovechar más la memoria principal.

- A largo plazo

Su tarea es ir mezclando algunos trabajos con los del planificador de CPU (corto plazo). Estos trabajos suelen utilizar mucho el procesador o periféricos de E/S. Suele activar cuando la utilización de CPU es algo baja.

•Reloj de interrupciones



! TAMBICN MUY TÍPICA

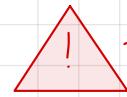
Reloj Hardware != Reloj de Interrupciones

Es clave para que ningún proceso accapare la utilización del procesador. IMPORTANTE, no debe confundirse con el reloj de la máquina/reloj hardware, ya que el reloj de interrupciones es software.

Gracias al reloj de interrupciones se garantizan tiempos de respuesta aceptables para los usuarios interactivos, evitando que el SO se bloquee entrando en un ciclo infinito.

El reloj de interrupciones debe tener una frecuencia menor que el reloj hardware.

Criterios sobre planificación



TÍPICA DE TIPO TEST

- Apropiación

Una planificación es apropiativa si un proceso puede quitarle la CPU al proceso activo.

Apropiativa	No Apropiativa
Útil en sistemas de <u>tiempo real</u> y sistemas de <u>tiempo compartido</u> [enlace]	Se suele utilizar en programas de <u>Sistemas por lotes</u> [Wikipedia]
Garantiza tiempos de respuesta aceptables	Tiempos de respuesta predecibles.
Implica un gasto extra al cambiar de proceso y contextos	Los procesos que toman más tiempo retrasan a los cortos
Es necesaria una asignación de prioridades coherente	Trato más justo ya que no interfiere tanto con la CPU

- Prioridad de un proceso

Un proceso que está listo puede apropiarse de la CPU si es más prioritario que el proceso activo, en este caso, el planificador selecciona el proceso con mayor prioridad para que sea ejecutado proximamente.

Distintas formas para asignar prioridades	
Asignación automática	Asignación según las acciones externas
Ganarse	Comprarse
Estáticas [Fáciles de implementar, bajo gasto extra, no responden a cambios en el entorno del sistema]	Dinámicas [Más complejos, implican más gasto extra, más sensibles con los cambios en el sistema]
Racionalmente	Arbitrariamente

- Retraso

Es el periodo de tiempo de un proceso en el que necesita usar un recurso del sistema, es decir, el número de trabajos por unidad de tiempo.

- Inanción

Situación en la cual un proceso está listo para ejecutarse pero se le niega el acceso al procesador continuamente ya que tienen preferencia el resto de procesos.

Una planificación debe ser equilibrada, eficiente, con un tiempo de procesamiento global predecible y con una alta productividad.

• Algoritmos de planificación

Debemos tener en cuenta las siguientes medidas:

- Tiempo de llegada (T_L): Momento en el que llega un proceso, es proporcionado
- Tiempo de Servicio (T_S): Tiempo que necesita un proceso para prestar todos sus servicios
- Tiempo de Finalización (T_F): Tiempo en el que un proceso ha terminado de ofrecer sus servicios al sistema
- Tiempo de Espera (T_E): Tiempo que un proceso ha permanecido esperando desde su llegada hasta que ha comenzado a prestar sus servicios
- Porcentaje de uso (% recurso): Mide el porcentaje de uso de un recurso mediante la fórmula $100 \cdot \frac{\text{Tiempo Usado}}{\text{Tiempo Total}}$

TÍPICO PARA PROBLEMA

Los algoritmos son los siguientes:

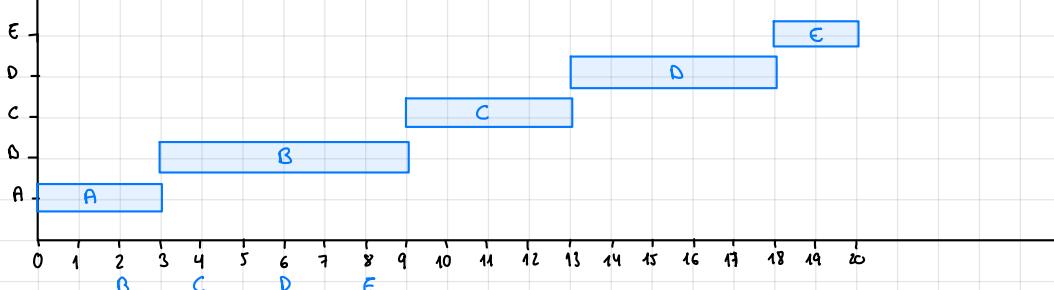
- Primero en entrar - primero en salir, FIFO o FCFS

Los procesos se despiden en orden de llegada.

No es apropiativa, los trabajos largos pueden hacer esperar a los cortos o los trabajos sin importancia pueden hacer esperar a los importantes. No suele garantizar buenos tiempos de respuesta.

Proceso	A	B	C	D	E
Tiempo de llegada	0	2	4	6	8
Tiempo de servicio	3	6	4	5	2

	A	B	C	D	E	Media
Tiempo de Finalización (T_F)	3	9	13	18	20	63/5 = 12.6
Tiempo de Espera (T_E)	0	3-2=1	5	7	10	23/5 = 4.6

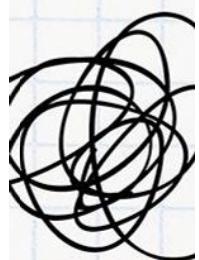


Importante

Puedo eliminar la publi de este documento con 1 coin

→ Plan Turbo: barato
→ ¿Cómo consigo coins? → Planes pro: más coins

pierdo espacio



Necesito concentración

ali ali ooooh
esto con 1 coin me
lo quito yo...

wuolah

- Turno rotatorio o Round Robin (RR)

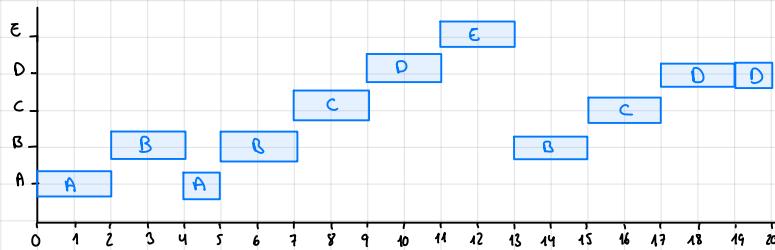
Es un FIFO pero que tiene un quantum (q), esto es un intervalo de tiempo de ejecución que al agotarse puede provocar un cambio de proceso.

Es un algoritmo apropiado.

Proceso	A	B	C	D	E
Tiempo de llegada	0	2	4	6	8
Tiempo de servicio	3	6	4	5	2

Con $q=2$

	A	B	C	D	E	Média
Tiempo de Finalización (T_f)	5	15	17	20	13	14
Tiempo de Espera (T_e)	0	0	3	3	3	1'8



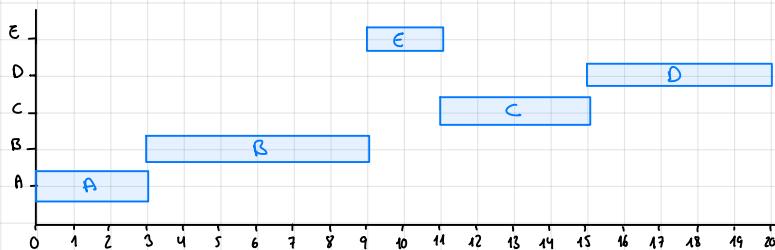
- Prioridad al más corto o Short Job First (SJF o SPN)

Se deben de conocer los tiempos de ejecución previamente, se ejecuta aquel proceso que necesite el menor tiempo.

Puede causar inanición, es no apropiativa.

Proceso	A	B	C	D	E
Tiempo de llegada	0	2	4	6	8
Tiempo de servicio	3	6	4	5	2

	A	B	C	D	E	Média
Tiempo de Finalización (T_f)	3	9	15	20	11	11'6
Tiempo de Espera (T_e)	0	1	7	9	1	3'6

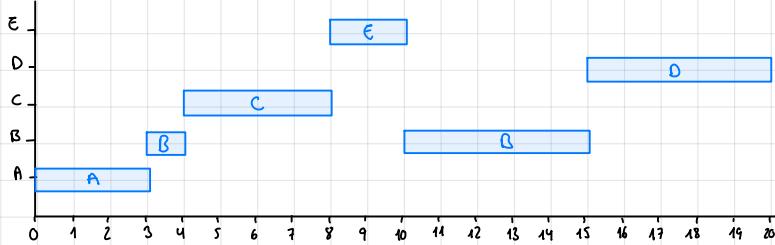


- Prioridad al Tiempo Restante más corto o Short Remaining Time First (SRTF o SRT)

Es una versión apropiativa del SJF, el proceso que está siendo ejecutado puede ser cambiado por otro con menor tiempo de ejecución.

Proceso	A	B	C	D	E
Tiempo de llegada	0	2	4	6	8
Tiempo de servicio	3	6	4	5	2

	A	B	C	D	E	Média
Tiempo de Finalización (T_f)	3	15	8	20	10	11'2
Tiempo de Espera (T_e)	0	7	0	9	0	3'2



- Planificación a la Tasa de Respuesta más alta o **Highest Response Ratio Next** (HRRN o HRN)

Esta planificación da prioridad dependiendo del tiempo de servicio y el tiempo de espera, la prioridad viene de la siguiente fórmula:

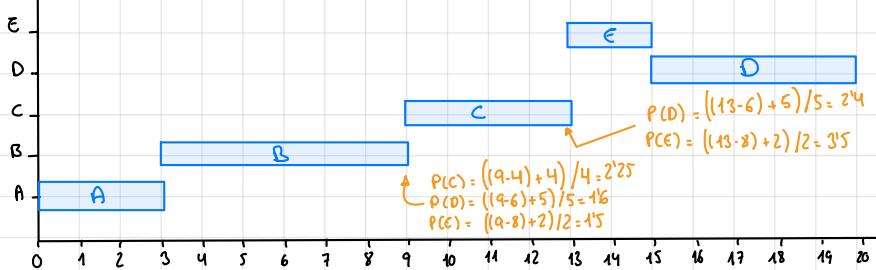
Es un algoritmo no apropiativo donde la prioridad es dinámica.

$$\text{Prioridad} = \frac{T_E + T_S}{T_S}$$

Proceso	A	B	C	D	E
Tiempo de llegada	0	2	4	6	8
Tiempo de servicio	3	6	4	5	2

CONOCER
FÓRMULA
PARA DAR
PRIORIDAD

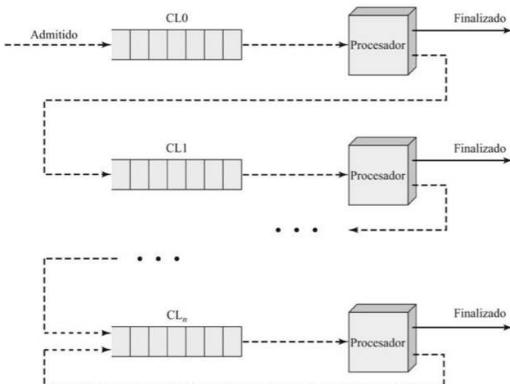
	A	B	C	D	E	Media
Tiempo de Finalización (T_F)	3	9	13	20	15	12
Tiempo de Espera (T_E)	0	1	5	9	5	4



- Colas multivelvel de retroalimentación o Feedback

Este algoritmo:

- Separa los procesos en categorías basadas según su necesidad de la CPU
- Posee mecanismos adaptativos para responder a algunos cambios del comportamiento en el sistema.
- Tiene un mayor gasto extra.



En esta planificación, a los procesos se les asigna una prioridad inicial. Estos cambian sus posiciones en las distintas colas de preparados según dichas prioridades para llegar finalmente a ejecutarse.

Cada cola de listos puede tener una planificación propia. Normalmente se utiliza RR [ROUND ROBIN], donde cada cola de listos [CL0, CL1, ..., CLn] tiene un quantum de tiempo único [para 0 q=2, para 1 q=4, etc...].

- Rendimiento o porcentaje de uso

! SE JUELE PEPIN
TAMBIÉN

Dado el siguiente problema:

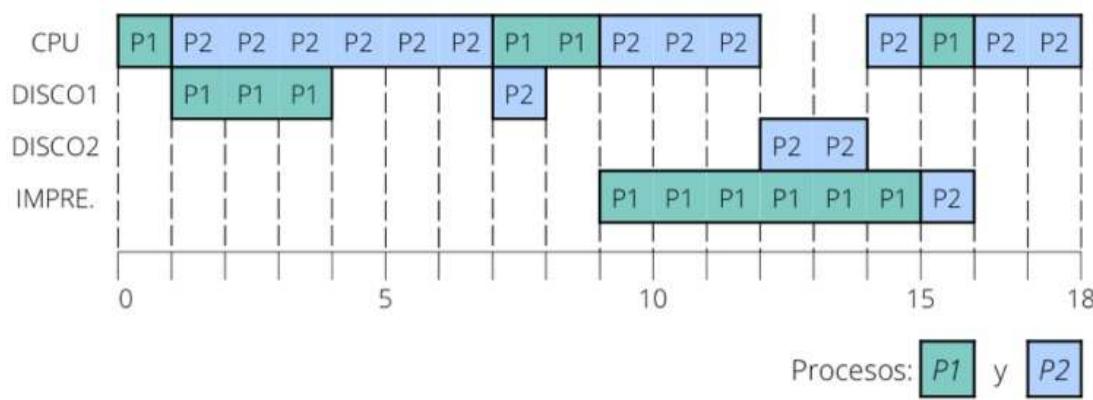
Un sistema tiene los siguientes recursos: una CPU, dos discos y una impresora. En el sistema se van a ejecutar dos procesos con las siguientes características:

- P1: 13 unidades de tiempo en el orden siguiente: 1 para CPU, 3 para Disco1, 2 para CPU, 6 para impresora y 1 para CPU.
- P2: 16 unidades de tiempo en el orden siguiente: 6 para CPU, 1 para Disco1, 3 para CPU, 2 para Disco2, 1 para CPU, 1 para Impresora y 2 para CPU.

Calcule el tiempo de finalización y tiempo de espera medio de los procesos y la utilización de CPU y de cada uno de los dispositivos durante el tiempo que dura la ejecución de ambos procesos usando el algoritmo de planificación FIFO.

Nota: P1 y P2 se crean a la vez, pero en t=0 la CPU es para P1.

Generemos el siguiente gráfico:



Ejecución en la CPU	Tiempo de llegada, T_L	Tiempo de finalización, T_F	Tiempo de espera, T_E	T_F Medio	T_E Medio
P1	0 ms	16 ms	16-0-4=12	$\frac{16 \text{ ms} + 18 \text{ ms}}{2} = 17 \text{ ms}$	$\frac{12+6}{2} = 9 \text{ ms}$
P2	0 ms	18 ms	18-0-12=6		

Donde podemos calcular el rendimiento / porcentaje de uso:

$$\%_{CPU} = 100 \cdot \frac{16 \text{ ms}}{18 \text{ ms}} = 88'88\%$$

$$\%_{DISCO1} = 100 \cdot \frac{4 \text{ ms}}{18 \text{ ms}} = 22'22\%$$

$$\%_{DISCO2} = 100 \cdot \frac{2 \text{ ms}}{18 \text{ ms}} = 11'11\%$$

$$\%_{IMPRESA} = 100 \cdot \frac{7 \text{ ms}}{18 \text{ ms}} = 38'88\%$$

Importante

Puedo eliminar la publi de este documento con 1 coin

→ Plan Turbo: barato
→ Planes pro: más coins

3 - Hilos

pierdo espacio



Necesito concentración

ali ali ooooh
esto con 1 coin me
lo quito yo...

wuolah

En el modelo de proceso clásico, un proceso es un programa de ejecución con un único hilo de control. Sobre dicha ejecución, esto tiene dos grandes limitaciones:

- Muchas aplicaciones utilizan los recursos del sistema en gran medida y de manera independiente.
- Los procesos tradicionales no pueden aprovechar las arquitecturas con más de una CPU, donde se logra compartir recursos y sincronizar sus tareas.

Utilizar múltiples procesos conlleva una serie de desventajas:

- Cada proceso debe tener su propio espacio de direcciones, utilizando mecanismos de comunicación entre procesos.
- Hay un gasto extra para despachar los procesos sobre diferentes máquinas o procesadores.

