SISTEMAS OPERATIVOS

Índice:

Ejercicio 1 – Memoria Virtual: Paginación

Tenemos un ordenador que utiliza memoria virtual con paginación. Las direcciones virtuales son de 32 bits y el tamaño de página es de 4KB. La memoria física (RAM) del ordenador es de 16MB (cada posición es de 1 Byte). Contesta:

a) ¿Cuántos bits son necesarios para direccionar un marco de página?

b) ¿En cuántos marcos de página se divide la memoria física?

c) ¿Cuántos bits de la dirección virtual se utilizan para identificar la página?

d) ¿Cuántas entradas tiene la tabla de página (suponiendo que es de un nivel)?

e) Sin contar los bits de permisos, de presente/ausente, M y R, ¿cuántos bits tiene como mínimo una entrada en la tabla de página (suponiendo que es de un nivel)?

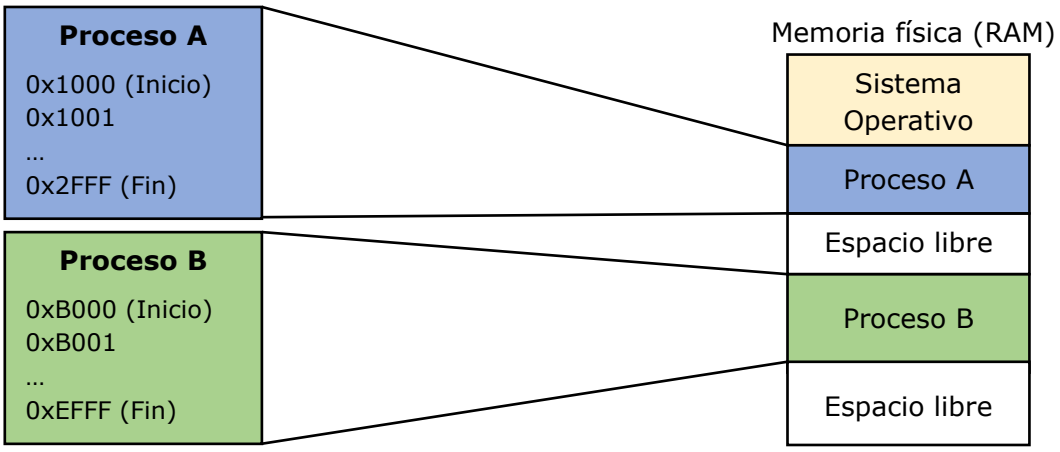
f) ¿Cuál es el tamaño de la tabla de la página (suponiendo que es de un nivel)?

g) Supón ahora que tenemos una tabla de página de 2 niveles y que las tablas de 2º nivel tienen 2 entradas. ¿cuántas entradas tiene la tabla de primer nivel?

Consideraciones previas

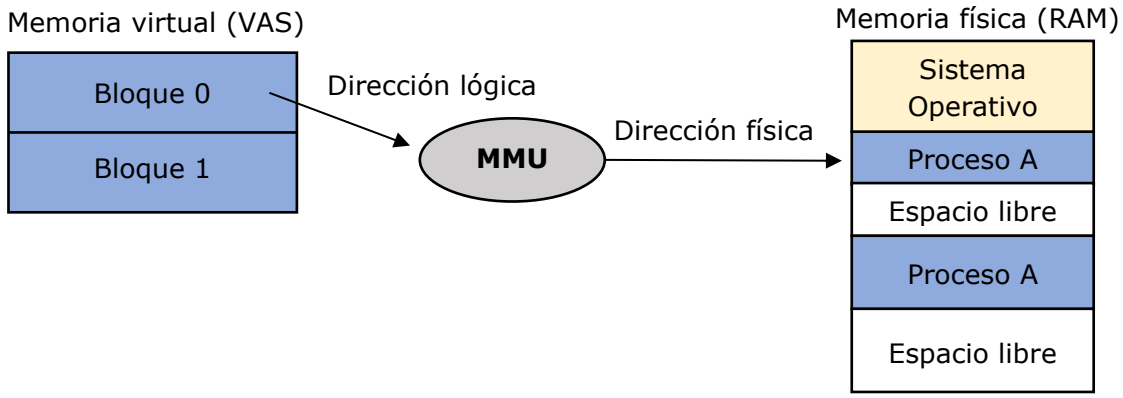
Antes de empezar, se van a explicar algunos conceptos básicos sobre la gestión de memoria. La asignación de memoria puede ser contigua y no contigua.

En la asignación de **memoria contigua**, el sistema operativo asigna a cada proceso un único bloque de memoria y todo el espacio lógico del proceso (código, datos, pila, etc.) se almacena en direcciones físicas consecutivas.



En la asignación de **memoria no contigua**, el sistema operativo divide cada proceso en varios bloques que pueden ser almacenados en zonas distintas de la memoria principal, sin necesidad de ser adyacentes. Es más difícil para un sistema operativo controlar la asignación de almacenamiento, pero permite aprovechar los huecos de memoria que van quedando al liberar procesos. Imagina que en el esquema anterior surge un tercer proceso que no cabe de forma completa en ninguno de los 2 espacios libres que hay, pero si se dividiera en 2 sí cabría una parte en un hueco y la otra parte en el otro hueco.

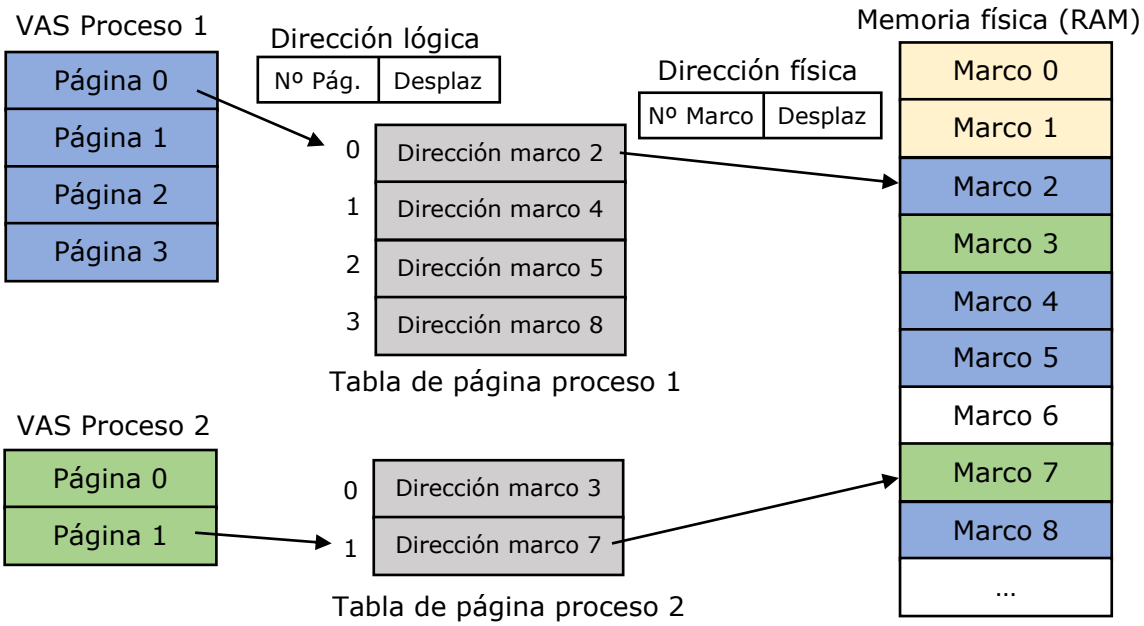
Para conseguirlo, cada proceso utiliza un **espacio de direcciones virtuales** o VAS (*Virtual Address Space*), que es un conjunto de direcciones virtuales o lógicas. Este espacio es único para cada proceso y privado (otro proceso no puede verlo a no ser que lo comparta). Por tanto, cada proceso, en lugar de acceder a la memoria física, accede a su memoria virtual (VAS) y la MMU (*Memory Managment Unit*), que es un dispositivo *hardware*, traduce la dirección virtual en la dirección física real.



Para realizar la asignación de memoria no contigua existen varias técnicas como la paginación, la segmentación o la segmentación paginada.

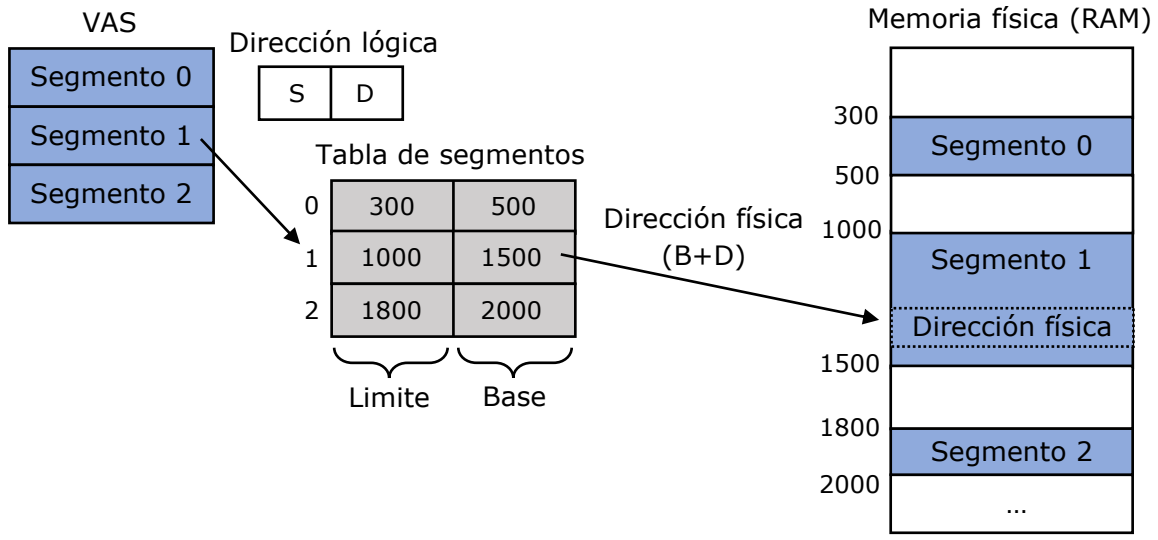
Con la técnica de **paginación**, la memoria virtual se divide en bloques llamados "páginas" y la memoria física se divide en bloques llamados "marcos de página" (*frames*). Todos los bloques (marcos y páginas) tienen el mismo tamaño.

