1) I FALSO.

Los enlaces simbólicos consumen más espacio en disco que los enlaces duros; los primeros ocupan bloques de datos y los segundos entradas de directorios.

Se puede minimizar el espacio consumido por el enlace simbólico en los casos donde la ruta del nombre del elemento compartido es corta y el sistema almacene los atributos del archivo en nodos-i.

Las ventajas de los enlaces simbólicos consisten en:

* Enlazar cualquier archivo de la estructura de directorio, tanto de la máquina local como de máquinas remotas.
* Borrar un enlace no afecta al elemento compartido.
* Si el usuario propietario borra el elemento compartido, el espacio liberado pasa a estar disponible para él.

1) II VERDADERO.

En entornos de tiempo compartido varios usuarios pueden estar ejecutando simultáneamente el mismo programa; si el programa no se puede modificar, es decir, posee código reentrante, se mantiene cargada en memoria principal una única copia de las páginas de código de dicho programa. Cada instancia tendrá asociada un proceso y en memoria principal una sola copia de las páginas de código del programa que son compartidas en modo sólo lectura.

III VERDADERO

El formateo de bajo nivel es realizado por el fabricante del disco y consiste en dividir cada superficie de los platos en pistas y cada pista en sectores

Un sector es una estructura de datos que principalmente consta de una cabecera, un área de datos y un código de corrección de errores.

La capacidad del disco formateado a bajo nivel es inferior por:

* La separación física entre pistas contiguas
* Reserva de espacio para paliar posibles sectores defectuosos.
* La cabecera y el código de corrección de errores de un sector consumen espacio.

Después de un formateo a bajo nivel la capacidad del disco se reduce aproximadamente un 20%.

IV FALSO

La técnica de gestión de la memoria que no considera la existencia de memoria virtual, para poder ejecutar un proceso su

espacio de direcciones lógicas debe estar cargado por completo en la memoria principal. Contrariamente, la técnica de gestión de memoria virtual consiste en tener cargada en memoria principal sólo aquellas partes (páginas o segmentos) del espacio de direcciones lógicas de un proceso que se van necesitando durante su ejecución.

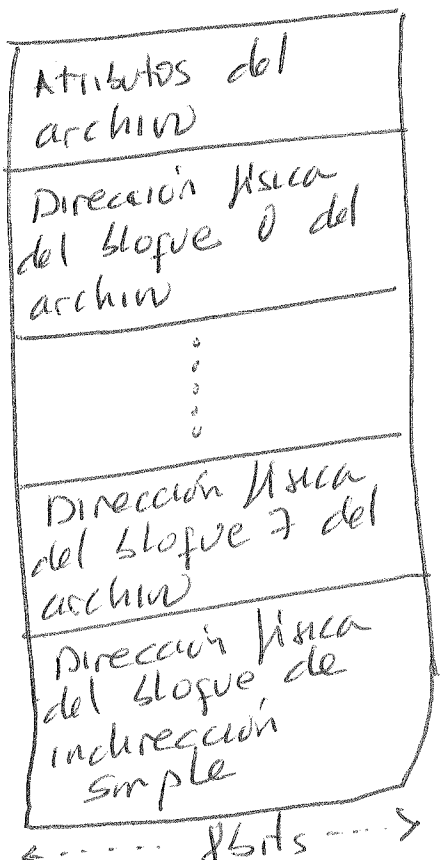
Así pues, se puede aumentar la cantidad de procesos cargados en memoria principal hasta un cierto límite (grado de multiprogramación) aumentando su eficiencia.

SO_PED2_Arenas_López_Miguel.pdf LINUX

2) En el método de asignación indexada almacenamos en un nodo-i los atributos de un archivo y las direcciones físicas de los ocho primeros bloques del archivo; además almacenamos la dirección física de un bloque de indirección simple, que almacena, a su vez, direcciones físicas de bloques del mismo.

Con esta estructura de nodo-i el número máximo de bloques que puede tener un archivo es de 8 bloques más los bloques que se puedan almacenar en el bloque de indirección simple.

Nodo índice



El área de datos consta de 256 bloques = 2^8 bloques.

El número de bits de una dirección física de un bloque se calcula resolviendo la desigualdad

$$\min_n \{ N_T \leq 2^n \}$$

como $N_T = 2^8$ entonces $n = 8$ bit

Ahora calculamos el número de direcciones físicas N_D que se puede almacenar en un

Bloque de indirección simple

$$N_D = \text{floor}\left(\frac{16\ \text{bytes}}{8\ \text{bits}}\right) = \text{floor}\left(\frac{16 * 8\ \text{bits}}{8\ \text{bits}}\right) =$$

$= 16$ direcciones únicas.

Con esta estructura de nodos-i el número máximo de bloques que puede tener un archivo es de $8 + 16 = 24$ bloques; como el tamaño de bloque es de 16 bytes, el tamaño máximo de un archivo en bytes es de 24 bloques * 16 bytes = 384 bytes

3) Para determinar la dirección de la última palabra que se lee en la memoria principal, en primer lugar pasamos la dirección física inicial de hexadecimal a decimal

$DIR_0 = AA00 = 1010\,1010\,0000\,0000_2 =$

$= 1*2^{15} + 1*2^{13} + 1*2^{11} + 1*2^9 =$

$= 43520_{10}$

El número de bytes a leer es de 1000 y una palabra tiene 16 bits = = 2 bytes, es decir, tenemos que leer 500 palabras empezando desde la dirección inicial; La dirección de la última palabra que se lee en la memoria principal es $43520 + 500 - 1 = 44019_{10}$ que pasando a hexadecimal es

ABF3

SO_PED2_Arenas_López_Miguel.pdf NO/AS

4) La paginación simple consiste en dividir la memoria principal en bloques del mismo tamaño $S_p = 4KiB$; como la capacidad de la memoria de este computador es de 200 KiB, el número de marcos de páginas o páginas físicas es N_{MP}

$$N_{MP} = floor\left(\frac{C_{MP}}{S_p}\right) = \frac{200\ KiB}{4\ KiB} = 50 \text{ marcos}$$

siendo C_{MP} = capacidad de la memoria.

Sea C_x el tamaño del espacio de direcciones lógicas del proceso A, $C_x = 50\ KiB$, y sea N_p el número de páginas de tamaño S_p en que se descompone el espacio del proceso

$$N_p = ceil\left(\frac{C_x}{S_p}\right) = ceil\left(\frac{50\ KiB}{4\ KiB}\right) =$$

$$= ceil(12,5) = 13 \text{ páginas}$$

La tabla de páginas del proceso A tiene 13 entradas; y su contenido después

de haberse cargado el proceso en memoria es

	Otros campos	Marco j
i = 0		1
i = 1		3
i = 2		23
i = 3		24
i = 4		26
i = 5		27
i = 6		28
i = 7		29
i = 8		30
i = 9		32
i = 10		34
i = 11		36
i = 12		38

(b) El proceso A adolece de fragmentación interna, ya que la última página solo está ocupada en la mitad. El espacio desaprovechado es de

$$\frac{4KB}{2} = 2KB = 2 * 2^{10} * 2^{3} = 2^{14} \text{ bits} = 16384 \text{ bits}.$$

(c) El estado del mapa de bits despues de haber cargado en la memoria el proceso A es:

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0

1 0 0 0 0 1 0 0 0 0