Por ello surge el concepto de hilo y los sistemas operativos multihilo.

Definición de hilo

Es una unidad que posee recursos: A un proceso se le asigna un espacio de memoria y en ocasiones se le asigna más para acceder a otros tipos de recursos en el sistema.

Es una unidad que se despacha: Un proceso es un flujo de ejecución a través de uno o más programas, por lo que dicha ejecución se intercambia con las de otros procesos.

-Proceso: Unidad que posee recursos, por lo que dentro de un proceso se pueden tener varios hilos de control.

-Hilo: Es una instancia de control dentro de un proceso que ejecuta sus instrucciones de forma totalmente independiente, es decir, es la unidad básica de asignación de la CPU.

Cuenta con:	Cosas que comparte con los hilos pertenecientes al mismo proceso:
<input type="radio"/> Identificador de hilo	<input type="radio"/> Sección de código
<input type="radio"/> Estado de la ejecución	<input type="radio"/> Sección de datos
<input type="radio"/> Contexto de la ejecución	<input type="radio"/> Otros recursos del SO [archivos abiertos y señales]
<input type="radio"/> Conjunto de registros	
<input type="radio"/> Pila de ejecución	

Diferencias entre Hilos y Procesos:

Diferencias entre Hilos y Procesos	
Procesos	Hilos
Se necesitan recursos del SO para cambiar un proceso.	No se necesitan recursos del SO para un cambio de hilo.
Si se bloquea un proceso, lo hace en su totalidad; no es posible ejecutar una parte del proceso.	Si se bloquea un hilo, otro hilo del mismo proceso aún puede ejecutarse.
Cada proceso usa el mismo código y tiene su propio espacio de memoria.	Todos los hilos pueden compartir ficheros y procesos hijo.
Una aplicación que use múltiples procesos consumirá muchos recursos del sistema.	Los procesos que usan múltiples hilos usan menos recursos del sistema.
Cada proceso trabaja por su cuenta.	Los hilos pueden acceder a datos de otros hilos.

SUELLE CAER EN EL TIPO TEST

Los hilos tienen varias ventajas: se tarda menos en crear un nuevo hilo de un proceso ya existente, al igual que también se tarda menos al finalizar el hilo o hacer un cambio de contexto de hilos en un mismo proceso. Además, se mantiene una única copia en memoria del código.

Concurrencia y Paralelismo

Paralelismo: Grado real de ejecución paralela y está limitado por el número de procesadores.

Concurrencia: Máximo paralelismo que se puede alcanzar sin estar limitado por los procesadores.

Concurrencia	
A nivel de sistema	A nivel de aplicación
El kernel suministra concurrencia de sistema reconociendo múltiples hilos de control dentro de un proceso y planificándolos.	Las bibliotecas de hilos a nivel de usuario no son reconocidas por el kernel.
El kernel multiplexa hilos entre los procesadores disponibles o dentro de un único procesador.	Los hilos son planificados y manejados por la propia aplicación.
Consumo recursos valiosos.	Permite estructurar aplicaciones concurrentes.
	Con llamadas al sistema no bloqueantes, las aplicaciones pueden mantener a la vez varias iteraciones en progreso.

Muchas SO implementan un modelo de concurrencia mixta, combinando entre la concurrencia de sistema y de usuario.

wuolah

• Modelos multihilo

Existen dos categorías de implementación de hilos en sistemas multihilo:

- Hilos a nivel de Usuario (ULT)

Son los que utilizamos para programar, pueden crearse desde lenguajes de programación como Java

Características de hilos ULT	
Aqui toda la gestión de hilos es realizada por la aplicación y el núcleo / kernel no sabe de la existencia de estos hilos.	
Ventajas	Desventajas
Para intercambiar hilos de un mismo proceso no hace falta cambiar al modo núcleo	Las llamadas al sistema son bloqueantes. Cuando un hilo se bloquea, se bloquean todos los hilos del proceso.
Cada aplicación puede tener su propio algoritmo de planificación específico	El núcleo asigna un proceso a una CPU cada vez. No se puede ejecutar en paralelo los hilos de un mismo proceso.
Se pueden ejecutar en cualquier SO sin cambiar su núcleo	

Para superar el problema del bloqueo de hilos se emplea el "jacketing", convirtiendo una llamada bloqueadora en no bloqueadora.

- Hilos a nivel de Núcleo (KLT)

Los hilos a nivel de núcleo son los del propio sistema operativo y pueden dar soporte a los hilos de usuario

Características de hilos KLT	
La gestión de los hilos la realiza el núcleo / kernel.	
Ventajas	Desventajas
Los hilos pueden hacer llamadas al sistema y bloquearse por E/S o recursos, sin bloquear al proceso	La creación, destrucción y sincronización de hilos necesitan de llamadas al sistema, lo que es costoso
Cada hilo puede despacharse para ejecutarse en un procesador diferente	Si los hilos acceden frecuentemente a datos compartidos, la sobrecarga de sincronización puede anular cualquier beneficio de rendimiento
	La mayoría de los sistemas multiprocesadores suministran cerros que pueden adquirirse a nivel de usuario si no están bloqueados por otros hilos
	La espera ocupada es razonable para recursos que están bloqueados cortos períodos de tiempo; en otro caso es necesario bloquear el hilo, lo que aumenta el coste

• Biblioteca de hilos

Una biblioteca de hilos proporciona al programador una API para crear y gestionar hilos. Existen dos formas principales de implementar una biblioteca de hilos:

- En el Espacio de Usuario: El código y estructuras de datos están en el espacio de usuario, por lo que invocar una función es como hacer una llamada a esa función
- En el Espacio de Kernel: El código y estructuras de datos están en el espacio de Kernel, por lo que invocar una función en la API es hacer una llamada al sistema.

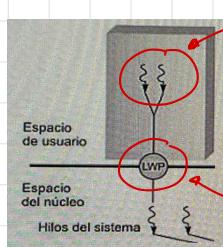
• Planificación de hilos

Los sistemas como Solaris se planifican a 2 niveles:

- Un nivel que asigna los hilos LWT a Procesos Ligeros
- Otro nivel que asigna los Procesos Ligeros a hilos KLT

Tenemos los siguientes tipos:

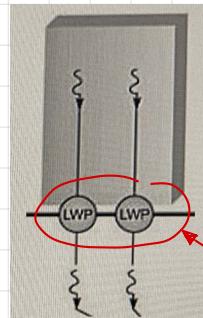
- Muchos hilos en un procesador lógico



- Todos los hilos de usuario compiten por ejecutarse en el único procesador lógico del proceso
- Una llamada bloqueante blocca el proceso
- La creación de hilos y la planificación no usa recursos Kernel, por lo que lo hace más rápido

Procesador lógico del proceso

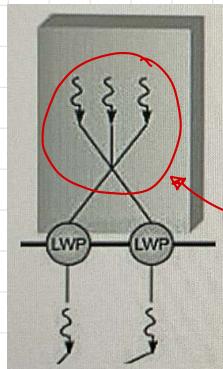
- Un hilo por procesador lógico



- Cada hilo de usuario tiene un procesador lógico
- Los hilos se ejecutan en diferentes procesadores físicos
- Hay más carga para el núcleo

Cada hilo es ejecutado en un procesador diferente

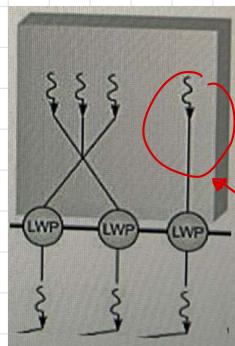
- Muchos hilos en muchos procesadores lógicos (estricto)



- Los hilos se multiplexan en un número lógico igual o menor
- Hilos en paralelo en varias CPU's
- Las llamadas bloqueantes no bloccan al proceso entero

Los hilos son multiplexados en un número igual o menor

- Muchos hilos en muchos procesadores lógicos (no estricto)



- Los hilos se multiplexan en un número lógico igual o menor
- Un hilo se puede asociar a un procesador lógico

Un hilo puede quedar solo para asociarse a un procesador lógico

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

Tema 3. Memoria

La memoria es una matriz de palabras o bytes donde cada palabra posee una dirección propia.

1-Gestión de Memoria

En los SO monoprogramados la memoria era dividida en 2 partes (una para el SO y otra para el programa actualmente en ejecución), mientras que los sistemas multiprogramados la memoria se subdividía para acomodar múltiples procesos.

Definición y conceptos necesarios

Para un proceso poder ejecutarse debe encontrarse en memoria principal.

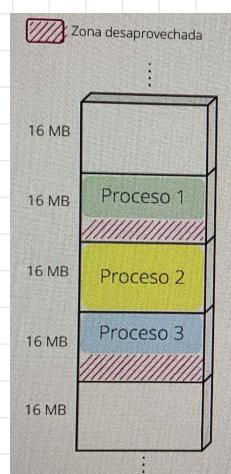
Los procesos están formados por 4 secciones: Código, Variables Globales y constantes, una pila/stack y un Heap.

Las principales funciones de un Gestor de Memoria son:

- Supervisa que los procesos render juntos y sin conflictos
- Contabiliza todas las zonas (libres u ocupadas) por proceso
- Recupera memoria cuando terminan los procesos
- Intercambiar los procesos entre disco y memoria cuando no se está utilizando

Se puede producir 2 tipos de fragmentación:

Fragmentación interna:

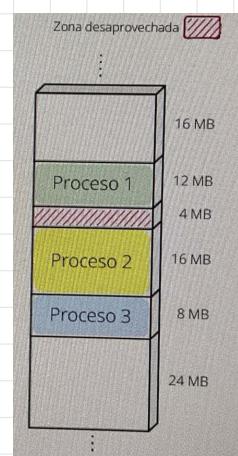


Ocurre al malgastar espacio interno, sucede ya que se introduce un bloque de datos el cual es menor que la partición de memoria.

Se meten bloques de datos que no ocupan todo el espacio de memoria por lo que dejan un pequeño hueco

TÍPICA DE TIPO TEST

Fragmentación externa:



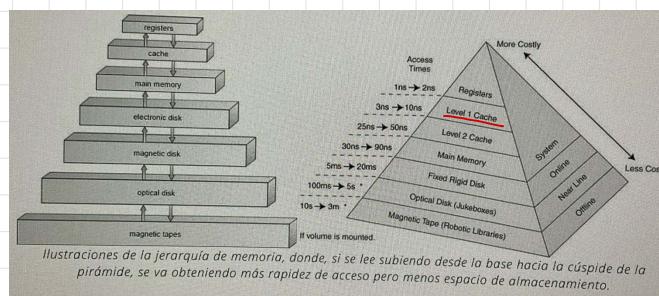
La zona libre de memoria se va fragmentando más conforme pasa el tiempo. Con el tiempo se van produciendo huecos más y más pequeños que quedan moltijados.

La zona de la memoria se va fragmentando cada vez mas, produciendole huecos cada vez más pequeños

Podemos incluir las siguientes definiciones:

- Dirección física: Es la dirección absoluta/real, es una ubicación real de la memoria principal.
- Dirección lógica/virtual: Es una referencia a una ubicación de memoria independiente de la asignación actual de datos en la memoria principal.
- Dirección relativa: Es un ejemplo particular de la dirección lógica, donde la dirección se expresa como una ubicación relativa a algún punto conocido.
- Espacio de direcciones: Es el rango de direcciones de memoria disponibles para un proceso.

Clasificación de la memoria



Como podemos ver en la base están los soportes de memoria secundaria (cintas o disco) que son más baratas que la memoria principal y su capacidad es mucho mayor aunque el acceso a la información es más lento.

En el nivel adicional está la memoria caché, memoria de alta velocidad (más que la memoria principal), esta es muy cara y por ello suelen ser pequeñas.

wuolah

wuolah

Gestión en sistemas monoprogramados

En estos sistemas solo existe un proceso de usuario el cual dispone de todos los recursos. Normalmente, los programas que se suelen escribir en memoria principal lo hacen con direcciones relativas. Por ello, los programas suelen empezar en la dirección relativa cero. Los compiladores y traductores de muchos programas escritos en lenguaje de alto nivel son los encargados de generar estas direcciones relativas.

Generalmente los sistemas monoprogramados y monousuario no poseen protección de memoria y un proceso usuario podría invadir la memoria del sistema.

Gestión en sistemas multiprogramados

La memoria es compartida por varios procesos por lo que se necesita una gestión de la memoria más compleja, por lo que se tiene en cuenta:

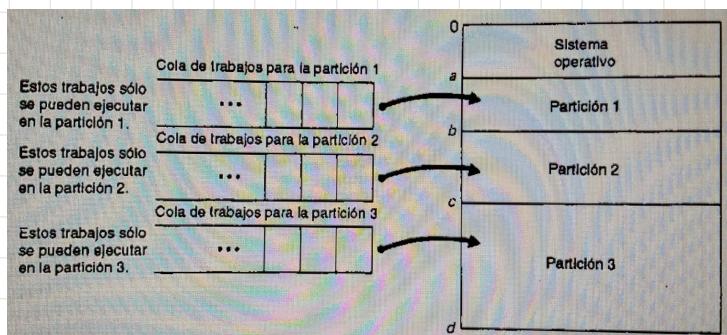
- Recuento de las zonas de memoria ocupadas por los procesos
- Al escribir un programa en memoria principal no se sabe en qué zona se ubicará, es posible que cambie varias veces de lugar.
- Se tiene que proteger las zonas de memoria ocupadas de los procesos

Alloctación de Memoria Contigua

Se tienen dos métodos:

- Particiones estáticas

Consiste en dividir la memoria en varias zonas, da igual que sea de diferentes tamaño o no ya que puede ser modificada.

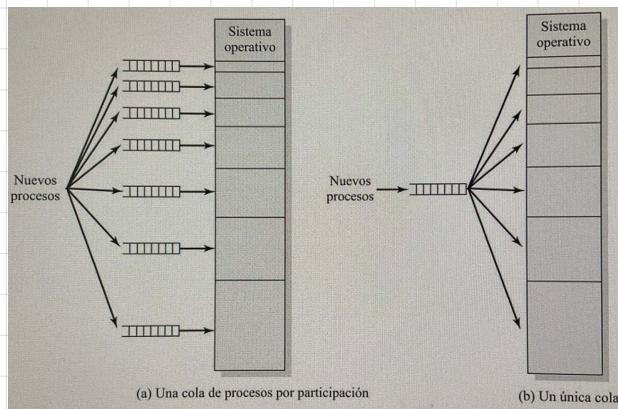


Es poco eficiente ya que un trabajo puede tener que esperar porque la partición ya está ocupada, en este caso se tendría que hacer una reasignación, para ello hay 2 formas:

• Hardware: Registro base con la dirección física del comienzo del programa, es decir: $\text{Dir Física} = \text{Dir Lógica (Programa)} + \text{Reg Base}$

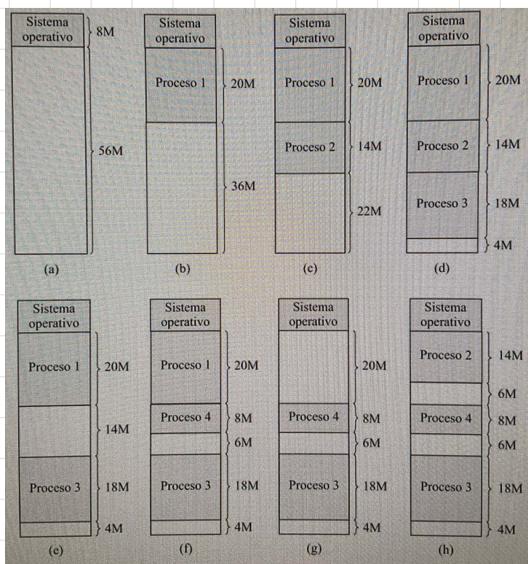
• Software: Se modifican las instrucciones cuando el programa se carga en memoria, para ello el Enlazador (programa que toma uno o varios archivos creados por un compilador o ensamblador y los combina en un único archivo ejecutable) incluye en el programa una lista que indica qué partes son reasignables.

Esto causa fragmentación interna.



- Particiones dinámicas

La memoria se va asignando dinámicamente a los procesos, a cada proceso se le asigna exactamente la memoria que necesita.