Para transformar una dirección lógica a la dirección física correspondiente, se necesita un documento llamado **tabla de página**, que es utilizado por la MMU. Cada entrada de la tabla de página contiene el número de marco que corresponde con la página a la que hace referencia, además de otros bits de control como el estado o bit de validez, que indica si la página se encuentra cargada en la memoria física (presente) o hay que cargarla del disco (ausente), bits de permisos, M (modificado), R (referenciado), etc.



La dirección lógica consta de 2 partes: número de página y desplazamiento dentro de la página, mientras que la dirección física consta de número de marco y desplazamiento. El desplazamiento sirve para referenciar los datos que una página contiene, por tanto, su tamaño es el mismo que el tamaño de la página y del marco.

Con la técnica de **segmentación**, el espacio de direcciones se divide en segmentos, cada uno de los cuales corresponde a una rutina, un programa o un conjunto de datos, es decir, cada segmento corresponde con un subespacio independiente (código, datos, pila, etc.) y puede tener un tamaño distinto.



La dirección lógica consta de 2 partes: número de segmento (S) y desplazamiento (D), mientras que la dirección física se calcula sumando la base más el desplazamiento (B+D), siendo el segmento que ocupa el área que va desde la base hasta la base más el límite (B+L).

Solución

a) ¿Cuántos bits son necesarios para direccionar un marco de página?

La memoria física tiene una capacidad de 16MB = 2^{24} Bytes, por lo que las direcciones físicas tienen un tamaño de 24 bits.

Por otra parte, el tamaño de página es de 4KB = 2^{12} Bytes, por lo que se emplean 12 bits para el desplazamiento dentro de la página.

Por tanto, para direccionar un marco de página se utilizan 24-12 = **12 bits.**

Dirección física (24 bits)

Nº de marco (12 bits)	Desplazamiento (12 bits)

b) ¿En cuántos marcos de página se divide la memoria física?

Como se usan 12 bits para direccionar los marcos, se divide en 2^{12} = **4096 marcos.**

c) ¿Cuántos bits de la dirección virtual se utilizan para identificar la página?

La dirección virtual es de 32 bits y como se utilizan 12 bits para el desplazamiento, obtenemos que se usan **20 bits** (32-12) para identificar la página.

Dirección virtual (32 bits)

Nº de página virtual (20 bits)	Desplazamiento (12 bits)

d) Supón que tenemos una tabla de página de un solo nivel, ¿cuántas entradas tiene la tabla de la página?

Suponiendo que la cantidad de entradas que tiene la tabla de página coinciden con la cantidad de páginas de la memoria virtual, disponemos de 2^{20} = **1048576 entradas**.

e) Sin contar los bits de permisos, de presente/ausente, M y R, ¿cuántos bits tiene como mínimo una entrada en la tabla de página (suponiendo que es de un nivel)?

12 bits, ya que cada entrada contiene como mínimo la dirección de un marco.

f) ¿Cuál es el tamaño de la tabla de la página (suponiendo que es de un nivel)?

Si hay 1048576 entradas en la tabla de página y cada una contiene 12 bits, el tamaño de la tabla de página es de 1048576*12 = 12582912 bits = **1,5MB**.

g) Supón ahora que tenemos una tabla de página de 2 niveles y que las tablas de 2º nivel tienen 2 entradas. ¿cuántas entradas tiene la tabla de primer nivel?

Que las tablas de 2º nivel tengan 2 entradas, quiere decir que necesitamos 1 bit para direccionarlas. De los 20 bits que se usaban para referenciar el número de página virtual en la tabla de página, ahora se necesitan 20-1=19 bits.

Dirección virtual (32 bits)

Tabla de nivel 1 (19 bits)	Tabla de nivel 2 (1 bit)	Desplazamiento (12 bits)

Por tanto, la tabla de primer nivel tiene 2^{19} = **524288 entradas**

Ejercicio 2 – Memoria Virtual: Paginación 2

Un ordenador utiliza páginas de 4KB y su tabla de página en un determinado instante es la siguiente (la primera columna hace referencia al marco y la segunda al bit de validez):

0	010	1
1	001	1
2	110	1
3	000	0
4	100	1
5	011	1
6	000	0
7	000	0
8	000	0
9	101	1

Bit de validez

Se pide lo siguiente:

a) ¿Cuál es el tamaño de los marcos de página?

b) ¿Cuál es el tamaño de las direcciones físicas?

c) ¿Cuál es el tamaño de las direcciones virtuales?

d) ¿Cuál es la dirección física que se genera para la dirección lógica 2989?

e) ¿Cuántos marcos de página tiene?

f) ¿Cuántas páginas virtuales tiene?

g) ¿Se están utilizando en ese instante todos los marcos de página?

h) Dibuja un esquema que muestre la distribución de la memoria virtual y la física

Solución

a) ¿Cuál es el tamaño de los marcos de página?

En paginación, el tamaño de página (bloques de la memoria virtual) y el tamaño del marco (bloques de la memoria física) tienen que ser iguales, por tanto, el tamaño de los marcos es de **4KB.**

b) ¿Cuál es el tamaño de las direcciones físicas?

Se observa en la tabla de página que cada entrada emplea 3 bits para referenciar al número de marco.

Por otra parte, el tamaño de página es de 4KB = 2^{12} Bytes, por lo que se emplean 12 bits para el desplazamiento dentro de la página.

Por tanto, las direcciones físicas tienen un tamaño de 3 + 12 = **15 bits.**

Dirección física (15 bits)

Nº de marco (3 bits)	Desplazamiento (12 bits)

c) ¿Cuál es el tamaño de las direcciones virtuales?

Las direcciones virtuales deben de ser capaces de referenciar todas las entradas de la tabla de páginas. Como la tabla tiene 10 entradas y el número de entradas debe de ser múltiplo de 2, suponemos que tiene capacidad para 16 (2^4) entradas. Es decir, que se usan 4 bits para referenciar una página virtual.

Por tanto, las direcciones virtuales tienen un tamaño de 4 + 12 = **16 bits.**

Dirección virtual (16 bits)

Nº de página virtual (4 bits)	Desplazamiento (12 bits)

d) ¿Cuál es la dirección física que se genera para la dirección lógica 2989?

Primero pasamos 2989 a binario → $2989_{(10}$ = $101110101101_{(2}$

Y obtenemos la dirección virtual completa:

Dirección virtual	
0000	101110101101

El número de página virtual corresponde con la entrada "0000" de la tabla de páginas y esa entrada contiene los siguientes 2 valores:

- Número de marco: 010
- Bit de validez: 1 (está cargado en la RAM y se produce un "acierto")

Por tanto, la dirección física es:

Dirección física	
010	101110101101

Que pasándolo a decimal da **11181.**

e) ¿Cuántos marcos de página tiene?

Si se utilizan 3 bits para indicar el número de marco, la RAM tiene 2^3 = **8 marcos**

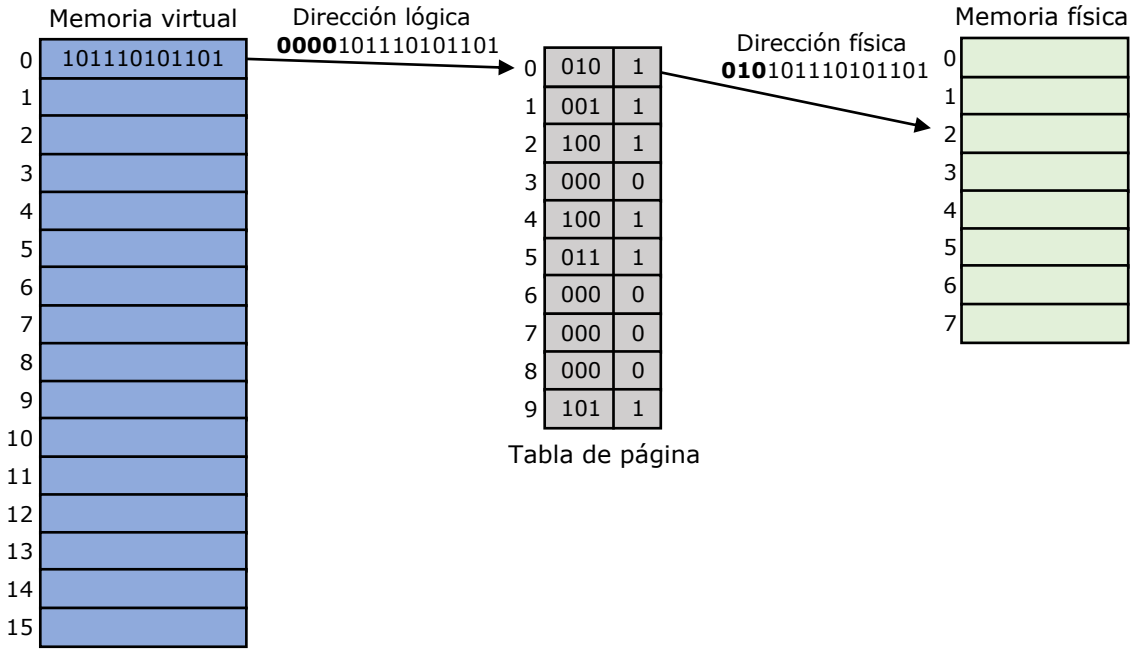
f) ¿Cuántas páginas virtuales tiene?

La memoria virtual está compuesta de 2^4 = **16 páginas**

g) ¿Se están utilizando en ese instante todos los marcos de página?