En este caso no existe la fragmentación interna pero si la fragmentación externa, ya que aparecen huecos demasiado pequeños, pero si estos huecos se unen y se compacta la memoria esto se podría solucionar.

Para colocar los procesos se puede elegir una de las Estrategias de Colocación:

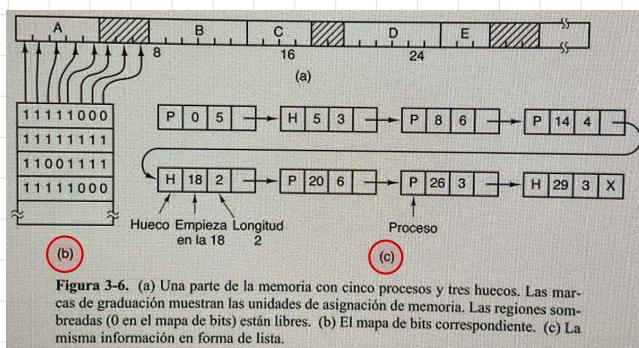
- **Primer en ajustarse:** Meten el proceso en el primer hueco que encuentra.
- **El mejor en ajustarse:** Escoge el hueco que más se acerca al tamaño del bloque.
- **El peor en ajustarse:** Escoge el hueco más grande disponible.

- Ocupación de la memoria

Para asignar la cantidad de memoria:

- Si no va a cambiar el tamaño se le puede dar al proceso la cantidad justa necesitada.
- Si va a cambiar de tamaño durante la ejecución se le puede dar más memoria de la que necesita.

En las particiones estáticas se guarda solo qué partición está libre u ocupada y su dirección de comienzo y fin. Por otro lado, en las particiones dinámicas el tamaño varía por lo que se utilizan Mapas de Bits y Listas Encadenadas.



(b) En el mapa de Bits el 1 indica que esta ocupado la memoria por un proceso y un 0 (zona sombreada) indica que esta libre (hueco).

(c) La P indica que hay un proceso y la H que hay un hueco, el primer número indica donde comienza y el segundo su longitud.

- Intercambio (swapping)

En un sistema de particiones estáticas el número de procesos en estado listo viene determinado por la cantidad de particiones, mientras que en un sistema de particiones dinámicas por el tamaño de la memoria principal y de los procesos.

No es viable tener procesos bloqueados, por ello los almacenan en la memoria secundaria (es más lento pero con mayor capacidad).

- Asignación de Memoria no Contigua

Un proceso puede dividirse en bloques y situarse de forma no consecutiva en la memoria principal. Si los bloques tienen el mismo tamaño son llamadas páginas y se van a organizar mediante paginación. Si los bloques tienen diferentes tamaño se llaman segmentos y se organizan mediante segmentación. Ambas técnicas se pueden combinar consiguiendo como resultado una Segmentación paginada, donde segmentos de diferentes tamaños se componen con páginas de tamaño fijo.

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

pierdo espacio



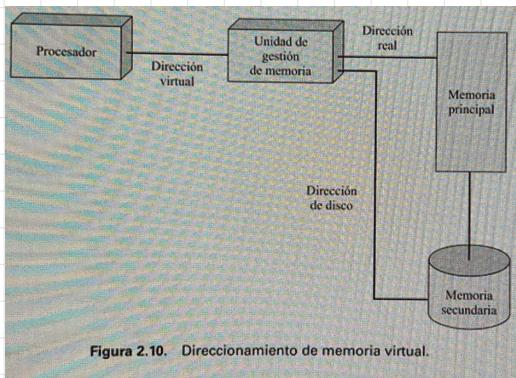
Necesito concentración

ali ali ooooh
esto con 1 coin me
lo quito yo...

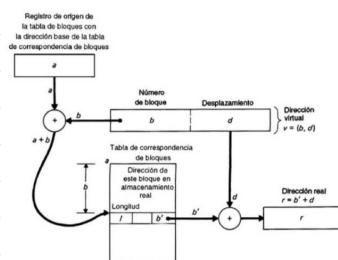
wuolah

Ejemplo general de traducción

Los procesos solamente manejan direcciones virtuales, por lo que las direcciones físicas de memoria principal se deben transformar. Esta conversión de direcciones virtuales a físicas es realizada por una unidad hardware llamada **Unidad de Gestión de Memoria (MMU)**, situado entre procesador y almacenamiento.



Una traducción de direcciones **suele** seguir la siguiente serie de pasos:



- La CPU pide traducir una dirección: $\text{Dir}_{\text{V}} = (b, d)$
- A partir de donde se sitúa la tabla de correspondencias, accedemos a la entrada de la tabla con el número del bloque [b] que nos interese.
- Esta entrada de la tabla tiene varios campos dependiendo de como sea el sistema, pero siempre estará la dirección física [b'] donde **empieza** este bloque de memoria.
- Ahora sabemos que la dirección que buscamos empieza en b' y termina antes de acabar el bloque de memoria. Entonces, sólo queda desplazarnos cierta cantidad [d] en dicho bloque de memoria para buscar el dato que andamos buscando.

La dirección en memoria principal es: $\text{Dir}_{\text{REAL}} = b' + d$

Nota: En paginación b' = tamaño de página · n.º de marco, pues no basta sólo con el n.º de marco

-Paginación-

Tanto memoria principal como los procesos se dividen en **bloques del mismo tamaño**, los bloques son llamados **páginas** y se asignan a un **bloque disponible** en memoria principal conocidos como **marco de página o páginas físicas**

Marco número	Memoria principal
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

(a) Quince marcos disponibles

Marco número	Memoria principal
0	A.0
1	A.1
2	A.2
3	A.3
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

(b) Cargar proceso A

Marco número	Memoria principal
0	A.0
1	A.1
2	A.2
3	A.3
4	B.0
5	B.1
6	B.2
7	
8	
9	
10	
11	
12	
13	
14	

(c) Cargar proceso B

Marco número	Memoria principal
0	A.0
1	A.1
2	A.2
3	A.3
4	B.0
5	B.1
6	B.2
7	C.0
8	C.1
9	C.2
10	C.3
11	
12	
13	
14	

(d) Cargar proceso C

Marco número	Memoria principal
0	A.0
1	A.1
2	A.2
3	A.3
4	B.0
5	B.1
6	B.2
7	C.0
8	C.1
9	C.2
10	C.3
11	
12	
13	
14	

(e) Intercambiar B

Marco número	Memoria principal
0	A.0
1	A.1
2	A.2
3	A.3
4	D.0
5	D.1
6	D.2
7	C.0
8	C.1
9	C.2
10	C.3
11	
12	
13	
14	

(f) Cargar proceso D

Figura 7.9. Asignación de páginas de proceso a marcos libres.

Cuando se ejecute un proceso, sus páginas serán cargadas desde el almacenamiento secundario en marcos libres de la memoria física

0	0
1	1
2	2
3	3

Tabla de páginas del proceso A

0	—
1	—
2	—
3	—

Tabla de páginas del proceso B

0	7
1	8
2	9
3	10

Tabla de páginas del proceso C

0	4
1	5
2	6
3	11
4	12

Tabla de páginas del proceso D

Estructuras de datos para el ejemplo de la Figura 7.9 en el instante (f)

El mecanismo que logra funcionar a la paginación se encarga de dos tareas importantes:

1. La transformación de direcciones virtuales
2. Transferir páginas entre la memoria secundaria y la memoria principal en ambos sentidos.

Recientemente el proceso B está vacío

Indica los marcos que se han quedado libres

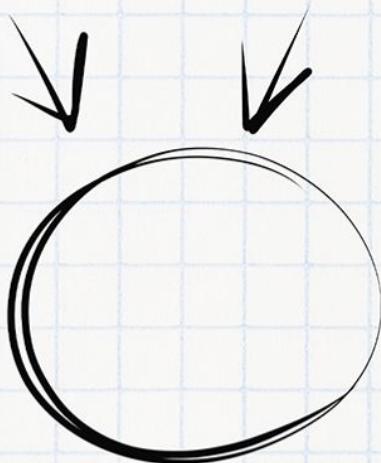
Ocupa las páginas en los cuadros anteriormente estaba el proceso B

Imagínate aprobando el examen

Necesitas tiempo y concentración

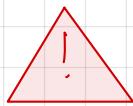
Planes	PLAN TURBO	PLAN PRO	PLAN PRO+
diamond Descargas sin publi al mes	10 🟡	40 🟡	80 🟡
clock Elimina el video entre descargas	✓	✓	✓
folder Descarga carpetas	✗	✓	✓
download Descarga archivos grandes	✗	✓	✓
circle Visualiza apuntes online sin publi	✗	✓	✓
glasses Elimina toda la publi web	✗	✗	✓
€ Precios	Anual <input type="checkbox"/>	0,99 € / mes	3,99 € / mes
			7,99 € / mes

Ahora que puedes conseguirlo,
¿Qué nota vas a sacar?

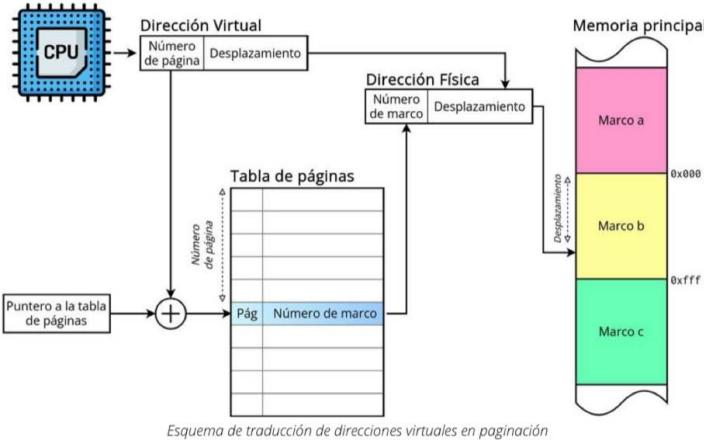


WUOLAH

- Traducción de direcciones en paginación



CONOCER BIEN



1. Cada dirección es dividida en 2: un número de página y un desplazamiento
2. Consultamos el número de página en la tabla de páginas, obtenemos el número de marco
3. Combinamos el número de marco con el desplazamiento para generar la dirección física que va a ser enviada por el bus de direcciones hacia la memoria principal.

Cada dirección virtual contiene:

- 1) Número de la entrada de la página que se va a acceder.
- 2) El desplazamiento o número de palabra que la CPU necesita dentro de dicha página

Destacar como la paginación causa fragmentación interna.

MEDIDA	CÁLCULO
Tamaño de Memoria Virtual, T_{MV}	Se proporciona o bien: $2^{BV} \cdot \text{Palabra} = P_{GS} \cdot T_p$
Tamaño de Memoria Física, T_{MF}	Se proporciona o bien: $M_{CS} \cdot T_p$
Bits de dirección virtual, B_V	Se proporciona o bien: $\log_2(T_{MV}) = B_p + B_D$
Bits para el número de página, B_p	Se proporciona o bien: $\log_2(P_{GS}) = B_V - B_D$
Bits de desplazamiento, B_D	Se proporciona o bien: $\log_2(T_p) = B_V - B_p$
Tamaño de página, T_p	Se proporciona o bien: $2^{BD} \cdot \text{Palabra} = \frac{T_{MV}}{P_{GS}}$
Número de páginas en MV, P_{GS}	Se proporciona o bien: $\frac{T_{MV}}{T_p}$
Número de marcos en MF, M_{CS}	Se proporciona o bien: $\frac{T_{MF}}{T_p}$
Fragmentación interna, F_i	$F_i = \text{Tamaño de página} - \text{Espacio usado}$
Tamaño de tabla de páginas, T_P	Se estudiará en Paginación bajo demanda [ENLACE]

NOTA: Los bits de marcos no tiene por qué coincidir con B_p

Si $T_{MV} = T_{MF}$ entonces $P_{GS} = M_{CS}$

Si $T_{MV} > T_{MF}$ entonces $P_{GS} > M_{CS}$

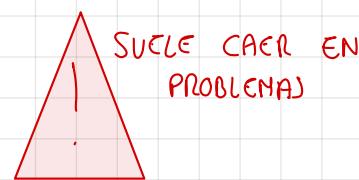
VER EJEMPLO
PAGINA 88

- Soporte del Hardware

Para acelerar la traducción de direcciones se añade una registro asociativo conocido como TLB (cache hardware que contiene las direcciones traducidas más populares), de virtual a física. Primero de todo el hardware verifica si la traducción está en la TLB, si es así, la traducción se realiza rápidamente sin tener que consultar la tabla de páginas. La eficiencia depende de la tasa de aciertos.

EJEMPLO: ¿Cuál sería el tiempo efectivo de acceso a memoria si se dispone de una TLB que acierta el 80% de las veces el número de página deseado en 20 nanosegundos y donde el acceso a la Memoria Principal es de 100 nanosegundos?

Si el número de página está en la TLB $T_{ACIERTO} = 20\text{ns} + 100\text{ns} = 120\text{ns}$ <i>Sumamos directamente</i> $T_{TLB} + \text{Acceso a la página en memoria} = 120\text{ns}$	Si falta en los registros Asociativos $T_{FALLO} = 20\text{ns} + 100\text{ns} + 100\text{ns} = 220\text{ns}$ $T_{TLB} + \text{Tabla de páginas} + \text{Acceso a la página} = 220\text{ns}$
$T_{EFFECTIVO} = 0'80 \cdot 120\text{ns} + 0'2 \cdot 220\text{ns} = 140\text{ns}$ $T_{EFFECTIVO} = (\%_{ACIERTO} \cdot T_{ACIERTO}) + (\%_{FALLO} \cdot T_{FALLO})$	



- Protección

Para proteger la memoria en la paginación se utilizan bits de protección por cada entrada de la tabla de páginas, también se puede incluir un bit para conocer si una página es de Lectura/escritura o sólo Lectura. En caso de intentar escribir en una de solo Lectura se provocaría una interrupción hardware.

EJEMPLO

A modo de ejemplo, se nos pide traducir la dirección lógica de 16 bits:

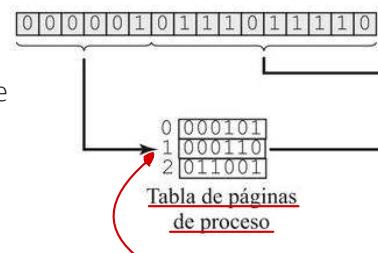
$$0000010111011110_2 \text{ (BINARIO)} = 1502_{10} \text{ (DECIMAL)}$$

También se nos dice que se usan 6 bits de la dirección para el número de página y nos dan su tabla de páginas correspondiente. ¿Cómo se traduciría esta dirección?

0	$000101_2 = 5_{10}$
1	$000110_2 = 6_{10}$
2	$011001_2 = 25_{10}$

Los bits de mayor peso de la dirección determinan el número de página. Como sabemos que se usan 6 bits para el número de página, tomamos esos bits de la dirección [los otros 10 son el tamaño de página]:

$$\underbrace{000001}_{\text{6 bits}} \underbrace{0111011110}_{\text{10 bits}} \Leftrightarrow \frac{1502}{2^{10}} = 1$$



Entonces buscamos el bloque 000001_2 que corresponde a la entrada 1_{10} de la tabla de páginas. La entrada 1 nos dice el marco de memoria que buscamos:

$$\rightarrow 1 \quad \boxed{000110}$$

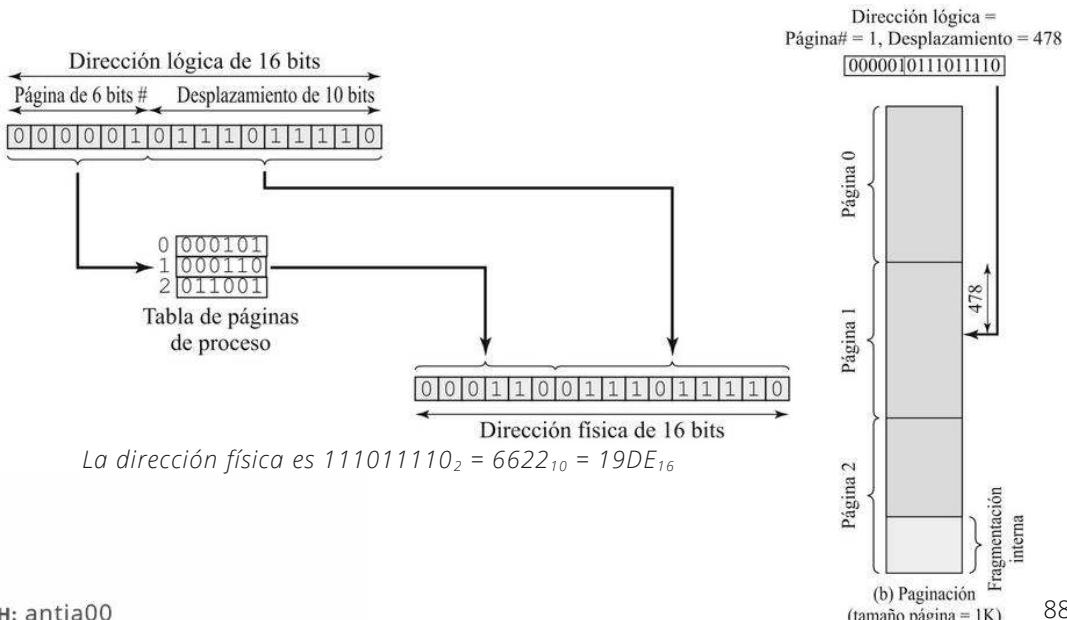
Los otros 10 bits son los de menor peso y estos nos dicen el número de palabra que buscamos dentro de esta página. Esto significa que los 10 bits se utilizarán para el desplazarse dentro del marco en memoria principal, de ahí que se le llame desplazamiento o tamaño de página.