Se comprueban si todos los valores posibles (de 000 a 111) tienen el bit de validez a "1" y se observa que se están utilizando todos los marcos menos el "000" y el "111", es decir, todos menos el primer y último marco.

h) Dibuja un esquema que muestre la distribución de la memoria virtual y la física



Ejercicio 3 – Memoria Virtual: Paginación 3

En un sistema en el que se administra la memoria por paginación se dispone de 256 KB de memoria física, siendo el tamaño de página empleado 4 KB y las direcciones virtuales de 32 bits.

a) Razona si las direcciones de memoria virtuales 0xabc10008 y 0xabc100aa hacen referencia al mismo marco.

b) ¿Y las direcciones 0xabc1fa00 y 0xabc2fa08?

c) Indica el tamaño máximo de la tabla de páginas de un proceso suponiendo que cada entrada ocupa 8 bits.

d) ¿De cuántos marcos de página dispone el sistema?

Solución

a) Razona si las direcciones de memoria virtuales 0xabc10008 y 0xabc100aa hacen referencia al mismo marco.

El tamaño de página es de 4KB = 2^{12} Bytes, por lo que se emplean 12 bits para el desplazamiento dentro de la página.

La dirección virtual es de 32 bits y como se utilizan 12 bits para el desplazamiento, obtenemos que se usan 20 bits (32-12) para identificar la página.

Dirección virtual (32 bits)

Nº de página (20 bits)	Desplazamiento (12 bits)

Ahora pasamos cada dirección virtual a binario y vemos si coincide el número de página virtual. Se ponen en negrita los bits que indican el número de página.

$ABC10008_{(16}$ = **1010 1011 1100 0001 0000** 0000 0000 1000

$ABC100AA_{(16}$ = **1010 1011 1100 0001 0000** 0000 1010 1010

Como se puede observar, apuntan a la misma dirección de la tabla de página, por tanto, hacen referencia al mismo marco.

b) ¿Y las direcciones 0xabc1fa00 y 0xabc2fa08?

Hacemos lo mismo:

$ABC1FA00_{(16}$ = **1010 1011 1100 0001 1111** 1010 0000 0000

$ABC2FA08_{(16}$ = **1010 1011 1100 0010 1111** 1010 0000 1000

Como se puede ver, no apuntan a la misma dirección de la tabla de página, por tanto, hacen referencia a distintos marcos.

c) Indica el tamaño máximo de la tabla de páginas de un proceso suponiendo que cada entrada ocupa 8 bits.

Si se usan 20 bits para identificar la página, la tabla de páginas dispone de 2^{20} = 1048576 entradas como máximo. Si cada una ocupa 8 bits, el tamaño total de la tabla es 1048576*8 = 8388608 bits → **1MB**.

d) ¿De cuántos marcos de página dispone el sistema?

La memoria física es de 256KB = 2^{18} Bytes, por lo que las direcciones físicas tienen un tamaño de 18 bits.

Como se utilizan 12 bits para el desplazamiento dentro de la página, para direccionar un marco se utilizan 18-12 = 6 bits.

Dirección física (18 bits)

Nº de marco (6 bits)	Desplazamiento (12 bits)

Si se utilizan 6 bits para indicar el número de marco, la RAM se divide en 2^6 = **64 marcos**

NOTA: En el apartado b) se menciona que cada entrada de la tabla de páginas ocupa 8 bits, pero en este apartado se ha comprobado tan solo se necesitan 6 bits para referenciar el número de marco. Esto es posible porque la tabla de páginas contiene, además del número de marco, otros bits adicionales como el bit de validación.

Ejercicio 4 – Memoria Virtual: Paginación 4

Supón un sistema de memoria virtual con paginación por demanda que tiene un tamaño de página de 512 palabras, una memoria virtual de 16 páginas numeradas del 0 al 15 y una memoria física de 4 marcos (*frames*) numerados de 0 a 3. El contenido actual de la memoria física es el siguiente:

0	Página 4 del proceso P
1	Página 9 del proceso P
2	Página 5 del proceso P
3	Página 1 del proceso P

a) Mostrar el contenido de la tabla de páginas.

b) Suponiendo un algoritmo de reemplazo de páginas con estrategia óptima, mostrar el contenido de la tabla de páginas, tras generar cada una de las siguientes direcciones lógicas: 1112, 1645, 2049, 622, 2776.

c) Indica las direcciones físicas equivalentes a las lógicas 1628, 851, 2700 y 2432.

d) ¿Qué pasa cuando se referencia la dirección lógica 1330?

Solución

a) Mostrar el contenido de la tabla de página.

Cantidad de marcos = 4 = 2^2 → 2 bits para direccionar los marcos.

La tabla de páginas tiene 16 entradas y mirando el contenido de la memoria física, obtenemos la siguiente tabla de página.

	Marco	Bit de validez
0	-	0
1	11 (3)	1
2	-	0
3	-	0
4	00 (0)	1
5	10 (2)	1
6	-	0
7	-	0
8	-	0
9	01 (1)	1
10	-	0
11	-	0
12	-	0
13	-	0
14	-	0
15	-	0

b) Suponiendo un algoritmo de reemplazo de páginas con estrategia óptima, mostrar el contenido de la tabla de páginas, tras generar cada una de las siguientes direcciones lógicas: 1112, 1645, 2049, 622, 2776.

Tamaño de página = 512 = 2^9 páginas → 9 bits para el desplazamiento

Cantidad de 16 páginas = 2^4 → 4 bits para direccionar las páginas.

Dirección lógica (13 bits)

Nº de página (4 bits)	Desplazamiento (9 bits)

Pasamos las direcciones lógicas a binario y ponemos en negrita el número de página:

1112 = **0010**001011000 → Página 2
1645 = **0011**001101101 → Página 3
2049 = **0100**000000001 → Página 4
662 = **0001**010010110 → Página 1
2776 = **0101**011011000 → Página 5

Representamos cómo quedaría la memoria tras estas referencias aplicando el algoritmo óptimo para el reemplazo de páginas:

Página →		Inicial	2	3	4	1	5
Marcos de página	0	4	4	4	4	4	4
	1	9	2	3	3	3	3
	2	5	5	5	5	5	5
	3	1	1	1	1	1	1
Fallo/Acierto →			F	F	A	A	A

Tras generar las direcciones lógicas anteriores, la tabla de página queda así:

	Marco	Bit de validez
0	-	0
1	11 (3)	1
2	01 (1)	0
3	01 (1)	1
4	00 (0)	1
5	10 (2)	1
6	-	0
7	-	0
8	-	0
9	01 (1)	0
10	-	0
11	-	0
12	-	0
13	-	0
14	-	0
15	-	0

c) Indica las direcciones físicas equivalentes a las lógicas 1628, 851, 2700 y 2432.

Pasamos las direcciones lógicas a binario, vemos qué marco corresponde con qué página y transformamos a decimal:

1628 = **0011**001011100 → Página 3 → Marco 1 → **01**001011100 → 604
851 = **0001**101010011 → Página 1 → Marco 3 → **11**101010011 → 1875
2700 = **0101**010001100 → Página 5 → Marco 2 → **10**010001100 → 1164
2432 = **0100**110000000 → Página 4 → Marco 0 → **00**010001100 → 140

d) ¿Qué pasa cuando se referencia la dirección lógica 1330?

1628 = **0010**100110010 → Página 2 → Fallo de página ya que el bit de validez está a 0

Ejercicio 5 – Memoria Virtual: Paginación vs Segmentación

Un proceso ocupa 1KB de código, 4KB de pila y 5KB de constantes. ¿Cuánta memoria virtual ocupará el proceso en los siguientes casos?

a) Utilizando paginación pura con un tamaño de página de 4KB.

b) Utilizando segmentación pura.

c) ¿Cuál hace un uso más eficiente del espacio físico (RAM)?

Solución

a) Utilizando paginación pura con un tamaño de página de 4KB.

El código necesita 1 página → 4KB (Se desperdician 3KB por fragmentación interna)

La pila necesita 1 página → 4KB (No se produce fragmentación interna)

Las constantes necesitan 2 páginas → 8KB (Se desperdician 3KB)

El proceso ocupa 4 + 4 + 8 = **16KB**

b) Utilizando segmentación pura.

El código ocupa un segmento de 1KB, la pila un segmento de 4KB y las constantes ocupan un segmento de 5KB. En total, el proceso ocupa **10KB**

c) ¿Cuál hace un uso más eficiente del espacio físico (RAM)?

Aunque con la segmentación los procesos ocupan menos espacio en memoria debido a que se produce una menor fragmentación interna, en realidad es menos eficiente porque produce una mayor fragmentación externa. Explicación:

Cuando se libera un proceso usando paginación, se liberan marcos del mismo tamaño que podrán ser aprovechados al 100% por otro proceso nuevo (no hay fragmentación externa). Sin embargo, cuando se liberan procesos con la segmentación, se liberan trozos de memoria de distinto tamaño, por ejemplo, si se libera un segmento de 300KB y, posteriormente, queremos almacenar un segmento de 320KB, ese hueco no nos servirá y habrá que buscar otro más grande.

Ejercicio 6 – Memoria Virtual: ¿Se puede cargar el programa?

Un sistema posee una memoria física de 64KB dividido en marcos de páginas de tamaño 4KB. Un programa tiene un código de tamaño 32768 bytes, un conjunto de datos de 16386 bytes y una pila de 15870 bytes. ¿Se podrá cargar este programa en la memoria? Razonar si influye el tamaño de la página.

Solución

Si el tamaño de página es de 4KB la memoria total consta de: 64/4 = 16 marcos.

Para el código del programa se necesitan: 32768/4096 = 8 marcos.

Para los datos del programa se necesitan: 16386/4096 = 4,000048 → 5 marcos.

Para la pila del programa se necesitan: 15870/4096 = 3,8745 → 4 marcos

En total se necesitan 17 marcos para cargar el programa, por lo que es imposible ya que disponen de 16 marcos de página.

El tamaño de la página sí influye en el tamaño que ocupa el proceso en la RAM, por ejemplo, el tamaño del programa es 32768+16386+15870 = 65024 Bytes → 63,5 KB y se observa que es menor que el tamaño de memoria física (64KB), pero como se utilizan páginas de 4KB, requiere de 68KB. Si se usarán tamaños de página más pequeños, por ejemplo, de 512Bytes, el proceso ocuparía menos RAM debido a que se reduce la fragmentación interna, pero también daría lugar a tablas de página más grandes.

Ejercicio 7 – Memoria Virtual y Gestión de memoria: LRU y FIFO

Considera un sistema de paginado bajo demanda con tamaño de página de 128 direcciones, espacio de direccionamiento virtual de 1024 direcciones y tamaño de memoria física de 512 direcciones.

a) Determina el número de páginas virtuales y físicas (marcos) y el tamaño de las direcciones virtuales y físicas.

b) Teniendo en cuenta que la secuencia de referencia a memoria de un proceso es la siguiente: 425, 570,200, 198, 426, 353, 512, 427, 380, 202, 103, 514. Calcula el ritmo de paginación generado por dicho trozo de direcciones al usar los algoritmos de reemplazo de páginas LRU y FIFO.

Solución

a) Determina el número de páginas virtuales y físicas (marcos) y el tamaño de las direcciones virtuales y físicas.

Tamaño de página = 128 direcciones = 2^7 direcciones → **7 bits**

Dirección virtual = 1024 direcciones = 2^{10} direcciones → **10 bits**

Número de páginas virtuales = 10 - 7 = 3 bits → 2^3 = **8 páginas**

Dirección física = 512 = 2^9 direcciones → **9 bits**

Número de páginas físicas = 9 - 7 = 2 bits → 2^2 = **4 marcos de página**

Dirección virtual (10 bits)

Nº de marco (3 bits)	Despl. (7 bits)

Dirección física (9 bits)

Nº de marco (2 bits)	Despl. (7 bits)

b) Teniendo en cuenta que la secuencia de referencia a memoria de un proceso es la siguiente: 425, 570,200, 198, 426, 353, 512, 427, 380, 202, 103, 514. Calcula el ritmo de paginación generado por dicho trozo de direcciones al usar los algoritmos de reemplazo de páginas LRU y FIFO.

Primero se pasan los números a binario (se deja en negrita los 3 bits de la página virtual)

425 = **011**0101001 → 3
570 = **100**0111010 → 4
200 = **001**1001000 → 1
198 = **001**1000110 → 1
426 = **011**0101010 → 3
353 = **010**1100001 → 2
512 = **100**0000000 → 4
427 = **011**0101011 → 3
380 = **010**1111100 → 2
202 = **001**1001010 → 1
103 = **000**1100111 → 0
514 = **100**0000010 → 4

Y ahora se comprueba el comportamiento de los 2 algoritmos teniendo en cuenta lo siguiente:

El tamaño de memoria física es menor que el tamaño de memoria virtual (RAM + área de intercambio). Se dice que se ha producido un fallo cuando la CPU tiene que acceder al área de intercambio o *swap* (zona del disco secundario) en busca de una referencia y que se ha producido un acierto si la referencia se encuentra en la memoria física (RAM).

Los algoritmos de reemplazo de páginas se utilizan para decidir qué páginas pueden ser sacadas de la memoria RAM cuando se necesita cargar una página nueva y no hay marcos de página libres.

Algoritmo LRU (*Last Recently Used*): Sale el que más tiempo lleva sin usarse.

Página →		3	4	1	1	3	2	4	3	2	1	0	4
Marco de página (2^2 = 4 páginas)	0	3	3	3	3	3	3	3	3	3	3	3	4
	1		4	4	4	4	4	4	4	4	4	0	0
	2			1	1	1	1	1	1	1	1	1	1
	3						2	2	2	2	2	2	2
Fallo/Acierto →		F	F	F	A	A	F	A	A	A	A	F	F

Algoritmo FIFO (*First In, First Out*): Primero en entrar, primero en salir.

Página →		3	4	1	1	3	2	4	3	2	1	0	4
Marco de página (2^2 = 4 páginas)	0	3	3	3	3	3	3	3	3	3	3	0	0
	1		4	4	4	4	4	4	4	4	4	4	4
	2			1	1	1	1	1	1	1	1	1	1
	3						2	2	2	2	2	2	2
Fallo/Acierto →		F	F	F	A	A	F	A	A	A	A	F	A

Ejercicio 8 – Gestión de memoria: LRU y Óptimo

Considera un programa que genera una secuencia de referencias a direcciones virtuales que corresponden a la siguiente secuencia de referencias de páginas: 1, 2, 3, 4, 1, 2, 5, 6, 1, 3, 1, 2, 5

a) Muestra cómo son alojadas las páginas en memoria física para los algoritmos de reemplazo de páginas LRU y Óptimo. Inicialmente se dispone de 5 marcos vacíos.

b) ¿Cuál de los 2 algoritmos es mejor? ¿Se puede mejorar el resultado?

Solución

a) Muestra cómo son alojadas las páginas en memoria física para los algoritmos de reemplazo de páginas LRU y Óptimo. Inicialmente se dispone de 5 marcos vacíos.

Algoritmo LRU (*Last Recently Used*): Sale el que más tiempo lleva sin usarse.

Referencias de página →		1	2	3	4	1	2	5	6	1	3	1	2	5
Marcos de página	0	1	1	1	1	1	1	1	1	1	1	1	1	1
	1		2	2	2	2	2	2	2	2	2	2	2	2
	2			3	3	3	3	3	6	6	6	6	6	6
	3				4	4	4	4	4	4	3	3	3	3
	4							5	5	5	5	5	5	5
Fallo/Acierto →		F	F	F	F	A	A	F	F	A	F	A	A	A

Algoritmo Óptimo: Se sustituye la página que tardará más en volverse a utilizar. Hay que comprobar futuras referencias.

Referencias de página →		1	2	3	4	1	2	5	6	1	3	1	2	5
Marcos de página	0	1	1	1	1	1	1	1	1	1	1	1	1	1
	1		2	2	2	2	2	2	2	2	2	2	2	2
	2			3	3	3	3	3	3	3	3	3	3	3
	3				4	4	4	4	6	6	6	6	6	6
	4							5	5	5	5	5	5	5
Fallo/Acierto →		F	F	F	F	A	A	F	F	A	A	A	A	A

b) ¿Cuál de los 2 algoritmos es mejor? ¿Se puede mejorar el resultado?

El mejor algoritmo es el óptimo ya que el porcentaje de fallos que produce (6/13*100 = 46,15%) es menor que el LRU (7/13*100 = 53,84%).

No se puede mejorar el resultado, ya que el número de fallos que se producen (6) coincide con el número de páginas que se hacen referencia.

Ejercicio 9 – Gestión de memoria 2: LRU, Óptimo y FIFO

Un proceso puede disfrutar como máximo de 5 marcos de página y realiza las siguientes referencias a páginas: 9, 1, 3, 6, 4, 2, 3, 7, 0, 4, 6, 9, 2, 4, 1, 2, 3. El contenido inicial de la memoria está formado por las páginas: 1, 7, 3, 6, 9.

Realiza una tabla donde se muestre la evolución del contenido de la memoria asignada a este proceso y al número de fallos provocados por los siguientes algoritmos:

a) LRU

b) Óptimo

c) FIFO

Solución

a) LRU

Se sustituye la página que más tiempo lleva sin usarse.

Referencias		t=0	9	1	3	6	4	2	3	7	0	4	6	9	2	4	1	2	3
Marcos de página	M_0	1	1	1	1	1	1	1	1	7	7	7	7	7	2	2	2	2	2
	M_1	7	7	7	7	7	4	4	4	4	4	4	4	4	4	4	4	1	1
	M_2	3	3	3	3	3	3	3	3	3	3	3	3	9	9	9	9	9	3
	M_3	6	6	6	6	6	6	6	6	6	0	0	0	0	0	0	1	1	1
	M_4	9	9	9	9	9	9	2	2	2	2	2	6	6	6	6	6	6	6
Fallo/Acierto			A	A	A	A	F	F	A	F	F	A	F	F	F	A	F	A	F

Fallos = 9

b) Óptimo

Se miran referencias futuras y se sustituye la página que tardará más en volverse a utilizar. Si hay 2 o más páginas que no se van a referenciar, se sustituye una al azar.

Referencias		t=0	9	1	3	6	4	2	3	7	0	4	6	9	2	4	1	2	3
Marcos de página	M_0	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4	4	4	4
	M_1	7	7	7	7	7	7	7	7	7	0	0	0	0	0	0	1	1	1
	M_2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	M_3	6	6	6	6	6	6	6	6	6	6	6	6	9	9	9	9	9	9
	M_4	9	9	9	9	9	9	2	2	2	2	2	2	2	2	2	2	2	2
Fallo/Acierto			A	A	A	A	F	F	A	A	F	A	A	F	A	A	F	A	A

Fallos = 5

c) FIFO

Se reemplaza la página que llegó antes a memoria. Suponemos que la primera página que llegó fue la 1, luego la 7, después la 3, etc.

Referencias		t=0	9	1	3	6	4	2	3	7	0	4	6	9	2	4	1	2	3
Marcos de página	M_0	1	1	1	1	1	4	4	4	4	4	4	4	9	9	9	9	9	9
	M_1	7	7	7	7	7	7	2	2	2	2	2	2	2	2	4	4	4	4
	M_2	3	3	3	3	3	3	3	3	7	7	7	7	7	7	7	1	1	1
	M_3	6	6	6	6	6	6	6	6	6	0	0	0	0	0	0	0	2	2
	M_4	9	9	9	9	9	9	9	9	9	9	9	6	6	6	6	6	6	3
Fallo/Acierto			A	A	A	A	F	F	A	F	F	A	F	F	A	F	F	F	F

Fallos = 10

Ejercicio 10 – Gestión de procesos: FIFO, SJF y SRTF

Se están ejecutando los siguientes procesos y queremos aplicar un algoritmo de planificación de la CPU, pero nos falta decidirnos entre los 3 siguientes: Primero en entrar, primero en salir (FIFO), tiempo de ejecución más corto primero (SJF) y tiempo restante más corto primero (SRTF).