Es en este caso nuestro desplazamiento sería: $0000010111011110_2 = 478_{10}$

Por último, queda combinar el inicio del marco 000110_2 con el desplazamiento 0111011110_2 y obtenemos finalmente la dirección física:

$$\underbrace{000110}_{\text{Marco}} \underbrace{0111011110_2}_{\text{Desplazamiento}} \Leftrightarrow (6 \cdot 2^{10}) + 478 = 6622_{10}$$

Nota: La conversión a decimal de 111011110 es 6622



WUOLAH: antja00

88

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Importante

Puedo eliminar la publi de este documento con 1 coin

→ Plan Turbo: barato
→ Planes pro: más coins

pierdo espacio



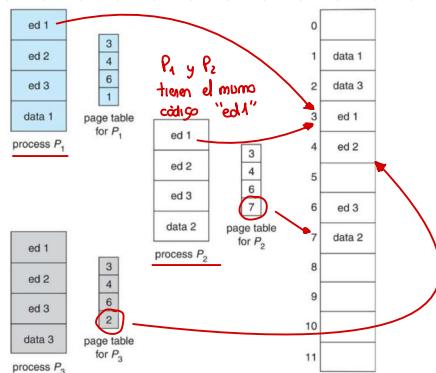
Necesito concentración

ali ali ooooh
esto con 1 coin me
lo quito yo...

wuuuh

- Compartición

Se puede compartir todo aquél código común entre usuarios y que nunca se ve alterado durante su ejecución.



Parte del código normalmente se destina para almacenar datos introducidos por el usuario [data 1, data 2, ...] y otra parte del código se destina al funcionamiento del programa y este puede tratarse perfectamente de código reentrant [ed1, ed2, ed3, ...]

- Segmentación

En la segmentación cada bloque tiene un tamaño diferente, cada segmento está compuesto por un nombre y un número de segmento.

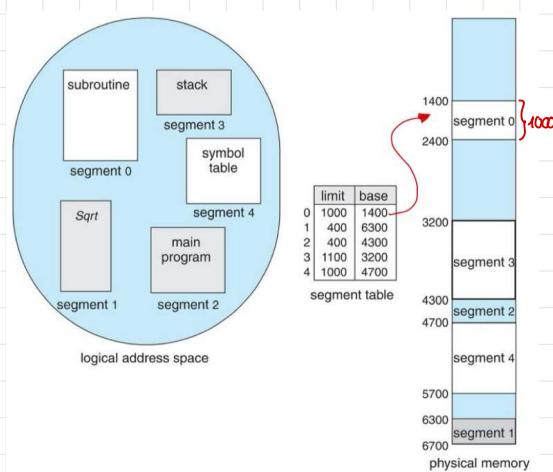
- Traducción

La transformación de direcciones virtuales se realiza gracias a una tabla de segmentos.

Una dirección virtual consta de dos partes: número de segmento y desplazamiento.



La tabla de segmentos se almacena en memoria principal y se accede mediante un registro que apunta a dicha tabla.



- Protección

Una ventaja de la segmentación es la protección de los segmentos. El hardware verifica los bits de protección asociados a cada entrada en la tabla de segmentos para impedir acceso ilegal a memoria.

- Compartición

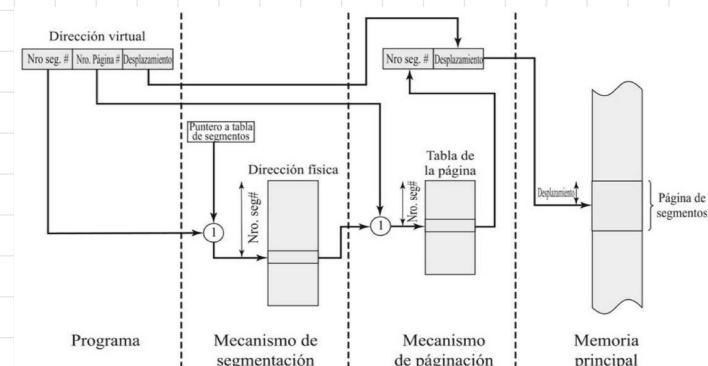
Otra ventaja es la de compartir código y datos. La compartición se produce a nivel de segmento y puede hacerse asignando un segmento a cualquier información.

- Fragmentación

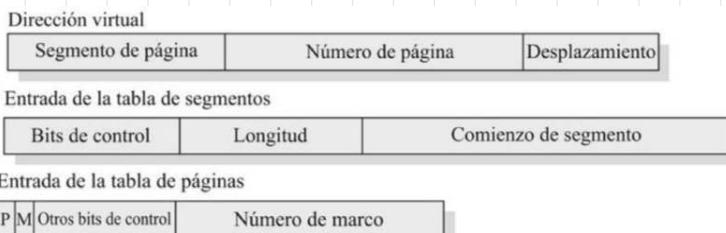
La segmentación ocasiona fragmentación externa, en este caso se puede esperar hasta que haya disponible más memoria o compactar los huecos para crear huecos mejores.

• Segmentación paginada

Al combinar la segmentación con la paginación obtenemos la segmentación paginada, es decir, un segmento donde hay varias páginas.



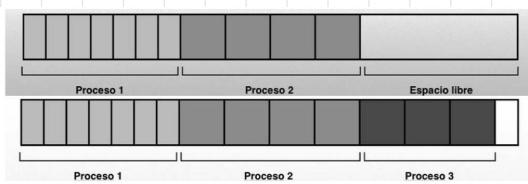
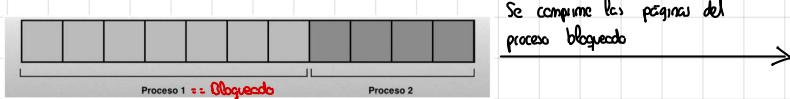
Ahora debemos tener una tabla de páginas independiente para cada segmento de cada proceso. Ahora tenemos una tabla de páginas para cada segmento.



Generalmente, la última página de cada segmento no estará totalmente llena. Por lo que aunque se haya eliminado la fragmentación externa, hemos introducido la fragmentación interna e incrementado la sobrecarga de espacio de la tabla.

• Memoria comprimida

La memoria comprimida es una estrategia que intenta evitar el traspaso de los procesos a disco en un entorno con memoria virtual. Por lo que se comprimen las páginas de los procesos bloqueados o que llevan mucho tiempo sin ejecutarse.



Esto ayuda mucho a maximizar el uso de la memoria RAM, almacenamiento de datos, uso de CPU y mejorando la eficiencia energética.

2. Memoria Virtual

• Definición, ventajas y otros aspectos

Paralelamente se usa por completo un programa y no se suele necesitar todo el código de un programa de forma simultánea.

Básicamente la **Memoria Virtual** es una técnica que permite a un proceso acceder a direcciones de memoria de forma lógica, sin importar cuánto ocupe físicamente en la memoria principal. Esto es realmente útil ya que permite la ejecución de procesos sin que estos tengan que estar completamente cargados en la memoria principal. La memoria virtual tiene los siguientes ventajes e inconvenientes:

Ventajas	Desventajas
Los programas pueden ser más grandes que la memoria física.	No es fácil de implementar.
Libera a los programadores de las preocupaciones de las limitaciones de la memoria.	Usarlo sin cuidado puede disminuir sustancialmente el rendimiento.
Los programas se cargan en memoria más rápido.	
Proporciona un mecanismo eficiente para la creación de procesos.	
Aumenta el grado de multiprogramación* pues no se guarda un proceso entero en memoria y esto permite alojar más procesos y que se pueda encontrar un proceso listo.	* La cantidad de procesos que pueden estar en memoria se llama grado de multiprogramación.

Además, podemos crear las siguientes definiciones:

• **Memoria Virtual:** Especifica de asignación del almacenamiento en el que la memoria secundaria puede ser direccionada como si fuera parte de la memoria principal.

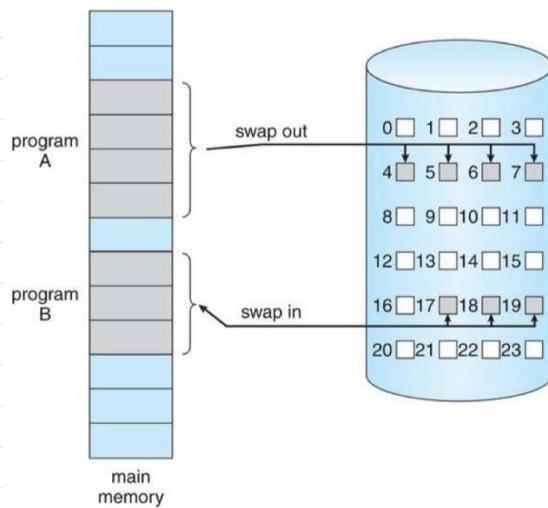
• **Espacio de direcciones virtual:** Es la cantidad de memoria virtual asignada a un proceso.

• Estrategias de gestión de Memoria Virtual

Estrategias de obtención / recuperación	Determinan la transferencia de páginas desde el almacenamiento masivo a la memoria principal.
	<ul style="list-style-type: none"> ● <u>Paginación bajo demanda</u> [enlace] Se espera a que un proceso referencia a una página antes de traerla/lo. Lo normal es que se necesiten más páginas/segmentos al principio pero luego decrecen las peticiones de estos. ● <u>Obtención anticipada / Prepaginación</u> [enlace] Se obtienen por adelantado páginas antes de que un proceso las refiera explícitamente sólo si la probabilidad es alta y hay espacio disponible.
Estrategias de colocación / ubicación	Nos dicen en qué lugar de la memoria principal se debe colocar una página o un segmento entrante. <ul style="list-style-type: none"> ● La decisión de colocación de una página es trivial, porque se puede ubicar una página en cualquier marco disponible de la memoria. ● Los sistemas con segmentación requieren de las <u>estrategias de colocación</u> [enlace] [como en las <u>particiones dinámicas</u> [enlace]].
Estrategias de reemplazo	Deciden qué página o segmento se debe reemplazar por otra/o que necesita entrar en la memoria principal cuando está ocupada por completo [enlace]. Se verá en este tema, en Algoritmos de reemplazo de páginas

- Paginación bajo demanda

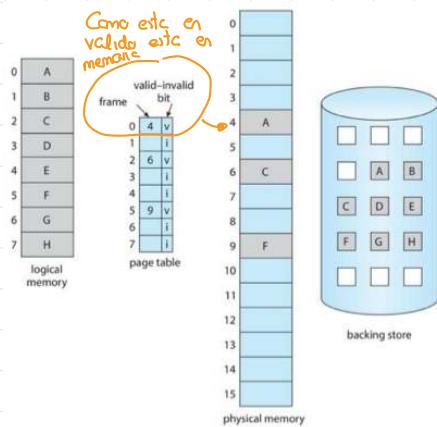
La Paginación bajo demanda decide que páginas estarán en la memoria principal y cuáles en memoria secundaria, la Paginación bajo demanda sugiere mantener todas las páginas de los marcos en la memoria secundaria hasta que sean requeridas, es decir, no se introducirá ninguna página en la memoria principal hasta que la CPU lo requiera.



Cuando el sistema necesita una página se hace referencia a ella y habrá dos casos en los que el paginador fallará:

- Si la referencia es inválida, entonces se aborta la operación
- Si la página no está en memoria principal, entonces se hace swap in.

La paginación bajo demanda requiere de apoyo hardware para traducir direcciones lógicas a físicas, esto se logra mediante un bit de validez, si este bit está a valido indica que la página asociada se encuentra en memoria, si el bit es inválido indica que la página está en disco, una página que está marcada como inválida no tendrá ningún efecto si el proceso nunca intenta acceder a ella.



Es importante también como se gestiona un fallo de página, para ello el Hardware lo cubre y lo hace posible

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

-Tabla de páginas

Para que un proceso lea una palabra de la memoria se utilizan el esquema de paginación y la tabla de páginas.

pierdo espacio



Necesito concentración

ali ali ooooh
esto con 1 coin me
lo quito yo...

wuolah

Entrada de la tabla de páginas

P	M	Otros bits de control	Número de marco
---	---	-----------------------	-----------------

Donde:

- 1) Bit de validez /Presencia. Toma el valor valido - invalido
- 2) Bit de Modificación
- 3) Otros bits de control
- 4) Número del marco en memoria principal

El tamaño que ocupa cada tabla de páginas es calculada con la fórmula: $Tamaño_{TABLA\ PÁGINAS} = N^{\circ}\ entradas \cdot Tamaño\ de\ la\ entrada$

Por ejemplo, imagine una arquitectura donde un proceso pueda tener hasta $2^{31} = 2\ Gbytes$ de memoria virtual.

→ Si sus páginas ocupan $2^9 = 512$ palabras y se dedican 31bits - 9bits = 22 bits a numerar páginas. Esto representa un total de 2^{22} entradas en la tabla de página de cada proceso.

→ Cada entrada de la tabla de páginas al menos debe tener: 22 bits para codificar el número de marco, el bit de presencia y el bit modificación, es decir $22 + 1 + 1 = 24$ bits

Calculamos:

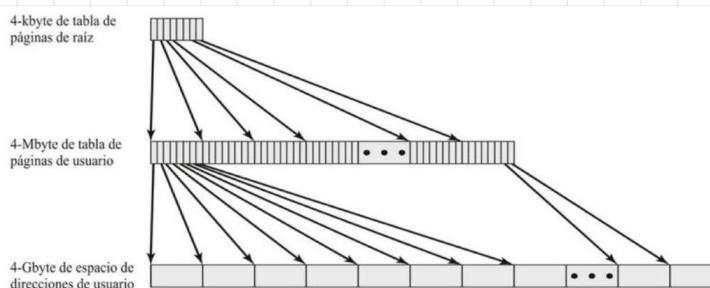
$$Tamaño_{TABLA\ PÁGINAS} = N^{\circ}\ entradas \cdot Tamaño\ de\ la\ entrada = \\ \times \frac{entrada}{proceso} \cdot (Bits\ marco + Bit\ de\ validez + Bit\ de\ modificación) \frac{bits}{entrada} = \\ 2^{22} \frac{entrada}{proceso} \cdot (22 + 1 + 1) \frac{bits}{entrada} = 100663296 \frac{bits}{proceso} \cdot \frac{1\ MB}{2^{23}\ bits} = 12 \frac{MB}{proceso}$$

Serían 12 Mega Bytes por cada Tabla de Páginas, lo cual es demasiado grande.

-Tabla de páginas en dos niveles

Si la cantidad demandada por las tablas de página es demasiado grande se pueden almacenar las tablas de páginas en la memoria virtual.

Algunos procedimientos utilizan un esquema de dos niveles para organizar las tablas de páginas de gran tamaño, en este esquema existe un directorio de páginas, donde cada entrada apunta a una tabla de páginas.



Supongamos un esquema de dos niveles que usa 32 bits para la dirección. Asumimos un direccionamiento a nivel de byte y páginas de tamaño 2^{12} bytes [4 Kbytes], ergo el espacio de direcciones virtuales de 2^{32} bytes [4 Gbytes] se compone de 2^{20} páginas.

Si cada una de las Entrada la Tabla de Páginas [ETP] de un proceso ocupa 4-bytes, su Tabla de Páginas requerirá 2^{20} entrada $\cdot 2^2$ bytes/entrada = 2^{22} bytes = 4 Mbytes

Esta enorme tabla de páginas ocupa 2^{10} páginas [2^{22} bytes / 2^{12} bytes] puede mantenerse en memoria virtual y hacerse referencia desde una Tabla de Páginas Raíz con 2^{10} entradas.

Esta Tabla de Páginas Raíz ocuparía 4 Kbytes ya que tenemos 2^{10} entrada $\cdot 2^2$ bytes/entrada = 2^{12} = 4Kbytes , que coincide con el tamaño de una página [2^{12} = 4Kbytes] y por tanto sólo sería necesaria mantener una sola página en memoria principal.

Los primeros 10 bits de la dirección virtual se usan para la tabla de páginas.

- Si la página no está en memoria principal se produce un fallo de página
- Si la página está en memoria principal, los siguientes 10 bits de la dirección virtual se usan como una paginación simple, es decir, para encontrar en la tabla de páginas de voluntad el número de página se hace referencia desde la dirección virtual original.

• Tabla de páginas invertida

En las tablas de páginas en varios niveles tenemos la desventaja de que el tamaño es proporcional al espacio de direcciones virtuales. Por ello, se invierten las tablas de páginas invertidas, donde se utiliza un valor hash para referenciar el número de página de la dirección virtual, el hash produce un puntero de la tabla de páginas invertida.

Se tiene una entrada por cada marco de página real en lugar de una por cada página virtual, por lo que las tablas de página solo necesitan de fijarse en una parte de la memoria física.

O decir, se tiene una entrada de la tabla de páginas por cada número de marco y no por el número de página virtual.

La entrada de este tipo de tabla de páginas incluye:

- Número de página
- Identificador del proceso
- Bits de control
- Puntero de la página

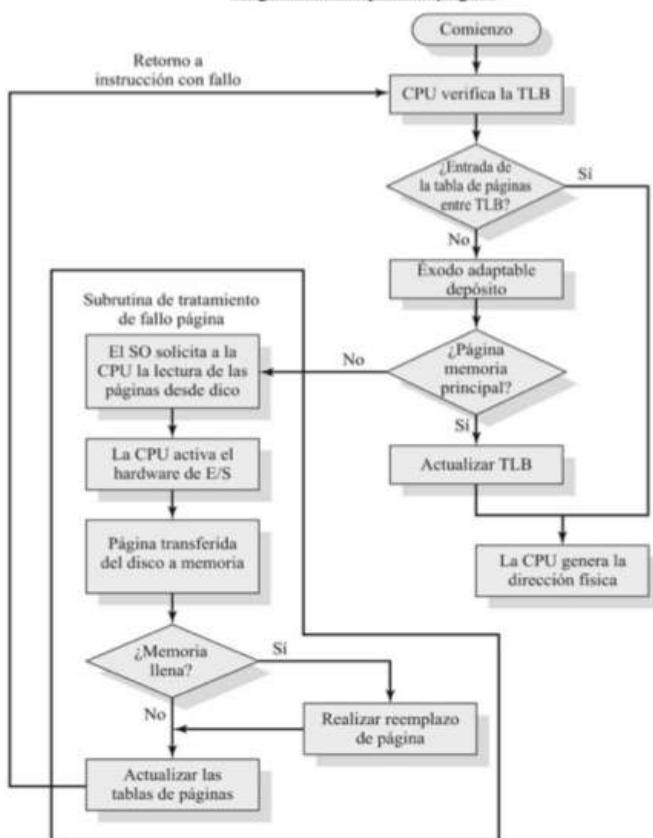
• Gestión de un fallo de página

Un fallo de página ocurre cuando un proceso trata de usar una página que no está en memoria principal.

Procedimiento:

1. Consultamos una tabla interna [que por lo general se conserva en el PCB del proceso] para determinar si la referencia fue un acceso a memoria legal [si es alguna de las páginas posibles del proceso].
2. Si fue ilegal, abortamos el proceso.
Si se trató de una referencia legal pero aún no hemos traído la página, será inválida. La MMU [enlace] lanzará una interrupción que recogerá el SO para ejecutar una subrutina interna para el tratamiento de fallos de página y traer la página faltante.
3. Encontramos un marco libre en la memoria física.
4. Planificamos una operación para leer de disco la página deseada en el marco recién asignado.
5. Cuando concluya la lectura de disco, actualizamos la tabla interna que se conserva junto con el proceso y la tabla de páginas para indicar que ahora la página se encuentra en memoria [con el bit de presencia o bit de válido - inválido].
6. Reiniciamos la instrucción interrumpida por la trama de dirección no-válida. El proceso ahora puede acceder a la página como si siempre se hubiera encontrado en memoria.

Diagrama de un fallo de página



Rendimiento de la paginación bajo demanda

La paginación por demanda **puede sobrecargar el sistema y tener un impacto negativo en el rendimiento.**

Sea p la probabilidad de un fallo de página y que: $0 \leq p \leq 1$

Podemos esperar que p tenga un valor muy cercano a cero; es decir, que habrá sólo unos cuantos fallos de página. Entonces, el tiempo efectivo de acceso es

$$T_{\text{efectivo}} = (1-p) \cdot \text{tiempo}_{\text{ACceso a memoria}} + p \cdot \text{tiempo}_{\text{FALLO DE PÁGINA}}$$

Si tomamos como tiempo promedio de servicio de fallo de página 8 milisegundos y un tiempo de acceso a memoria total de 200 nanosegundos [esto incluye el tiempo de traducción, el tiempo de acceso a la Tabla de Páginas 100nseg y otros 100 nseg del tiempo de acceso a memoria principal], entonces:

$$\begin{aligned} \text{Tiempo efectivo de acceso} &= (1-p) \cdot 200\text{ns} + p \cdot 8\text{ms} = \\ &= (1-p) \cdot 200\text{ns} + p \cdot 8000000\text{ns} = 200\text{ns} + 7999800\text{ns} \cdot p \end{aligned}$$

Si 1 acceso de cada 1000 provoca un fallo de página, el tiempo de acceso efectivo es de 8,2 microsegundos. El computador sería un 40% más lento debido a la paginación por demanda.

Si queremos una degradación menor al 10%, esto es, que el tiempo no sea superior a 220nseg, necesitamos:

$$\begin{aligned} 220 &> 200 + 7999800 \cdot p, \\ 20 &> 7999800 \cdot p, \\ p &< 0.0000025 \end{aligned}$$

Es muy importante mantener **baja** la tasa de fallos de página en un sistema de paginación por demanda.

Marco de página que no se está utilizando ahora mismo

Reemplazo de páginas

Si se da el caso de que se demandan páginas para los que no hay marcos, entonces se debe reemplazar páginas. Si no hay marcos libres **escogemos uno que no se utilice en el momento (victima)** y lo liberamos modificando la tabla de páginas.

Para implementar la paginación por demanda **hace falta**:

- Un algoritmo de asignación de marcos
- Un algoritmo de reemplazo de páginas

Algoritmos de reemplazo de páginas

Vamos a tratar con la siguiente secuencia de referencias y con únicamente 3 marcos disponibles en memoria: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

FIFO (First In First Out)

Se reemplaza el que más tiempo lleva

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	0
F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	
7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	0
0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0
1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1

1	7	0	1
0	7	0	1
1	2	2	1

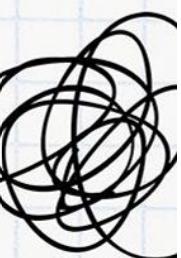
Se ha producido un total de **15** fallos de página.

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

pierdo espacio



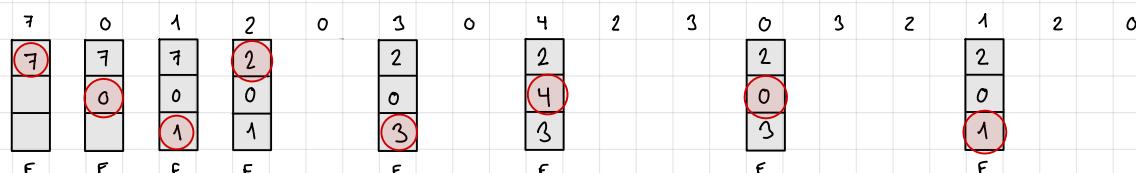
Necesito concentración

ali ali ooooh
esto con 1 coin me
lo quito yo...

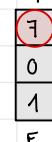
wuolah

• Reemplazo Óptimo

Se recomienda la página que más tarde en ser referenciada en el futuro (esta realmente no se puede llevar a la práctica ya que no se puede predecir).



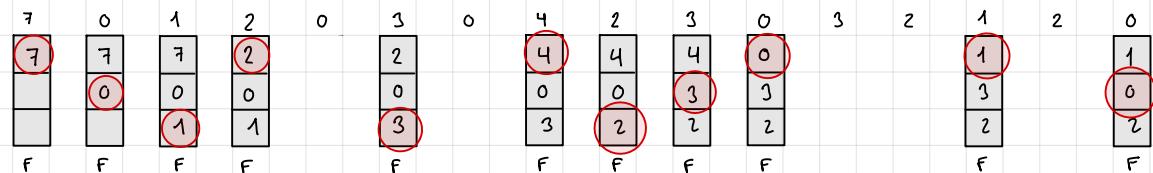
1 7 0 1



Se ha producido un total de 9 fallos de página.

• LRU (Least Recently Used)

Se recomienda la página que lleva más sin utilizar.



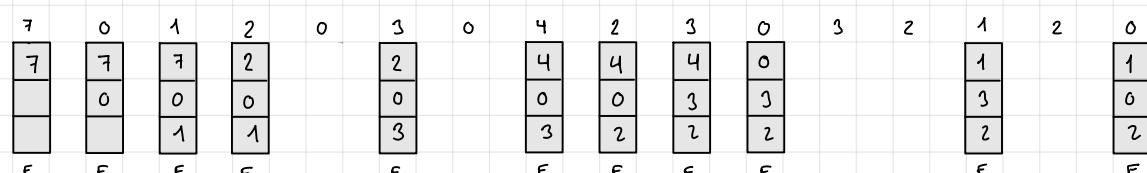
1 7 0 1



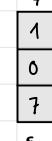
Se ha producido un total de 12 fallos de página.

• LRU con bits adicionales

NUNCA CAC



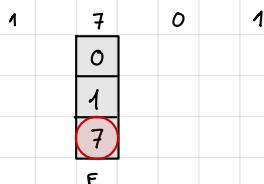
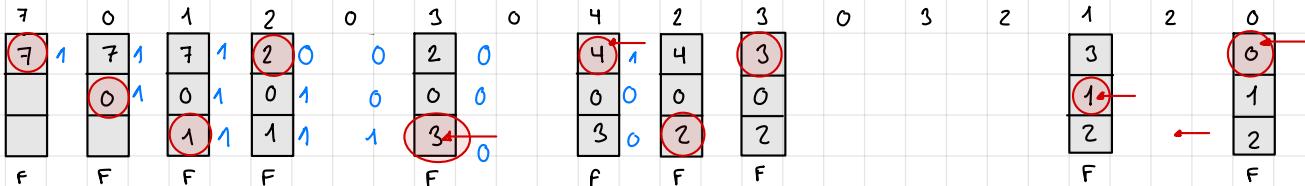
1 7 0 1



Se ha producido un total de 12 fallos de página.

•Algoritmo del Reemplazo

Se emplea el bit de referencia R para evitar deshacerse de la página más antigua. Se sustituye la página que es apuntada por el puntero.



Se ha producido un total de 11 Falta de página.

•LFU (Least Frequently Used)

Se tiene un contador para el número de referencias de cada página, se reemplaza la página con menor recuento, es decir, la página con el menor valor de contador.

•MFU (Most Frequently Used)

Se tiene un recuento en las páginas, se piensa que las páginas con menor recuento aún tienen que utilizarse por lo que se borra la página con mayor recuento.

•NRU (Not Recently Used)

Los ordenadores emplean un bit de referencia (R) y un bit de modificación (M). Al iniciar un proceso el SO asigna un 0 a ambos bits.

Si ocurre un fallo de página, el SO inspecciona todas las páginas y las divide en las siguientes 4 categorías:

- Clave 0: [0,0] NO referenciada, NO modificada.
- Clave 1: [0,1] NO referenciada, SÍ modificada.
- Clave 2: [1,0] SÍ referenciada, pero NO modificada.
- Clave 3: [1,1] SÍ referenciada, y SÍ modificada.

Donde el algoritmo NRU elimina una página no vacía de Clave 1 de manera aleatoria con el número más pequeño.

• Asignación de marcos

- N° minimo de marcos

El número mínimo de marcos es definido por la arquitectura del conjunto de instrucciones. Recuérdalo, que cuando ocurre un fallo de página, antes de terminar la ejecución de una instrucción, ésta debe reiniciarse. Destacar que el número máximo de marcos está definido por la cantidad de memoria física disponible.

- Algoritmos de asignación de máscaras

Se puede realizar una **asignación equitativa**, por ejemplo: Si tenemos 93 marcos y 5 procesos, cada proceso recibirá $93 \div 5 = 18$ marcos.

También se puede hacer una **asignación proporcional**, por ejemplo:

Si un pequeño proceso de 10K y una base de datos interactiva de 127K son los únicos procesos que se ejecutan en un sistema con 62 marcos libres, no tiene mucho sentido asignar 31 marcos a cada proceso. El proceso pequeño no necesita más de 10 marcos, por lo que los 21 restantes se desperdician.

Para la **asignación proporcional** dividiríamos los 62 marcos entre dos procesos, uno de 10 páginas y el otro de 127. Así asignamos 4 y 57 marcos a cada proceso, respectivamente:

$$\frac{10}{137} \cdot 62 = 4 \text{ marcos}, \quad \frac{127}{137} \cdot 62 = 57 \text{ marcos}$$

- Reemplazo global vs Reemplazo local

Podemos clasificar los algoritmos de reemplazo de páginas en dos categorías:

El reemplazo global permite que SO seleccione un marco para reemplazar de entre todo el conjunto de marcos, incluso si está asignado a otro proceso, mientras que con el reemplazo local requiere que el SO seleccione un marco de los asignados al proceso.

Con el reemplazo local, el número de marcos asignados a un proceso no cambia o podría aumentar si hay marco libre, mientras que con el reemplazo global es posible que un proceso solo seleccione marcos que pertenezcan a otros procesos, incrementando el número de marcos asignados.

Mediante el **reemplazo global** un proceso no puede controlar su propia tasa de fallos de página, esto no sucede con el **reemplazo local** ya que el conjunto de páginas en memoria para ese proceso solo depende de su comportamiento en cuanto a paginación.

Por todo ello, podemos considerar que el reemplazo global generalmente brinda una mayor productividad.

• Hyperpigmentation

Una gran otra actividad de paginación es denominada **hiperspaginación** o **thrashing**, un sistema se encuentra en hiperspaginación si este más tiempo paginando que ejecutando.

- (cont'd)

La hiperexcitación tiene (**importante**) problema de rendimiento.

Los efectos de la hipergeneración se puede limitar utilizando un algoritmo de reemplazo local, ya que si un proceso comienza a hipergenerar, no puede liberar marcas de otro proceso y provocar que este entre también en hipergeneración. Si los procesos están en hipergeneración, los nuevos nodos del timeline pueden emmigrar en la coda del dispositivo de memoria.

Para concretar la ciudadanía de marca que un grupo permite se emplea el modelo del conjunto de trabajo que este basado en el acuerdo de **flexibilidad**.

- (excluded)

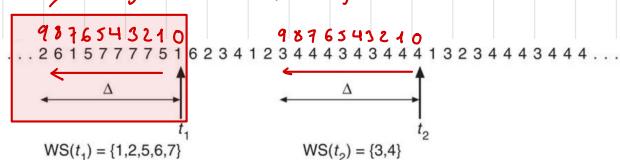
Una **localidad** es un conjunto de páginas que se utilizan conjuntamente. Un proceso durante su ejecución pone de una localidad a otra, estando entonces compuesto por varias localidades distintas que pueden superponerse.