Proceso	Tiempo de llegada	Tiempo de ejecución
1	0	5
2	1	3
3	2	10
4	3	1
5	4	2

Obtenga los tiempos de espera y el tiempo de retorno de cada proceso, así como el tiempo promedio de espera y de retorno de cada algoritmo.

Solución

Durante los diagramas se utilizará la siguiente nomenclatura:

X → Proceso en ejecución
P → Proceso en espera/preparado

Primero en entrar, primero en salir o FIFO (*First In, First Out*)

Este algoritmo no es apropiativo (cuando un proceso toma el control de la CPU, ningún otro proceso se lo arrebatará) ni emplea prioridades. Así pues, los procesos se van ejecutando según su orden de llegada.

Dibujamos un diagrama que indique el estado de cada proceso en cada unidad de tiempo (u.t.).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1	X	X	X	X	X																
P2		P	P	P	P	X	X	X													
P3			P	P	P	P	P	P	X	X	X	X	X	X	X	X	X	X			
P4				P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	X		
P5					P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	X	X

Los tiempos de espera y de retorno para cada proceso son:

Proceso	Tiempo de espera (P)	Tiempo de retorno (P+X)
1	0	5
2	4	7
3	6	16
4	15	16
5	15	17

Los tiempos de espera y de retorno promedio son:

$$\text{Tiempo de espera} = \frac{0+4+6+15+15}{5} = \frac{40}{5} = 8 \text{ u.t.}$$

$$\text{Tiempo de retorno} = \frac{5+7+16+16+17}{5} = \frac{61}{5} = 12{,}2 \text{ u.t.}$$

Tiempo de ejecución más corto primero o SJF (*Shortest Job First*)

Este algoritmo no es apropiativo ni emplea prioridades. El algoritmo otorga el uso de la CPU al proceso en cola que tenga el tiempo de ejecución más bajo.

Dibujamos un diagrama que indique el estado de cada proceso en cada unidad de tiempo.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1	X	X	X	X	X																
P2		P	P	P	P	P	P	P	X	X	X										
P3			P	P	P	P	P	P	P	P	P	X	X	X	X	X	X	X	X	X	X
P4				P	P	X															
P5					P	P	X	X													

Los tiempos de espera y de retorno para cada proceso son:

Proceso	Tiempo de espera (P)	Tiempo de retorno (P+X)
1	0	5
2	7	10
3	9	19
4	2	3
5	2	4

Los tiempos de espera y de retorno promedio son:

Tiempo de espera = $\frac{0+7+9+2+2}{5} = \frac{20}{5}$ = 4 u.t.

Tiempo de retorno = $\frac{5+10+19+3+4}{5} = \frac{41}{5}$ = 8,2 u.t.

Tiempo restante más corto primero o SRTF (*Shortest Remaining Time First*)

Este algoritmo es apropiativo (un proceso puede arrebatar la CPU a otro cuando la esté usando) y no emplea prioridades. En cada unidad de tiempo, comprueba qué proceso tiene el tiempo de ejecución restante más bajo, ya sea un proceso en cola o que esté usando la CPU.

Dibujamos un diagrama que indique el estado de cada proceso en cada unidad de tiempo.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1	X	P	P	P	P	P	P	X	X	X	X										
P2		X	X	X																	
P3			P	P	P	P	P	P	P	P	P	X	X	X	X	X	X	X	X	X	X
P4				P	X																
P5					P	X	X														

Los tiempos de espera y de retorno para cada proceso son:

Proceso	Tiempo de espera (P)	Tiempo de retorno (P+X)
1	6	11
2	0	3
3	9	19
4	1	2
5	1	3

Los tiempos de espera y de retorno promedio son:

Tiempo de espera = $\frac{0+7+9+2+2}{5} = \frac{17}{5}$ = 3,4 u.t.

Tiempo de retorno = $\frac{11+3+19+2+3}{5} = \frac{38}{5}$ = 7,6 u.t.

Ejercicio 11 – Gestión de procesos 2: Round-Robin

Considere el siguiente conjunto de procesos planificados con un algoritmo Round-Robin (RR) con 1 unidad de tiempo de cuantum. ¿Cuánto tardan en acabar todos ellos?

Proceso	Tiempo de llegada	Tiempo de ejecución
1	2	8
2	0	5
3	1	4
4	3	3

Solución

En la gestión de procesos bajo el algoritmo RR (Round-Robin), se asigna a cada proceso la porción de tiempo de cuantum de forma equitativa y ordenada, tratando a todos los procesos con la misma prioridad. Una vez usado el tiempo de cuantum, el proceso vuelve a la cola (si aún no ha terminado) y se le asigna la CPU al proceso según FIFO. Dibujamos un diagrama que indique el estado de cada proceso en cada unidad de tiempo.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
P1			X	P	P	P	X	P	P	P	X	P	P	P	X	P	X	X	X	X
P2	X	P	P	P	X	P	P	P	X	P	P	P	X	P	P	X				
P3		X	P	P	P	X	P	P	P	X	P	P	P	X						
P4				X	P	P	P	X	P	P	P	X								

Los tiempos de ejecución para cada proceso son:

Proceso	Tiempo de ejecución
1	18
2	16
3	13
4	9

Ejercicio 11 – Gestión de procesos 3: NPP, RR y FIFO

Se tienen los siguientes trabajos a ejecutar:

Proceso	Unidades de tiempo	Prioridad
1	8	2
2	5	4
3	2	2
4	7	3

Éstos llegan en el orden 1, 2, 3 y 4, y la prioridad más alta es la del valor 1. Se pide:

a) Escribir un diagrama que ilustre la ejecución de esos trabajos usando:

- Planificación de prioridades no expulsiva o NPP (*Nom-Preemptive Priority*).
- Planificación cíclica (Round-Robin) con una rodaja de tiempo (cuantum) de 2 ut.
- Planificación primero en entrar, primero en salir o FIFO (*First In, First Out*).

b) Indicar cuál es el algoritmo de planificación con menor tiempo medio de espera

Solución

a) Escribir un diagrama que ilustre la ejecución de esos trabajos usando:

Como no se indica el tiempo de llegada de cada proceso, se da por hecho que ya se encuentran todos los procesos cargados y listos para ser ejecutados.

NPP:

Se ejecuta el proceso que tenga más prioridad de los que hay cargados en la CPU y no será desalojado ni interrumpido hasta que no finalice su ejecución, aunque lleguen otros procesos más prioritarios.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
P1	X	X	X	X	X	X	X	X														
P2	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	X	X	X	X	X
P3	P	P	P	P	P	P	P	P	X	X												
P4	P	P	P	P	P	P	P	P	P	P	X	X	X	X	X	X	X					

Tiempo de espera = $\frac{0+17+8+10}{4} = \frac{35}{4}$ = 8,75 u.t.

Tiempo de retorno = $\frac{8+22+10+17}{4} = \frac{57}{4}$ = 14,25 u.t.

Round-Robin:

Se ejecuta el proceso que tenga mayor prioridad durante el tiempo de cuantum. Una vez acabo el cuantum, se pone a la cola siguiendo un orden FIFO.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
P1	X	X	P	P	P	P	P	P	X	X	P	P	P	P	X	X	P	P	P	X	X	
P2	P	P	X	X	P	P	P	P	P	P	X	X	P	P	P	P	X					
P3	P	P	P	P	X	X																
P4	P	P	P	P	P	P	X	X	P	P	P	P	X	X	P	P	P	X	X	P	P	X

Tiempo de espera = $\frac{13+12+4+15}{4} = \frac{44}{4}$ = 11 u.t.

Tiempo de retorno = $\frac{21+17+6+22}{4} = \frac{66}{4}$ = 16,5 u.t.

FIFO:

Se ejecutan los procesos en el mismo orden de llegada a la CPU.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
P1	X	X	X	X	X	X	X	X														
P2	P	P	P	P	P	P	P	P	X	X	X	X	X									
P3	P	P	P	P	P	P	P	P	P	P	P	P	P	X	X							
P4	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	X	X	X	X	X	X	X

Tiempo de espera = $\frac{0+8+13+15}{4} = \frac{36}{4}$ = 9 u.t.

Tiempo de retorno (ejecución) = $\frac{8+13+15+22}{4} = \frac{58}{4}$ = 14,5 u.t.

b) Indicar cuál es el algoritmo de planificación con menor tiempo medio de espera

Se observa que el algoritmo que menor tiempo de espera produce es el de prioridades no expulsiva.

Ejercicio 13 – Gestión de entrada y salida

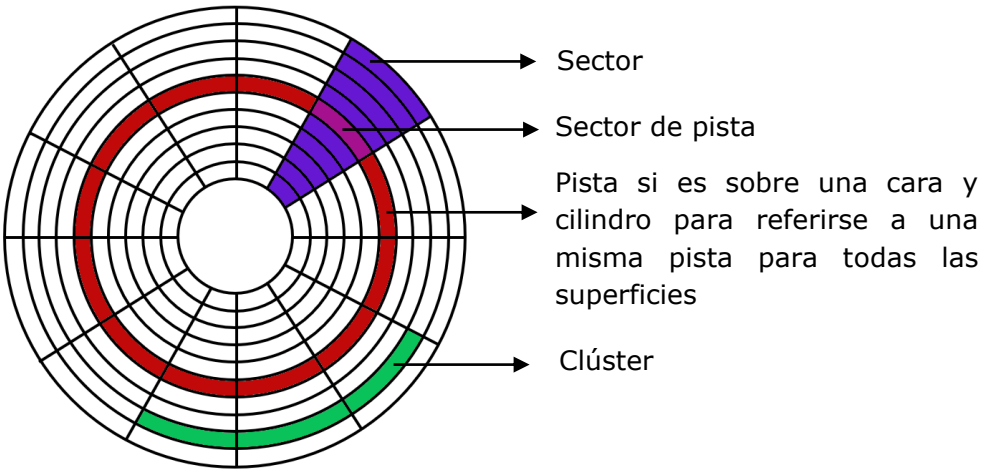
Un disquete de 3,5 pulgadas tiene 2 superficies, 80 cilindros, 18 sectores por pista y 512 bytes por sector.

a) Calcular su capacidad total

b) Suponiendo que el disco gira a 360 rpm, ¿cuál será la velocidad de transferencia?