Si asignamos menos marcos que el tamaño de la localidad, el proceso entrará en hipoperación ya que no puede mantener en memoria todas las páginas que está usando en ese momento.

- Modelo del conjunto de tubos

El modelo del conjunto de trabajo o Working Set Model, etc basado en el concepto de localidad. Este modelo usa el parámetro Δ para poder examinar las referencias más recientes a pagina.

Ejemplo: Con = 10 :



Si  es abarcado negativo no abarcaré todo el conjunto de trabajo y si es demasiado grande puede saltar la varia facilidades

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

pierdo
espacio



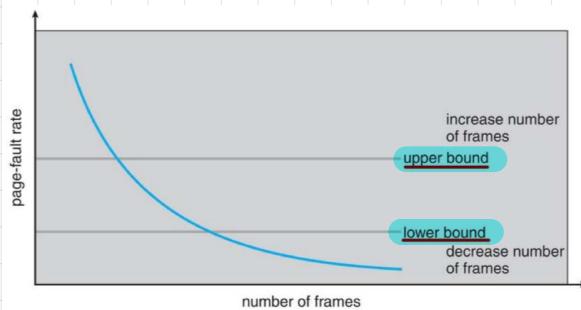
Necesito
concentración

ali ali ooooh
esto con 1 coin me
lo quito yo...

wuolah

- Frecuencia de fallos de página

Para evitar la hipergeneración, es decir, la elevada tasa de fallos de página, se pueden establecer límites superiores e inferiores para la tasa deseada de fallos de página.



- Si la tasa de fallos excede el upper bound, asignarán más marcos a ese proceso
- Si la tasa de fallos es inferior al lower bound, quitánes marcos al proceso

- Otras consideraciones

• Prepaginación: Estrategia que intenta el alto nivel de paginación inicial. Consiste en traer a memoria al mismo tiempo todas las páginas que se necesitaran.

• Fijación de páginas de E/S: Permite que las páginas se fijen en memoria, por ello a cada marco se le asigna un bit de fijación, si se ha fijado el marco no se puede seleccionar para reemplazo. Cuando termina la E/S se desactiva la fijación de las páginas. El uso del bit de fijación puede ser peligroso si se activa y nunca se desactiva.

- Tamaño de página

Tema 4. Dispositivos de E/S y ficheros

1- Gestión de Entrada/Salida

Los dos trabajos principales de un ordenador son la E/S y la computación. Un papel del SO es gestionar y controlar las distintas operaciones y dispositivos de E/S.

Definición de Entrada/Salida

E/S es cualquier transferencia de información desde o hacia la memoria o el procesador. Comprende la transferencia entre los distintos niveles de memoria y la comunicación con el exterior mediante periféricos.

El rendimiento es clave para el funcionamiento de la E/S, si se hace correctamente se evitan que sucedan fallos constantemente.

Los dispositivos de Entrada/Salida

Se pueden agrupar en 3 categorías:

- Legibles para el usuario: Para la comunicación del usuario con el computador (ratón, teclado...)
- Legibles para la máquina: Para la comunicación con equipamiento electrónico (cintas, sensores, controladores...)
- Comunicación: Para la comunicación con dispositivos remotos (Tarjetas de red...)

Entre las categorías o incluso dentro de cada una encontramos diversas diferencias: Velocidad, Unidad de transferencia, Representación de los datos, Operaciones permitidas, Condiciones de error

Objetivos de diseño

Para diseñar un dispositivo de Entrada/Salida necesita de unos objetivos que son:

- Independencia del código de los caracteres
- Independencia del periférico
- Eficiencia,
- Tratamiento uniforme de los periféricos

Esto causa una serie de consecuencias.

La información de un dispositivo se puede almacenar en su descriptor y almacena:

- La identificación del dispositivo
- Un puntero a tablas de traducción de caracteres
- Proceso de usuario en curso
- Instrucciones a través de los que actúa
- El estado actual (ocupado, libre o estropeado)

Sistema de E/S del Kernel

El kernel proporciona muchos servicios relacionados con la E/S, basados en la estructura del controlador del dispositivo y del hardware. Vamos a ir explicando cada uno de estos servicios:

Planeación de la E/S

Se debe determinar un orden adecuado para ejecutarlas. Esto mejora el rendimiento global del sistema y reduce el tiempo de espera medio requerido para que la E/S se complete.

Los SO tienen una cola de espera de solicitudes para cada dispositivo. El planificador recorre la cola para mejorar la eficiencia del sistema y el tiempo medio de respuesta.

Un kernel con E/S asincrona controla múltiples solicitudes de E/S al mismo tiempo. Por lo que tiene una tabla de estado del dispositivo con una entrada por cada dispositivo.

Almacenamiento en buffer

Un buffer es un área de memoria que almacena datos mientras se está transfiriendo entre dos dispositivos o entre un dispositivo y una aplicación. Esto se realiza para:

- Adaptar las velocidades entre 2 dispositivos
- Adaptar dispositivos con diferentes tamaños de transferencia de datos
- Permitir la semántica de copia, es decir, copiar los bytes que se quieren escribir o leer a un buffer del kernel para evitar que se sobrescriba

Almacenamiento en cache

Una cache es una región de memoria rápida que contiene copias de ciertos datos, ya que se accede más rápido que donde están almacenadas los datos originalmente.

Un buffer es utilizado para retener temporalmente un dato mientras que la cache es un almacenamiento donde se copian los datos para acceder más rápidamente. El kernel primero mira en la cache

- Spooling

Los dispositivos pueden clasificarse en: Compatibles, No compatibles

En caso de los no compatibles, si el dispositivo ya está asignado deberá esperar a que este quede libre. Para repartir la carga de estos dispositivos y evitar cuellos de botella se adopta algún mecanismo como el spooling. El spooling es una estrategia donde en lugar de efectuar la transferencia directamente al dispositivo asociado con un dispositivo virtual, se lleva a cabo sobre un soporte intermedio para actuar como una cola de operaciones de E/S. El que transfiere la información del disco al dispositivo es el spooler asociado al dispositivo.

- Tratamiento de errores

Una llamada de E/S al sistema devolverá un bit de información sobre el estado de la llamada, en UNIX es una variable entera llamada errno.

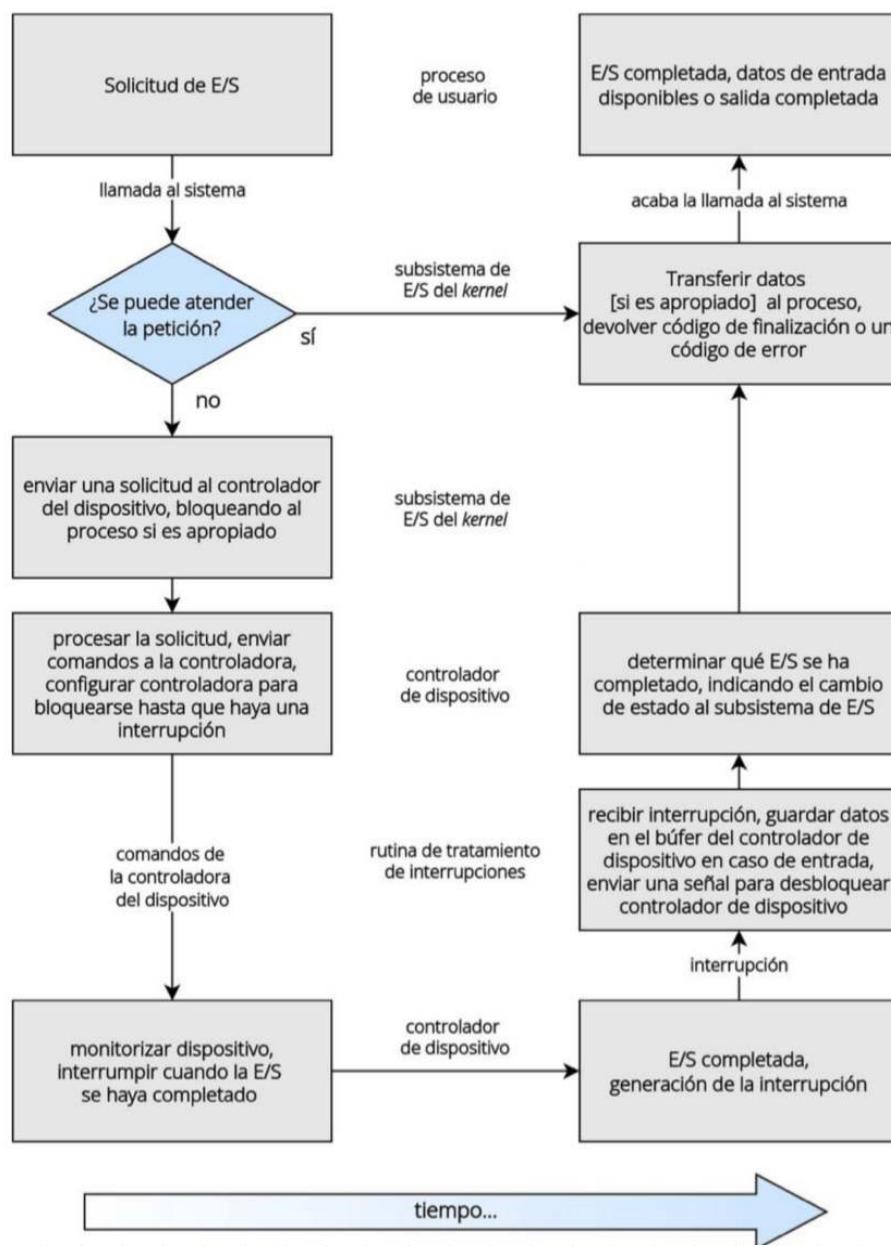
- Protección de E/S

Las instrucciones de E/S no pueden ser ejecutadas por un usuario, por ello son definidas como instrucciones privilegiadas. Por ello, para ejecutar instrucciones de E/S se hace solicitando una llamada al sistema y el sistema las realiza en modo núcleo. Se debe utilizar el sistema de protección para evitar los accesos a los usuarios.

- Estructuras de datos del Kernel

El kernel emplea diversas estructuras de datos internas como la tabla de Archivos abiertos.

• Diagrama de solicitud de una E/S



Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

2 - Planificación de HDD

pierdo espacio



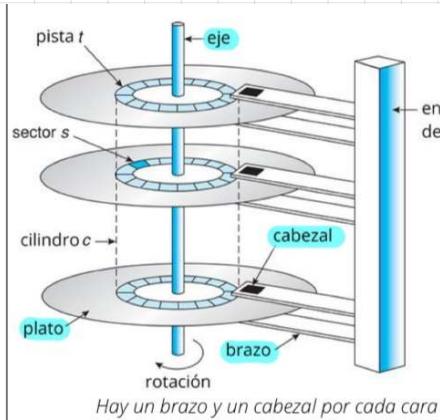
Necesito concentración

ali ali ooooh
esto con 1 coin me
lo quito yo...

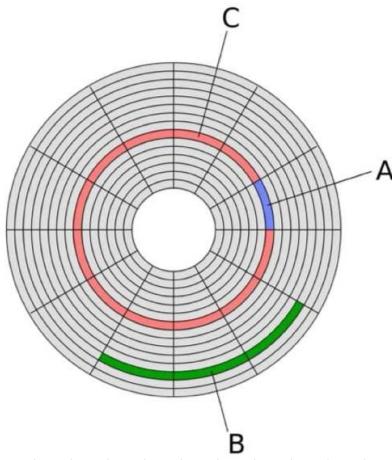
wuolah

Tenemos dos tipos de almacenamiento: Disco Duro (HDD) y Disco de estado Sólido (SSD), donde el SSD es más caro en el almacenamiento y el HDD es más barato.

• Parámetros de un HDD



Además, la estructura interna de un plato es la siguiente:



Donde tiene:

- Sector (A): Es cada una de las divisiones de una pista.
- Cluster (B): Es un conjunto contiguo de sectores.
- Pista (C): Es un anillo de sectores del plato.
- Cilindro: Acceso simultáneo de una pista en todos los discos

El Tiempo de Acceso (T_A) a un registro del disco depende de:

- Tiempo de Búsqueda (T_B): Tiempo requerido para mover la cabeza desde el cilindro actual al cilindro deseado.

$$T_B = m \cdot n + s$$

m: Constante del HDD

n: Cantidad de pistas requeridas

s: Tiempo de arranque

- Tiempo de Latencia (T_L): Tiempo esperado para situar el sector requerido por debajo del cabezal de lectura/escritura.

$$T_L = \frac{1}{2} \cdot \frac{60}{RPM}$$

- Tiempo de Transferencia (T_T): Tiempo necesario para recoger un sector y transferir los datos de él.

$$T_T = \frac{b}{N} \cdot \frac{60}{RPM}$$

b: número de bytes a transferir

N: número de sectores en una pista

• Políticas de planificación de disco

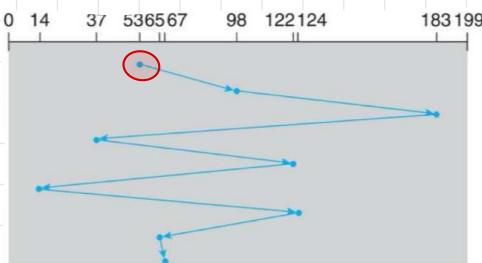
Para valorar una planificación tenemos diferentes criterios:

- Productividad
- Tiempo promedio de respuesta
- Varianza de los tiempos de respuesta

Tenemos las siguientes políticas de planificación de disco:

• FIFO o FCFS

Se trata la primera solicitud, es decir, la primera solicitud es la primera en ser servida. No suele dar un buen tiempo promedio.



98, 183, 37, 122, 14, 124, 65 y 67

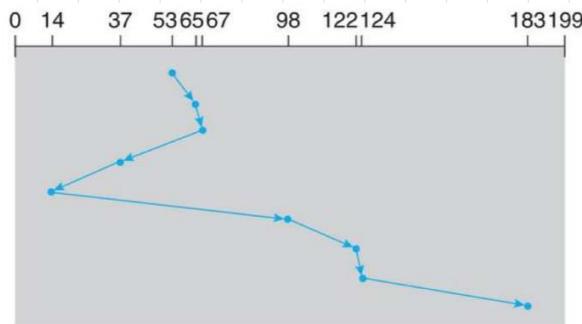
Y teniendo en cuenta que el cabezal se sitúa en la pista 53

La cabeza de lectura-escritura se moverá primero de la pista 53 a la 98, luego a la 183, 37, 122, 14, 124, 65 y por último a la 67, lo que genera un movimiento total de 640 pistas por parte del brazo del disco duro:

$$(98-53) + (183-98) + (183-37) + (122-37) + (122-14) + (124-14) + (124-65) + (67-65) = 640 \text{ pistas}$$

• SSTF (Shortest Seek Time First)

Selecciona la solicitud más cercana a la posición actual de la cabeza.



Las pistas más exteriores e interiores suelen recibir una atención deficiente.

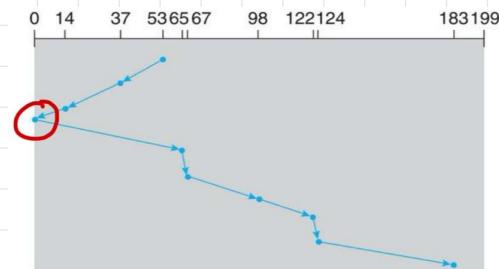
Esta es útil en sistemas de procesamiento por lotes ya que la productividad es lo más importante.

La solicitud más próxima a la posición inicial es la pista 65. Una vez ubicados en la pista 65, la siguiente pista más cercana es la 67.

Este método de planificación da como resultado un movimiento total de sólo 236 pistas, es decir, cerca de $\frac{1}{3}$ de distancia de lo que tomó recorrer con FIFO.

• SCAN (Algoritmo del Ascensor)

SCAN elige la solicitud más cercana pero en una dirección predefinida. SCAN no cambia de dirección hasta que llega a la pista más exterior o hasta que no haya más solicitudes pendientes en esa dirección.



Para el ejemplo que nos ocupa, necesitamos saber la dirección de movimiento de la cabeza, así como su posición más reciente. Aquí hemos supuesto que la cabeza se desplaza hacia las pistas de menor número y se parte de la 53.