Solución

Los soportes magnéticos almacenan la información mediante partículas magnéticas que hay en cada una de las caras de un plato o lámina. Cada cara o superficie está dividida por unas partes lógicas que hay que tener claras antes de empezar a hacer el ejercicio.



a) Calcular su capacidad total

Capacidad = 2 caras * 80 pistas/cara * 18 sectores/pista * 512 bytes/sector = 1474560 Bytes → 1,40 MBytes

b) Suponiendo que el disco gira a 360 rpm, ¿cuál será la velocidad de transferencia?

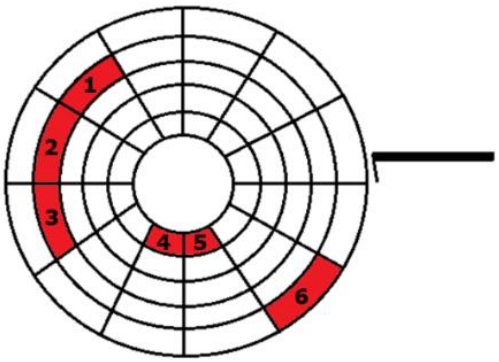
Para calcularlo, usamos la siguiente fórmula:

$$V_{transf} = \frac{Número\ de\ revoluciones\ /\ minuto}{60\ segundos\ /\ minuto} * Nº\ Sectores/pista * Nº\ Bytes/sector$$

$$V_{transf} = \frac{360\ rpm}{60\ segundos} * 18\ sectores/pista * 512\ Bytes/sector = 55296\ Bytes/seg.$$

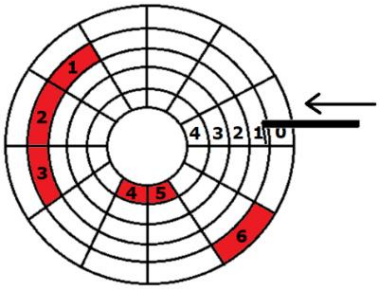
Ejercicio 14 – Gestión de entrada y salida 2

Una fotografía está almacenada en un disco duro. El archivo ocupa 6 clústeres que están repartidos por una superficie del disco (están fragmentados). Los clústeres que componen el archivo son los que están numerados y coloreados en rojo (cada clúster ocupa un sector). Escribe los movimientos que realizaría el disco duro para leer el archivo.

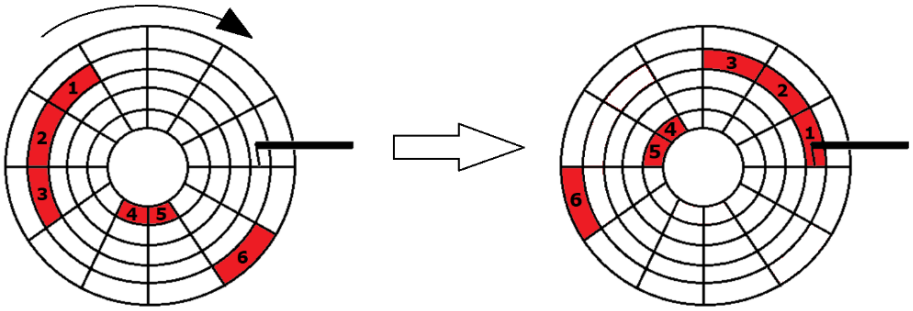


Solución

1. La cabeza se posiciona en la segunda pista (pista 1).



2. El disco rota 4 sectores.



3. El disco rota más lento 3 sectores mientras la cabeza lee la información que hay ahí (clústeres 1, 2 y 3).
4. La cabeza se posiciona en la quinta pista (pista 4).
5. El disco gira un sector.
6. El disco gira 2 sectores mientras la cabeza lee los clústeres 4 y 5
7. La cabeza se posiciona en la primera pista (pista 0)
8. El disco gira 1 sector mientras la cabeza lee el clúster 6.

Ejercicio 15 – Gestión de entrada y salida 3

Dado un disco duro con las siguientes características:

- El disco tiene 2 platos (4 superficies)
- Cada superficie tiene 120 pistas.
- Cada pista tiene 80 sectores.
- Cada sector de pista tiene una capacidad de 512 Bytes.
- Cada clúster es de 4KB (4096 Bytes)
- El tiempo que tarda la cabeza en recorrer una pista es de 0,25ms.
- El tiempo de estabilización de la cabeza es de 3ms.
- La sobrecarga debida al controlador es de 2ms.
- La velocidad de rotación es de 1500 rpm (revoluciones por minuto)

a) Calcular el tiempo total (tiempo de operación) que tardaría en leer 3 clústeres consecutivos. Para posicionarse la cabeza sobre el clúster que se va a leer, se tiene que desplazar 40 pistas y el disco tiene que rotar media vuelta.

b) Capacidad total del disco duro.

Solución

a) Calcular el tiempo total (tiempo de operación) que tardaría en leer 3 clústeres consecutivos. Para posicionarse la cabeza sobre el clúster que se va a leer, se tiene que desplazar 40 pistas y el disco tiene que rotar media vuelta.

Para obtener el tiempo de operación (T_{op}), hay calcular el tiempo de posicionado (T_{pos}), el tiempo de rotación (T_{rot}), el tiempo de transferencia (T_{transf}) y sumarlos junto con el tiempo de la controladora (T_{contr}).

$$T_{op} = T_{pos} + T_{rot} + T_{transf} + T_{contr}$$

Cálculo del Tiempo de Posicionado:

El tiempo de posicionado es el tiempo que tarda la aguja o cabeza en colocarse sobre la pista donde se encuentra el clúster a leer.

*T_{pos} = (Pistas a recorrer * T. que se tarda en recorrer una pista) + T. estabilización cabeza*

T_{pos} = (40 pistas * 0,25ms/pista) + 3ms = 10ms + 3ms = **13ms**

Cálculo del Tiempo de Rotación:

Primero se calcula el tiempo que tarda en dar una vuelta completa:

$$T_{rot\_completa} = \frac{60\ segundos\ /minuto}{Velocidad\ de\ giro\ (revoluciones\ /\ minuto)}$$

$T_{rot\_completa} = \frac{60}{1500}$ = 0,04 segundos = 40ms

Como el disco rota media vuelta, el tiempo de rotación es de **20ms**.

Cálculo del Tiempo de Transferencia:

Se van a leen 3 clústeres consecutivos de 4096 Bytes cada uno, por tanto, se van a leer un total de 12288 Bytes. Para calcular el tiempo de transferencia, primero es necesario calcular la velocidad de transferencia.

$$V_{transf} = \frac{Número\ de\ revoluciones\ /\ minuto}{60\ segundos\ /\ minuto} * Nº\ Sectores/pista\ *\ Nº\ Bytes/sector$$

$V_{transf} = \frac{1500\ \text{rpm}}{60\ \text{seg}}$ * 80 sectores/pista * 512 bytes/sector = 1024000 bytes/seg

Ahora ya se puede calcular el tiempo de transferencia.

$$T_{transf} = \frac{Cantidad\ de\ Bytes\ que\ se\ van\ a\ transferir}{Vtransf}$$

$T_{transf} = \frac{12288 \text{ Bytes}}{1024000 \text{ Bytes/segundo}}$ = 0,012 segundos = **12ms**

Cálculo del Tiempo de Operación:

Con todos los tiempos calculados, ya se puede calcular el tiempo de operación.

$$T_{op} = T_{pos} + T_{rot} + T_{transf} + T_{controladora}$$

T_{op} = 13ms + 20ms + 12ms + 2ms = **47ms**

b) Capacidad total del disco duro

Capacidad = 4 caras * 120 pistas/cara * 80 sectores/pista * 512 Bytes/sector = 19660800 Bytes → **18,75 MBytes**

Ejercicio 16 – Gestión de usuarios

Escribe los comandos de terminal de Linux que realicen las siguientes operaciones:

a) Crea un usuario con nombre de usuario "fulano", que pertenezca al grupo sudo, cuyo directorio de trabajo sea /home/temporal, que se cree el directorio de trabajo y que su shell sea /bin/bash.

b) Establece la contraseña "hola" al usuario "fulano".

c) Modifica el directorio de trabajo del usuario por /home/fulano, copiando los archivos que había en el antiguo directorio de trabajo al nuevo.

d) Añade un comentario al usuario que diga que es "desarrollador web".

e) Cambia de usuario efectivo al usuario "fulano".

f) Muestra un listado de usuarios conectados con cabeceras.

g) Muestra cuál es el usuario efectivo.

h) Elimina el usuario "fulano" y su directorio de trabajo.

Solución

a) Crea un usuario con nombre de usuario "fulano", que pertenezca al grupo sudo, cuyo directorio de trabajo sea /home/temporal, que se cree el directorio de trabajo y que su shell sea /bin/bash.

Se puede crear un nuevo usuario con 2 comandos: "*useradd*" y "*adduser*". Ambos comandos son similares, pero tienen un enfoque distinto.

El comando "***useradd***" se ejecuta en binario (archivo ejecutable) y se incluye en la mayoría de las distribuciones Linux, mientras que "***adduser***" es un script en Perl que usa el binario "*useradd*" en su ejecución, por tanto, es menos compatible, pero ofrece una interfaz más amigable.

La sintaxis de "*useradd*" es:

```
useradd [opciones] nombre_login
```

Las principales opciones del comando "*useradd*" son:

- -c [comentario]: Añade información del usuario como el nombre.
- -g [grupo]: Establece el grupo principal al que pertenecerá el usuario.
- -G [grupos]: Establecen los grupos secundarios separados por comas.
- -d [directorio de trabajo]: Establece la ruta del directorio de trabajo de ese usuario. Si no se especifica esta opción, se creará de forma predeterminada la ruta "/home/login_de_usuario/".
- -m: Se crea el directorio de trabajo, es decir, se copian los archivos de la carpeta "/etc/skel" en el directorio de trabajo. La carpeta "/etc/skel" contiene los archivos de configuración por defecto (están ocultos) que se añaden al directorio de trabajo de un usuario cuando este es creado.
- -u [userId]: Establece el ID del usuario.
- -s [shell]: Establece un intérprete de comandos al usuario. Si no se especifica, de forma predeterminada puede emplearse "/bin/bash", "/bin/csh" o "/bin/sh", dependiendo de la distro Linux.
- nombre_login: Nombre de usuario que se utilizará para iniciar sesión.