La secuencia de pistas recorridas con este algoritmo son:

37, 14, [desciende hasta 0 esté en la cola o no], 65, 67, 98, 122, 124 y 183.

En total se han recorrido 236 pistas.

• N-SCAN

El brazo se mueve en una y otra dirección como SCAN, pero solamente se atienden las solicitudes que ya estaban esperando cuando se inició un barrido específico. En cada barrido se atienden las N peticiones.

• C-SCAN

Es otra modificación del SCAN básico, en C-SCAN el brazo se mueve del cilindro exterior al cilindro interior, atendiendo las solicitudes más cercanas. Cuando llega al cilindro exterior salta directamente al interior.



La secuencia de pistas recorridas con este algoritmo son:

65, 67, 98, 122, 124, 183, [llega a la 199], [vuelve a 0], 14 y 37

En total se han recorrido 382 pistas.

C-SCAN se puede llevar a la práctica de manera tal que las solicitudes que lleguen durante un barrido sean atendidas en el siguiente barrido.

Algunos resultados de simulaciones indican que la mejor política de planificación de disco podría operar en dos etapas. Cuando la carga es ligera, la política SCAN es la mejor. Cuando la carga es mediana o pesada, C-SCAN produce los mejores resultados.

• Planificación de Disco en Linux

El planificador CFQ divide los procesos en 3 clases separadas: Tiempo real, El mejor esfuerzo y Ocio.

• Cache de disco

La memoria cache es una memoria más pequeña y rápida que la memoria principal. La memoria cache reduce el tiempo medio de acceso gracias al principio de localidad.

Al hacerse la petición de E/S para un bloque se comprueba si está en la cache del disco, si es así se cumple con la cache, en otro caso se lee el sector pedido de disco y se coloca en la cache.

• Consideraciones de diseño

Cuando una petición de E/S es servida con la cache, los datos son enviados:

- El envío puede hacerse por una transferencia en memoria del bloque de dato, desde la cache del disco a la memoria asignada.
- Puede usarse la memoria compartida y pasar un puntero a la entrada apropiada de la cache del disco.

Otra cuestión es la estrategia de reemplazo, ya que cuando se tira un nuevo bloque a la cache del disco, uno de los bloques existentes debe ser sustituido. El más común es LRU, reemplazándose el bloque que lleva más tiempo en cache sin ser referenciado.

• Consideraciones de rendimiento

El rendimiento de la cache será bueno siempre que la tasa de fallos sea mínima, esto dependerá de la localidad de las referencias al disco, del algoritmo de reemplazo y otros factores de diseño.

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

• Estructura RAID

RAID (Redundant Array o) Independent Disk) es un conjunto redundante de discos independientes, es un sistema de almacenamiento que usa múltiples discos duros.

Para mejorar la fiabilidad, optando por la redundancia y la técnica más sencilla que es duplicar en espejo.

Con múltiples discos se mejora la velocidad de transferencia dividiendo los datos entre distintos discos.

Existen 6 niveles RAID:

pierdo espacio



Necesito concentración

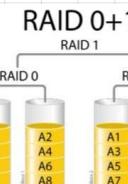
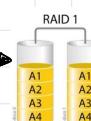
ali ali ooooh
esto con 1 coin me
lo quito yo...

wuolah

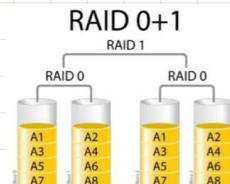


- RAID 0: Máticos de discos distribuyendo en el nivel de bloques pero sin redundancia

- RAID 1: Conjunto de datos en espejo, se crea una copia exacta del conjunto de datos

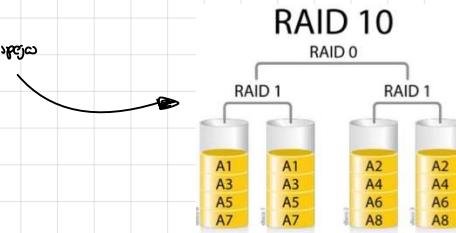


- RAID 5: Conjunto de datos dividido en bloques con paridad distribuida. Consiste en distribuir tanto los bloques como la paridad, bloques distribuidos entre n-1 discos

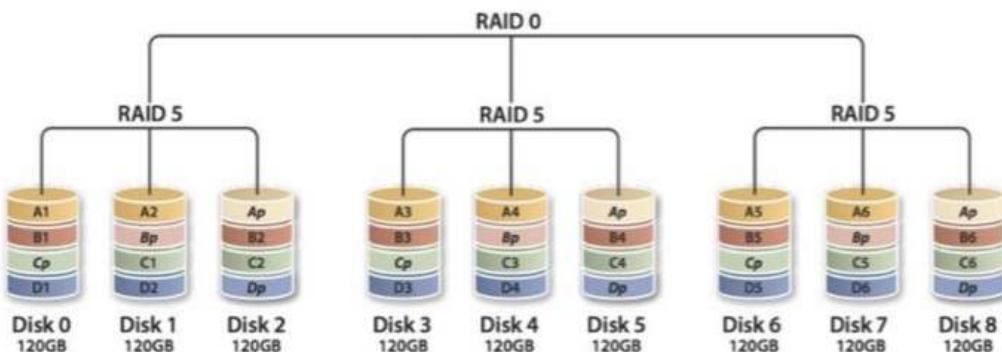


- RAID 0+1: RAID usado para replicar y compartir datos entre varios discos. Primero se crean dos conjuntos RAID 0 y sobre esto se crea un conjunto RAID 1

- RAID 1+0: Primero se hace un espejo y luego se distribuye los espejos



- RAID 50: Combina un RAID 0 con la paridad distribuida de un RAID 5, siendo un RAID 0 dividido de elementos RAID 5. Un disco de cada conjunto RAID 5 puede fallar sin que se pierdan datos



wuolah

Paridad

Los códigos de paridad son utilizados en telecomunicaciones para detectar y corregir errores en la transmisión, para ello se añade el bit de paridad. El bit de paridad será un 0 si el número total de unos a transmitir es par y un 1 para un número impar de unos.

3-Sistema de ficheros

La información se almacena en archivos o ficheros, los procesos pueden leer y escribir nuevos. La información almacenada en los ficheros debe ser persistente, es decir, no debe verse afectada por la creación y terminación de un proceso.

Los ficheros son administrados por el sistema operativo, la parte del SO que trabaja con ficheros es conocido como sistema de ficheros.

Para un usuario, lo más importante es el nombre y como se protegen los ficheros.

• El sistema de Ficheros y el Usuario

-Aspectos básicos sobre Ficheros

•Nombre de un Fichero

Cuando un proceso crea un archivo le da un nombre. Muchos sistemas emplean nombres de archivo con dos partes separadas por un punto, parte la cual indica la extensión del archivo.

•Tipos de Ficheros

UNIX distingue entre los siguientes tipos de ficheros:

- Ficheros regulares u ordinarios: Son archivos que contienen información que ha introducido un usuario
- Directorios: Son archivos ordinarios pero con una estructura concreta, lista una serie de archivos
- Especiales: Usados para acceder a los dispositivos periféricos. Pueden ser de carácter o de bloque
- Tubo con nombre: Utilizados para el intercambio de información entre las aplicaciones ejecutadas con nombres de usuarios y permisos diferentes
- Enlaces simbólicos: Es una referencia a otro archivo.

•Atributos asociados a los Ficheros

Cada archivo tiene su nombre y datos, además, algunos SO dan información adicional como la fecha, hora de creación y tamaño, estos diferentes elementos son conocidos como atributos.

-Operaciones con Ficheros

Los SO ofrecen diversas operaciones realizadas mediante llamadas al sistema. Algunas operaciones son:

1. CREATE. Crea un archivo
2. DELETE. Elimina el archivo
3. OPEN. Conecta el archivo al proceso
4. CLOSE. Desconecta el archivo del proceso
5. READ. Lee información del archivo
6. WRITE. Escribe información en el archivo
7. APPEND. Escribe información al final
8. SEEK. Se posiciona en un archivo de acceso aleatorio

•Ficheros mapeados en memoria

Algunos SO proporcionan una forma de asociar los ficheros con un espacio de direcciones de un proceso en ejecución, esto es mediante dos nuevas llamadas al sistema: MAP y UNMAP, la primera utiliza un nombre de archivo y una dirección virtual, haciendo que el SO asocie el archivo con la dirección virtual.

•Organización lógica de los Ficheros

Los ficheros pueden estructurarse de diversas formas:

-Una serie no estructurada de bytes

El SO lo único que ve son bytes, no se interesa por el contenido, tanto UNIX como Windows lo ve así

-Una secuencia de registros de longitud fija

La operación de lectura obtiene un registro y la de escritura escribe sobre un registro o añade uno nuevo

-Un árbol de registros (ISAM)

Cada registro tiene una longitud diferente, por ello cada uno tiene un campo clave. El árbol es ordenado por esa clave, esto es realizado para optimizar las búsquedas.

• Estructura del sistema de Ficheros

Los usuarios generalmente agrupan sus ficheros de manera lógica mediante directorios, con este mecanismo cada usuario puede tener tanto directorios como necesite para agrupar los ficheros.

Hay dos métodos para acceder a un fichero que se encuentra interno en varios directorios:

• Ruta absoluta: Es la ruta que hay que seguir desde el directorio raíz hasta ese fichero

• Ruta relativa: Los usuarios eligen un directorio actual, por lo que la ruta hasta llegar al fichero deseado se hace partiendo de dicho directorio.

• El sistema de ficheros y el Sistema Operativo

El SO se concentra en el diseño como:

- El almacenaje de ficheros y directorios
- La gestión del espacio de disco libre
- Logra una gestión del sistema de ficheros más eficiente y flexible

• El tamaño de bloque

Los ficheros suelen almacenarse en discos, existen dos métodos para almacenar un fichero de n bytes:

1. Utilizar un número de bytes consecutivo. Tiene el inconveniente de que si crecen hoy que moveo a otra área del disco.

2. Subdividir el fichero en una cantidad de bloques (la más utilizada ya que es menos costosa) que no tienen porque ser contiguos

Lo mas normal para elegir el tamaño de bloque es 512B, 1KB, 2KB o 4KB.

• Asignación del espacio actual a ficheros

Un aspecto clave es decidir como asignamos espacio a nuestros ficheros, esto debe hacerse usando el espacio de disco lo más eficientemente posible y manteniendo mínimos los tiempos de acceso a los ficheros.

- Asignación Contigua

Este enigma de asignación almacena cada fichero como un conjunto de bloques de datos adyacentes en el disco

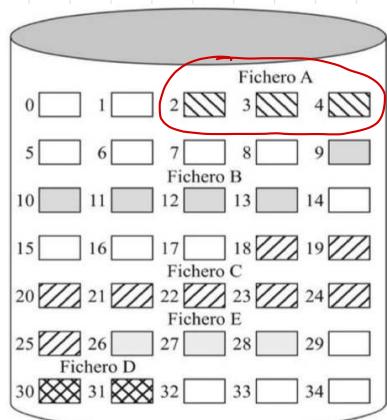


Tabla de asignación de ficheros

Nombre de fichero	Bloque inicial	Longitud
Fichero A	2	3
Fichero B	9	5
Fichero C	18	8
Fichero D	30	2
Fichero E	26	3

Tiene fragmentación externa, que puede solucionarse mediante la compactación pero tiene el problema de que los ficheros crecen por lo que es difícil saber el espacio que necesitarán, en este caso se tendría que reasignar

- Asignación Enlazada

Se almacenan los bloques de un fichero como una lista enlazada

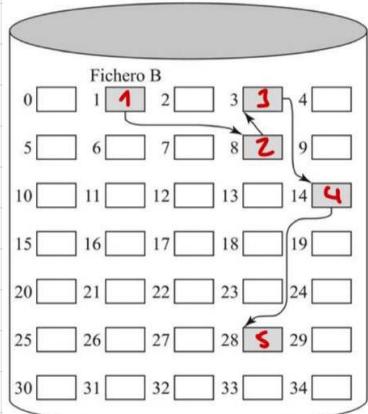


Tabla de asignación de ficheros

Nombre de fichero	Bloque inicial	Longitud
•••	•••	•••
Fichero B	1	5
•••	•••	•••

La asignación enlazada no permite el acceso directo porque tenemos bloques directos por todo el disco y punteros dispersos por todo el disco

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

pierdo espacio



Necesito concentración

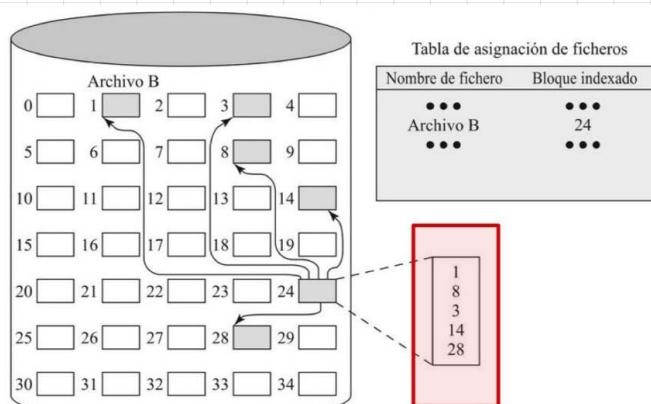
ali ali ooooh
esto con 1 coin me
lo quito yo...

wuolah

Asignación indexada

La asignación enlazada no permite soportar un acceso directo eficiente ya que los punteros a los bloques están dispersos en cada bloque del disco.

Para ello se inventó la asignación indexada, que reúne todos los punteros en una sola ubicación: el bloque índice/indexado.



Cada archivo tiene su propio bloque indexado, que es un vector de direcciones. Con este método se desperdicia memoria ya que un puntero de bloque indexado es por lo general mayor que cualquier puntero de la asignación enlazada.

Cada fichero posee una pequeña estructura de datos conocida llamada i-nodo que contiene la siguiente información:

- Identidad del propietario del fichero y grupo
- Los permisos del fichero
- El tamaño del archivo y sus zonas en el disco
- Fecha de creación, última modificación y último acceso
- Cantidad de enlaces duros a dicho fichero

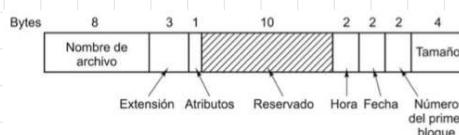
Estructura interna de los directorios

Los sistemas de directorios se organizan en sistemas con árboles jerárquicos, donde cada entrada tiene:

- Nombre del fichero y el número de su primer bloque
- Número del i-nodo en el caso de UNIX

Ejemplos de organización real de directorios

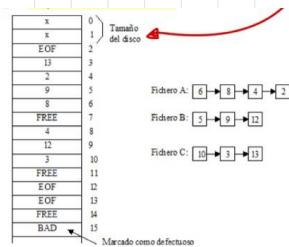
Cada directorio está compuesto de 32 bytes:



La FDT es una lista enlazada donde cada entrada referencia a los bloques de datos del dispositivo.

Las dos primeras posiciones especifican el tamaño del disco, el resto de posiciones:

- EOF: Último bloque de datos de un fichero
- FREE: Bloque de datos libre
- BAD: Bloque de datos defectuoso



Los directorios son ficheros donde cada entrada es de 16 bytes, donde 14 de ellos son del nombre del archivo y 2 para el número del i-nodo.

Cada directorio tiene las entradas .. y .. que indican el directorio actual y el directorio padre respectivamente.

wuolah

Ficheros compartidos

Se puede conectar un directorio a otro, esto es realizado mediante lo denominado como enlace, tenemos dos tipos de enlaces:

- Enlace duro. Mediante este enlace se apunta directamente a la estructura de datos, por lo que los directorios tienen el mismo i-nodo.
- Enlace simbólico. Se apunta a un directorio propio el cual contiene un archivo que almacena la ruta para acceder al fichero apuntado, por lo que se tienen números de i-nodo diferentes.

Gestión del espacio libre

Una vez elegido el tamaño del bloque, la siguiente cuestión es como llevar la cuenta de los bloques libres, para ello se tiene dos opciones:

- Una lista encadrada. Cada bloque contiene tantos números de bloques de disco libre como lo necesite.
- Un mapa de bits. Se busca mejorar el acceso directo, eliminando las diferencias de tiempo entre acceso directo y secuencial, un bloque libre es representado con un 1 y un bloque ya asignado con un 0. El mapa de bits emplea 1 bit por bloque.