Por tanto, el comando que hace lo que se pide con "*useradd*" es:

```
sudo useradd -g sudo -d /home/temporal -m -s /bin/bash fulano
```

La sintaxis de "*adduser*" es:

```
adduser [opciones] nombre_login
```

Las principales opciones del comando "*adduser*" son:

- --gecos [comentario]: Añade información del usuario como el nombre.
- --ingroup [grupo]: Establece el grupo principal al que pertenecerá el usuario.
- --home [directorio]: Establece la ruta del directorio de trabajo de ese usuario.
- --no-create-home: No crea directorio de trabajo a ese usuario.
- --uid [userId]: Establece el ID del usuario.
- --shell [shell]: Establece un intérprete de comandos al usuario.
- nombre_login: Nombre de usuario que se utilizará para iniciar sesión.

Por tanto, el comando que hace lo que se pide con "*adduser*" es:

```
sudo adduser --ingroup sudo --home /home/temporal --shell /bin/bash fulano
```

b) Establece la contraseña "hola" al usuario "fulano".

```
sudo passwd fulano
```

Una vez escrito el comando, solicitará la contraseña.

c) Modifica el directorio de trabajo del usuario por /home/fulano, copiando los archivos que había en el antiguo directorio de trabajo al nuevo.

Para modificar un usuario existente se utiliza el comando "usermod" cuya sintaxis es:

```
usermod [opciones] nombre_login
```

Las principales opciones del comando "usermod" son:

- -g [grupo]: Establece un nuevo grupo principal al usuario.
- -G [grupos]: Establecen los grupos secundarios separados por comas.
- -a [grupos]: Añade grupos nuevos al usuario, en vez de reemplazar la lista de grupos. Se debe usar con la opción -G.
- -d [directorio de trabajo]: Asigna un nuevo directorio de trabajo al usuario. Si se emplea junto con el modificador "-m", todo el contenido del antiguo directorio de trabajo se moverá al nuevo.
- -l [login]: Establece un nuevo nombre de usuario para iniciar sesión.
- -s [*shell]*: Modifica el anterior *shell* por uno nuevo.

Por tanto, el comando que hace lo que se pide es:

```
sudo usermod –d /home/fulano –m fulano
```

d) Añade un comentario al usuario que diga que es "desarrollador web".

```
sudo usermod –c "desarrollador web" fulano
```

e) Cambia de usuario efectivo al usuario "fulano".

```
su fulano
```

f) Muestra un listado de usuarios conectados con cabeceras.

Para mostrar los usuarios con sesión iniciada, se usa el comando "who" cuya sintaxis es:

```
who [opciones]
```

Las principales opciones del comando "usermod" son:

- Sin opciones: Muestra un listado detallado de los usuarios conectados.

- -H: Imprime las cabeceras, es decir, qué es cada columna.
- -q: Muestra solamente el nombre de usuario y cantidad de usuarios conectados.

```
who -H
```

g) Muestra cuál es el usuario efectivo.

```
whoami
```

h) Elimina el usuario "fulano" y su directorio de trabajo.

Se puede eliminar un usuario existente con 2 comandos: "*userdel*" y "*deluser*".

El comando "***userdel***" se ejecuta en binario (archivo ejecutable) y se incluye en la mayoría de las distribuciones Linux, mientras que "***deluser***" es un script en Perl que usa el binario "*useradd*" en su ejecución.

La sintaxis de "*userdel*" es:

```
userdel [opciones] nombre_login
```

Las principales opciones del comando "*userdel*" son:

- -r: Borra también el directorio de trabajo del usuario.
- -f: Fuerza el eliminado del usuario, incluso si está actualmente conectado.

Por tanto, el comando que hace lo que se pide con "*userdel*" es:

```
sudo userdel -r fulano
```

La sintaxis de "*deluser*" es:

```
deluser [opciones] nombre_login
```

Las principales opciones del comando "*deluser*" son:

- --remove-home: Borra también el directorio de trabajo del usuario.
- --group: Borra también el grupo del usuario.
- --backup: Crea una copia de seguridad de sus archivos antes de borrarlo.
- --backup-to: Crea la copia de seguridad en la ruta indicada.

Por tanto, el comando que hace lo que se pide con "*deluser*" es:

```
sudo deluser --remove-home fulano
```

Ejercicio 17 – Gestión de grupos

Escribe los comandos de terminal de Linux que realicen las siguientes operaciones:

a) Crea un grupo llamado "SistInformaticos" con GID 1105.

b) Reemplaza el GID por 1106 y el nombre del grupo "SistInformaticos" por "aula"

c) Añade usuario "mengano" al grupo "aula".

d) Comprueba a qué grupos pertenece el usuario "mengano".

e) Elimina al usuario "mengano" del grupo "aula".

f) Elimina el grupo "aula".

Solución

a) Crea un grupo llamado "SistInformaticos" con GID 1105.

Para crear un nuevo grupo, se utiliza el comando "*groupadd*" cuya sintaxis es:

```
groupadd [opciones] grupo
```

La principal opción del comando "*groupadd*" son:

- -g [GID]: Asigna el identificador (GID) indicado. Se recomienda usar un valor igual o superior a 1000 y no puede haber 2 grupos con el mismo GID. Si no se especifica ningún GID, se pone uno automáticamente.

Por tanto, el comando que hace lo que se pide es:

```
sudo groupadd -g 1105 SistInformaticos
```

b) Reemplaza el GID por 1106 y el nombre del grupo "SistInformaticos" por "aula"

Para modificar un grupo, se utiliza el comando "*groupmod*" cuya sintaxis es:

```
groupmod [opciones] grupo
```

Las principales opciones del comando "groupadd" son:

- -g [GID]: Reemplaza el antiguo GID por uno nuevo. También se actualiza el archivo /etc/passwd con el nuevo GID si es necesario.
- -n [Nombre nuevo]: Reemplaza el nombre de grupo por uno nuevo.

Por tanto, el comando que hace lo que se pide es:

```
sudo groupmod -g 1106 -n aula SistInformaticos
```

c) Añade el usuario "mengano" al grupo "aula".

```
sudo adduser mengano aula
```

O también:

```
sudo usermod -a -G aula mengano
```

d) Comprueba a qué grupos pertenece el usuario "mengano".

```
groups mengano
```

e) Elimina al usuario "mengano" del grupo "aula".

```
sudo deluser mengano aula
```

f) Elimina el grupo "aula".

```
sudo groupdel aula
```

Ejercicio 18 – Gestión de usuarios y grupos

Escribe los comandos de terminal de Linux que realicen las siguientes operaciones:

a) Crea un usuario llamado "us1" con *adduser* cuyo id sea 1111.

b) Crea un grupo llamado "g1" con *addgroup*.

c) Crea con *adduser* los grupos "g2" y "g3" con gid 2222 y 2223 respectivamente.

d) Intenta crear el grupo G33 (mayúsculas) con *addgroup.*

e) Ejecuta la siguiente instrucción "*sudo adduser us1 g1*" y explica qué ha ocurrido.

f) Crea un nuevo usuario "us2" haciendo que su grupo primario sea "g2".

g) Crea un nuevo usuario us3 haciendo que su carpeta personal sea "*/home/usuario3*" y su id sea 3333.

h) Crea el usuario "us4" con la siguiente orden "*sudo adduser –disabled-login us4*". Intenta entrar como us4 ¿qué ocurre? ¿Cómo puedes activarlo para que pueda entrar?

i) ¿Cuál es el grupo principal del usuario "us3"? ¿Como lo averiguaste?

j) Modifica el grupo principal del usuario "us3" para que sea el "g3".

k) Sin usar *adduser* pon como grupo secundario de "us1" el "g3".

l) Borra el grupo "G33" con *delgroup.*

m) Borra el grupo "g2" con *deluser* pero solo si el grupo está vacío.

n) Crea los usuarios Julio, Marieli, Pablo y Lisa y los grupos "profes" y "alumnos" de manera que al final queden los siguientes grupos:

- profes al que pertenecen Julio y Marieli
- alumnos al que pertenecen Pablo y Lisa
- Julio
- Pablo

Solución

a) Crea un usuario llamado "us1" con *adduser* cuyo id sea 1111.

```
sudo adduser –uid 1111 us1
```

b) Crea un grupo llamado "g1" con *addgroup*.

```
sudo addgroup g1
```

c) Crea con *adduser* los grupos "g2" y "g3" con gid 2222 y 2223 respectivamente.

```
sudo addgroup ––gid 2222 g2
sudo addgroup ––gid 2223 g3
```

d) Intenta crear el grupo G33 (mayúsculas) con *addgroup.*

```
sudo addgroup G33
```

Es posible que se produzca un error porque en Linux los nombres se componen de letras minúsculas y números. Se puede forzar con el siguiente comando:

```
addgroup –force–badname G33
```

e) Ejecuta la siguiente instrucción "*sudo adduser us1 g1*" y explica qué ha ocurrido.

El usuario "us1" se ha agregado al grupo "g1" como grupo secundario.

f) Crea un nuevo usuario "us2" haciendo que su grupo primario sea "g2".

```
sudo adduser ––ingroup g2 us2
```

O también:

```
sudo useradd –g g2 us2
```

g) Crea un nuevo usuario us3 haciendo que su carpeta personal sea "*/home/usuario3*" y su id sea 3333.

```
sudo adduser --home /home/usuario3 --uid 3333 us3
```

O también:

```
sudo useradd -u 3333 -d /home/usuario3 us3
```

h) Crea el usuario "us4" con la siguiente orden "*sudo adduser –disabled-login us4*". Intenta entrar como us4 ¿qué ocurre? ¿Cómo puedes activarlo para que pueda entrar?