Fiabilidad del sistema de ficheros

Cuando se tiene una lista de bloques defectuosos se tienen dos soluciones:

- Solución Hardware: Dedicar un sector en el disco a almacenar la lista de bloques defectuosos.
- Solución Software: Se construye un fichero que contenga todos los bloques defectuosos del disco.

Cohesión del sistema de ficheros

Es fundamental mantener la coherencia del sistema de ficheros.

Casi todos los sistemas disponen de un programa de servicio que comprueba la coherencia interna del sistema de ficheros, esto se puede autoactualizar cada vez que se inicie el sistema, especialmente cuando se ha caído.

Se tienen dos tipos de comprobaciones de coherencia:

Cohesión de bloques

El programa construye una tabla con dos contadores inicializados a 0, el primero cuenta las veces que el bloque está en el fichero y el segundo registra el número de veces que está presente en la lista de bloques libres.

Número de bloque	Bloques en uso	Bloques libres
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	1 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0	0 0 1 0 1 0 0 0 0 1 1 0 0 0 1 1
(a)		

Número de bloque	Bloques en uso	Bloques libres
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	1 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0	0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1
(b)		

Número de bloque	Bloques en uso	Bloques libres
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	1 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0	0 0 1 0 2 0 0 0 0 1 1 0 0 0 1 1
(c)		

Número de bloque	Bloques en uso	Bloques libres
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	1 1 0 1 0 2 1 1 1 0 0 1 1 1 0 0	0 0 1 0 1 0 0 0 0 1 1 0 0 0 1 1
(d)		

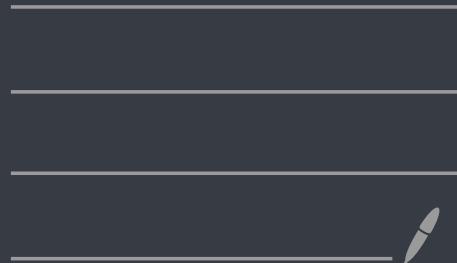
Cohesión de los directorios

Se emplea una tabla de contadores por directorio, no por bloque. Comienza en el directorio raíz y desciende de forma recursiva al nivel para inspeccionar cada uno de los directorios del sistema de ficheros. Finalmente se obtiene una lista indexada por el número de i-nodo, diciéndole cuantos directorios tiene cada archivo, tras esto podemos comparar la cantidad de enlaces duros, en un sistema consistente de ficheros ambas cantidades deben coincidir.

TIPO TEST

+

EJERCICIOS



WUOLAH

Todos los planes de suscripción incluyen descargas sin publicidad con coins

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

pierdo
espacio

T1

EJERCICIOS

Teoría



Necesito
concentración

ali ali ooooh
esto con 1 coin me
lo quito yo...

wuolah
XXXXXX

1. ¿Cuál de las siguientes afirmaciones es VERDADERA?
 - a) En un sistema monotarea, el procesador se mantiene ocioso mientras se realizan operaciones de E/S.
 - b) Un sistema operativo multitarea se basa, entre otras cosas, en la concurrencia real entre el procesador y las operaciones de E/S realizadas por los controladores de los periféricos.
 - c) En los sistemas de tiempo compartido, se divide el tiempo de procesamiento entre los diferentes procesos que se encuentren listos para ejecutar, asignando pequeñas porciones a cada uno de ellos.
 - d) Todas las afirmaciones son VERDADERAS.

2. Las llamadas al sistema [system calls] sirven de interfaz entre:
 - a) Los programas de usuario y el sistema operativo.
 - b) El sistema operativo y las estructuras de datos internas.
 - c) Las aplicaciones y el intérprete de órdenes.
 - d) El intérprete de órdenes y los programas del sistema.

3. ¿El hardware puede activar directamente al sistema operativo?
 - a) No, todos los eventos del hardware deben pasar antes por la jerarquía de memorias.
 - b) No, el sistema operativo sólo se activa mediante software.
 - c) Sí, por ejemplo mediante una interrupción.
 - d) No, sólo se activa mediante llamadas al sistema.

4. El paso de la monoprogramación a la multiprogramación tuvo ciertas implicaciones, ¿cuál de las siguientes es FALSA?
 - a) Implica una gestión de memoria más compleja.
 - b) Permite que varios dispositivos se estén utilizando en paralelo.
 - c) El sistema operativo asume más funciones en general, eso implica que tendrá que ejecutarse más veces lo que conlleva que el porcentaje de uso de los recursos sea inferior (peor) que en la monoprogramación.
 - d) Todas las afirmaciones son FALSAS.

5. ¿Cuál de las siguientes afirmaciones es correcta?
 - a) Todas las afirmaciones son correctas.
 - b) Los sistemas operativos con estructura monolítica son más lentos que los sistemas con estructura de micronúcleo.
 - c) Los sistemas operativos con estructura de micronúcleo puro se ejecutan en un único espacio de direcciones.
 - d) Los sistemas operativos con estructura monolítica se ejecutan en un único espacio de direcciones.

WUOLAH: antja00

20

WUOLAH

6. Las llamadas al sistema:
- a) Son servicios del sistema operativo que no pueden ser invocados por los procesos de usuario, ya que deben ejecutarse en modo privilegiado.
 - b) Son órdenes del intérprete de órdenes.
 - c) Son servicios proporcionados por el intérprete de órdenes.
 - d) Todas las afirmaciones son FALSAS.
7. ¿De qué manera sabe el sistema operativo que un proceso termina su ejecución?
- a) Porque el proceso invoca una llamada al sistema específica para finalizar.
 - b) Todas las afirmaciones son FALSAS.
 - c) Porque transcurre un tiempo determinado sin que el proceso ejecute instrucciones.
 - d) Porque el contador de programa del proceso alcanza la última instrucción de productivas de su código máquina.
8. ¿Cuál de las siguientes afirmaciones es correcta?
- a) Todas las afirmaciones son correctas.
 - b) Los programas de usuario pueden realizar el acceso a los dispositivos de manera directa, sin la intervención del sistema operativo.
 - c) La gestión de procesos se encarga de ejecutar los programas del usuario.
 - d) La gestión de memoria se encarga de asignar y liberar memoria principal y memoria secundaria.
9. ¿Cuál de las siguientes afirmaciones sobre el núcleo del sistema operativo es FALSA?
- a) Contiene las rutinas de atención a las interrupciones
 - b) Se ejecuta en modo supervisor
 - c) Se carga en la memoria principal durante el arranque del sistema
 - d) Cuando hay sobrecarga puede estar en estado suspendido.
10. Llega una interrupción procedente de un dispositivo de E/S mientras un proceso de usuario se está ejecutando. ¿Qué suele ocurrir justo a continuación?
- a) Se atiende la interrupción de forma inmediata, en el modo de operación en el que se encontraba el procesador en el momento de ocurrir la operación.
 - b) Se ignora la interrupción, ya que el sistema está ejecutando código en modo usuario y si se atiende aquella, podría haber problemas de seguridad.
 - c) Se conmuta a modo usuario para que el proceso dialogue con la E/S de acuerdo con sus necesidades.
 - d) Se conmuta a modo supervisor y se ejecuta la rutina de servicio de interrupción correspondiente.

¿?

11. Una característica de las instrucciones máquina privilegiadas es que:
- a) Pueden comprometer la seguridad del sistema.
 - b) Sólo pueden ser ejecutadas por el administrador del sistema.
 - c) Permiten a los usuarios normales obtener privilegios adicionales.
 - d) Son las únicas que pueden ejecutarse en modo supervisor (sistema).
12. ¿Cuál de las siguientes afirmaciones es VERDADERA?
- a) El sistema de tiempo real es un sistema con tiempos de respuesta óptimas.
 - b) Todas las afirmaciones son FALSAS.
 - c) Los sistemas de tiempo compartido son multiprogramados en los que se reparte el uso de la CPU entre los procesos existentes a intervalos regulares de tiempo.
 - d) Los sistemas multiprogramados son sistemas de tiempo compartido que permiten conmutar los trabajos existentes en el sistema.
13. En el diseño de los sistemas operativos, ¿qué conseguimos gracias a la independencia del dispositivo?
- a) El sistema operativo transfiere los datos desde o hacia la E/S de manera independiente a las interrupciones de los dispositivos
 - b) Que la CPU y la E/S puedan realizar sus tareas en paralelo.
 - c) Más rapidez en las transferencias con los dispositivos
 - d) Una interfaz de programación uniforme con los dispositivos.
14. Si estando en ejecución una tarea de usuario, se produce una interrupción hardware, se ejecutará una rutina del SO. Al terminar esta:
- a) Se volverá siempre a la tarea interrumpida.
 - b) La tarea interrumpida puede haber pasado al estado bloqueado en función de qué interrupción se haya producido.
 - c) Mientras se está ejecutando una tarea de usuario no pueden producirse interrupciones, sólo cuando se están ejecutando tareas del SO.
 - d) Todas las respuestas anteriores son falsas.
15. ¿Cuál de las siguientes tareas NO corresponde al sistema operativo?
- a) Garantizar la protección de las zonas de memoria utilizadas por el sistema operativo frente a los procesos de usuario.
 - b) Garantizar la protección de la memoria de un proceso de usuario frente a otro proceso de usuario.
 - c) Ofrecer llamadas al sistema a través de las cuales un proceso de usuario pueda ejecutar instrucciones privilegiadas (bajo estricto control).
 - d) Impedir que un programa de usuario contenga instrucciones privilegiadas.

¿?

¿?

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato
→ Planes pro: más coins

pierdo
espacio



Necesito
concentración

ali ali ooooh
esto con 1 coin me
lo quito yo...

wuolah

16. Se dice que un sistema operativo es multiprogramado cuando...
- a) Ha sido escrito por múltiples programadores.
 - b) Está constituido por múltiples programas.
 - c) Además del núcleo del sistema operativo sólo puede haber en memoria un proceso de usuario.
 - d) Varios procesos de usuario pueden evolucionar concurrentemente en el sistema.
17. Una llamada al sistema es:
- a) Una orden del intérprete de órdenes.
 - b) Un programa de sistema.
 - c) Una interrupción hardware
 - d) Un servicio del sistema operativo
18. ¿Cuáles de las siguientes afirmaciones son FALSAS?
- a) Las instrucciones de E/S se pueden ejecutar en modo supervisor y en modo usuario.
 - b) La multiprogramación permite aprovechar las ráfagas de espera (de la E/S) de un trabajo para ejecutar las ráfagas de CPU de otro trabajo.
 - c) El SO se ocupa, solamente, de la gestión de los procesos, la gestión de ficheros y de la gestión de memoria. De la protección de ficheros y de gestionar la E/S se encargan los programas de usuario.
 - d) El intérprete de órdenes es la interfaz primaria entre el usuario y el SO. Se implementa como un programa que lee de su entrada estándar una orden introducida por un usuario, la analiza y la ejecuta.
19. Sobre las llamadas al sistema:
- a) Se ejecutan en modo usuario.
 - b) Una llamada al sistema es una orden del intérprete de órdenes.
 - c) Son programas de sistema.
 - d) Las órdenes como ls o cd son programas que incluyen varias llamadas al sistema.
20. ¿Cuál de las siguientes afirmaciones es correcta?
- a) El sistema operativo coordina los componentes de la máquina, optimizando su rendimiento.
 - b) Todas las afirmaciones son correctas.
 - c) El sistema operativo simplifica el manejo del hardware a los programas que se ejecuten en la máquina.
 - d) El sistema operativo asigna recursos a los programas que se ejecuten en la máquina.
21. ¿Qué se entiende por "sistema operativo multiprogramado"?
- a) Es aquel que nunca consume su tiempo de CPU.
 - b) El que reparte tiempo de CPU cuando un proceso realiza una E/S.
 - c) El que puede ser utilizado por varios usuarios a la vez
 - d) El que utiliza swapping para ejecutar varios programas a la vez

WUOLAH: antja00

23

22. ¿Cuál es el objetivo principal que persigue la multiprogramación?

- a) Mejorar el rendimiento del sistema en términos de espacio de uso.
- b) Mejorar el rendimiento del sistema en términos de tiempo de uso.
- c) Mejorar la seguridad del sistema
- d) Mejorar la estabilidad del sistema

¿?

23. El intérprete de órdenes:

- a) Es un programa del sistema y constituye la única forma de solicitar servicios del SO.
- b) Ejecuta las órdenes indicadas por el usuario y para ello hace uso a su vez de las llamadas al sistema.
- c) Al ser un programa del sistema, no necesita realizar llamadas al sistema.
- d) Siempre forma parte del núcleo del sistema operativo.

24. El intérprete de órdenes o *shell*, en sistemas tales como Linux o Mac:

- a) No realiza llamadas al sistema porque es parte del sistema operativo.
- b) Se ejecuta en modo supervisor o en modo kernel.
- c) Se le solicitan operaciones mediante interrupciones software.
- d) Todas las afirmaciones son FALSAS.

25. Un sistema operativo ofrece una interfaz de usuario (mandatos del sistema operativo) y una interfaz de programación (conjunto de llamadas al sistema). ¿Cuál de las siguientes afirmaciones es cierta?

- a) Sólo la interfaz de usuario puede ser sustituida por otra sin que esto afecte al resto del sistema.
- b) Sólo la interfaz de programación puede modificarse sin que esto afecte al resto del sistema.
- c) Tanto una como otra pueden modificarse o sustituirse ya que no son características del sistema operativo.
- d) Ni una ni otra pueden modificarse o sustituirse por otros ya que son características del sistema operativo.

¿?

26. ¿Cuál de los siguientes es un objetivo esencial del sistema operativo?

- a) Aislar al programador de la complejidad del hardware.
- b) Optimizar el código de los programas para mejorar su rendimiento.
- c) Interpretar los programas de usuario.
- d) Permitir al programador el control absoluto del sistema.

¿?

27. Mientras el núcleo del sistema operativo está ejecutando código que atiende una llamada al sistema, llama al sistema una interrupción de un dispositivo *hardware*. El núcleo atiende la interrupción. ¿En qué modo de ejecución se atiende la interrupción?

- a) Algunas veces en modo núcleo y otras en modo usuario.
- b) En modo núcleo, ya que se va a atender una interrupción.
- c) En modo usuario, ya que se estaba atendiendo una llamada al sistema.
- d) En el modo que designa el dispositivo que interrumpió.

28. ¿Cuál de las siguientes afirmaciones es correcta?

- a) El reparto de los recursos de la máquina entre los procesos que coexisten en el ordenador es una tarea del sistema operativo
- b) La garantía de protección entre usuarios de un sistema es una tarea del sistema operativo
- c) **Todas las afirmaciones son correctas.**
- d) El sistema operativo se considera una máquina extendida porque además de facilitar el uso del hardware también incrementa los servicios.

29. El nivel de multiprogramación de un sistema indica:

- a) Cuántos programas diferentes se están ejecutando en el sistema en ese momento.
- b) Cuántos procesos están en estado de ejecución en ese momento.
- c) **Cuántos procesos están cargados en memoria principal, aunque sea parcialmente, en ese momento.**
- d) Cuántos usuarios diferentes puede haber en el sistema como máximo en un momento dado.

¿?

30. ¿Qué diferencias hay entre una excepción y una interrupción?

- a) Una excepción es una interrupción software mientras que una interrupción es siempre hardware.
- b) Una excepción la provoca el usuario intencionadamente en su programa para pedirle algo al SO mientras que la interrupción se provoca sin intervención del usuario.
- c) **Una excepción es una interrupción que provoca la CPU cuando no puede ejecutar una instrucción por la causa que sea, mientras que una interrupción la provocan los dispositivos o bien los programas.**
- d) Todas las afirmaciones anteriores son falsas.

?

31. Las llamadas al sistema:

- a) **Son servicios del sistema operativo que no pueden ser invocados por los procesos de usuario, ya que deben ejecutarse en modo privilegiado.**
- b) Son órdenes del intérprete de mandatos.
- c) Son rutinas del núcleo del sistema operativo.
- d) Ninguna de las anteriores es cierta.

32. El principal mecanismo en el que se apoyan los sistemas operativos multiprogramados son:

- a) **Las interrupciones**
- b) Todas las afirmaciones son FALSAS
- c) Las llamadas al sistema
- d) **Los lenguajes de programación de alto nivel**

?