No se puede iniciar sesión porque el usuario está deshabilitado. Se activa así:

```
sudo passwd -u us4
```

i) ¿Cuál es el grupo principal del usuario "us3"? ¿Como lo averiguaste?

```
sudo id -gn us3
```

j) Modifica el grupo principal del usuario "us3" para que sea el "g3".

```
sudo usermod -g g3 us3
```

k) Sin usar *adduser* pon como grupo secundario de "us1" el "g3".

```
sudo usermod -a -G g3 us1
```

l) Borra el grupo "G33" con *delgroup.*

```
sudo delgroup G33
```

m) Borra el grupo "g2" con *deluser* pero solo si el grupo está vacío.

```
sudo deluser --only-if-empty -group g2
```

n) Crea los usuarios Julio, Marieli, Pablo y Lisa y los grupos "profes" y "alumnos" de manera que al final queden los siguientes grupos:

- profes al que pertenecen Julio y Marieli
- alumnos al que pertenecen Pablo y Lisa
- Julio
- Pablo

1º Se crean los usuarios. Al hacerlo, se crea también un grupo principal que se llama igual que los usuarios:

```
sudo adduser Julio
sudo adduser Marieli
sudo adduser Pablo
sudo adduser Lisa
```

2º Se crean los grupos:

```
sudo addgroup profes
sudo addgroup alumnos
```

3º Se mete cada usuario en su grupo:

```
sudo adduser Julio profes
sudo adduser Marieli profes
sudo adduser Pablo alumnos
sudo adduser Lisa alumnos
```

Otra forma:

```
sudo gpasswd –a Julio profes
sudo gpasswd –a Marieli profes
sudo gpasswd –a Pablo alumnos
sudo gpasswd –a Lisa alumnos
```

Ejercicio 19 – Permisos

Indica el tipo de recurso y los permisos que tienen las siguientes máscaras:

a) -rwxr-----

b) -rwxrw-r—

c) drwx------

Consideraciones previas

El primer dígito identifica el tipo de recurso. Un "–" significa archivo y una "d" directorio.

El resto de los caracteres indican los permisos, donde los 3 primeros se refieren al propietario, los 3 siguientes a los usuarios que pertenecen al mismo grupo que el propietario y los 3 últimos al resto de usuarios. La "r" significa que puede leer, la "w" que puede escribir y la "x" que puede ejecutar el recurso.

Solución

a) -rwxr-----

El recurso es un archivo. El usuario propietario tiene permisos de lectura, escritura y ejecución, el grupo propietario de lectura y el resto de los usuarios no tienen permisos.

b) -rwxrw-r—

Es un archivo. El usuario propietario puede leer, modificar y ejecutarlo, el grupo propietario puede leerlo y modificarlo y el resto de los usuarios solo pueden leerlo.

c) drwx------

Es una carpeta. Sólo el usuario propietario puede acceder a la carpeta y visualizar, modificar, eliminar, crear, etc. archivos y subcarpetas.

Ejercicio 20 – Permisos 2

Con el comando "chmod" y empleando el sistema de numeración octal y el modo simbólico, escribe los comandos que realicen las siguientes operaciones:

a) Establece los permisos rwxrwx--- al archivo notas.txt.

b) Establece los permisos rwxr--r-- a todos los archivos con extensión "txt" del directorio activo.

c) Establece los permisos rwxrwxrwx a la carpeta "Hola" y a los recursos que hay en ella.

d) Establece los permisos rwx---r-- a la carpeta "Hola", pero no a los recursos que hay en ella.

Consideraciones previas

Para modificar los permisos de un recurso, se usa "chmod" con la siguiente sintaxis:

```
chmod [opciones] [permisos] [archivo/s o carpeta/s]
```

La opción más importante es:

- -R: Cambia los permisos de subcarpetas y archivos recursivamente.

Los permisos pueden especificarse de 2 formas: en modo numérico y modo simbólico.

Permisos en modo numérico:

La máscara se almacena en el inodo en binario, donde un bit a 1 indica que ese permiso está activado y a 0 que está desactivado. Ejemplo:

rwxrw-r-- → 111110100

Una forma de asignar o modificar los permisos de un recurso con el comando "chmod" es escribiendo la máscara en octal (un dígito octal por cada 3 binarios).

$$rwxrw-r-- \rightarrow 111110100_{(2} \rightarrow 764_{(8}$$

Permisos en modo simbólico:

Para indicar los permisos en modo simbólico, primero se escriben los destinatarios, luego el tipo de modificación y, por último, los permisos, pudiendo ser los siguientes:

- Destinatarios de los permisos a modificar:
  - "u": Propietario (*user*).
  - "g": Grupo (*group*).
  - "o": Otros usuarios (*others*).
  - "a": Todos los usuarios, es decir, propietario, grupo y otros (*all*).
- Tipo de modificación:
  - "+": Se añaden los permisos a los que hay ya establecidos.
  - "=": Establece esos permisos, anulando los que hay establecidos.
  - "-": Se quitan los permisos a los que hay ya establecidos.
- Permisos:
  - "r": Lectura (*read*).
  - "w": Escritura (*write*).
  - "x": Ejecución (*execute*).

Solución

Con el comando "chmod" y empleando el sistema de numeración octal y el modo simbólico, escribe los comandos que realicen las siguientes operaciones:

a) Establece los permisos rwxrwx--- al archivo notas.txt.

Modo numérico:

```
sudo chmod 770 notas.txt
```

Modo simbólico:

```
sudo chmod ug=rwx notas.txt
```

b) Establece los permisos rwxr--r-- a todos los archivos con extensión "txt" del directorio activo.

Modo numérico:

```
sudo chmod 744 *.txt
```

Modo simbólico:

```
sudo chmod u=rwx,go=r *.txt
```

c) Establece los permisos rwxrwxrwx a la carpeta "Hola" y a los recursos que hay en ella.

Modo numérico:

```
sudo chmod -R 777 Hola
```

Modo simbólico:

```
sudo chmod -R a=rwx Hola
```

d) Establece los permisos rwx---r-- a la carpeta "Hola", pero no a los recursos que hay en ella.

Modo numérico:

```
sudo chmod 704 Hola
```

Modo simbólico:

```
sudo chmod u=rwx,o=r Hola
```

Ejercicio 21 – Permisos 3

Con el comando "chmod" y empleando el modo simbólico, escribe los comandos que realicen las siguientes operaciones:

a) Añade permiso de ejecución para el propietario del fichero "archivo.txt".

b) Quita permiso de escritura a Julio sobre el archivo "archivo.txt".

c) Añade permiso de escritura y ejecución al grupo del fichero "archivo.txt".

d) Quita el permiso de lectura a la gente que no pertenece al grupo del fichero "archivo.txt".

e) Establece que los permisos para el grupo del fichero "archivo.txt" sean de lectura y escritura independientemente que los que ya tuviera.

f) Cuando sabemos cómo tienen que quedar todos los permisos, podemos usar el modo numérico. Haz que los permisos para "archivo.txt" sean los siguientes:

- -r-x-wxr--
- --w-r-x--x
- -rwxrwxrwx
- -r--r--r--
- --w--wxrw-

Solución

a) Añade permiso de ejecución para el propietario del fichero "archivo.txt".

```
sudo chmod u+x archivo.txt
```

b) Quita permiso de escritura a Julio sobre el archivo "archivo.txt".

```
sudo chmod u-w archivo.txt
```

c) Añade permiso de escritura y ejecución al grupo del fichero "archivo.txt"

```
sudo chmod g+wx archivo.txt
```

d) Quita el permiso de lectura a la gente que no pertenece al grupo del fichero "archivo.txt".

```
sudo chmod o-r archivo.txt
```

e) Establece que los permisos para el grupo del fichero "archivo.txt" sean de lectura y escritura independientemente que los que ya tuviera.

```
sudo chmod g=rw archivo.txt
```

f) Cuando sabemos cómo tienen que quedar todos los permisos, podemos usar el modo numérico. Haz que los permisos para "archivo.txt" sean los siguientes:

- -r-x-wxr--

```
sudo chmod 534 archivo.txt
```

- --w-r-x--x

```
sudo chmod 251 archivo.txt
```

- -rwxrwxrwx

```
sudo chmod 777 archivo.txt
```

- -r--r--r--

```
sudo chmod 444 archivo.txt
```

- --w--wxrw-

```
sudo chmod 236 archivo.txt
```

Ejercicio 22- Examen 2002

Tenemos la siguiente sucesión de referencias a memoria, generadas en la ejecución de un programa de usuario:

0097A, 0017C, 00314, 00380, 006A0, 004A0, 002B0, 002B6, 003BB, 00700, 00052, 004C00…

El proceso en cuestión puede disfrutar como máximo de 5 marcos de página. El contenido inicial de los 5 marcos de página asignados al proceso se representa en la primera columna: en el primer marco de página se almacena la página 1, en el segundo la 7, en el tercero la 3, etc.

	9	1	3	6	4	2	3	7	0	4	6	9	2	4	1	2	3
1																	
7																	
3																	
6																	
9																	

Se pide:

a) Completar la figura aplicando el algoritmo de sustitución ÓPTIMO o de Anticipación. En caso de igualdad entre 2 páginas, se elegirá una al azar.

b) ¿Cuántas faltas de páginas se producen?

Solución

a) Completar la figura aplicando el algoritmo de sustitución ÓPTIMO o de Anticipación. En caso de igualdad entre 2 páginas, se elegirá una al azar.

Se va a utilizar otra forma de rellenar la tabla que consiste en poner un * donde se produce acierto y en poner el número que sustituye donde se produce el fallo, dejando el resto de las celdas en blanco.

	9	1	3	6	4	2	3	7	0	4	6	9	2	4	1	2	3
1		*			4					*				*	1		
7								*	0			9					
3			*				*										*
6				*							*						
9	*					2							*			*	

b) ¿Cuántas faltas de páginas se producen?

Fallos = 5