



NOMBRE y APELLIDOS:

AULA:

PUESTO:

- Durante el examen solo se podrán utilizar libros (no apuntes)
- Este ejercicio supone un 20 % de la calificación de la asignatura.
- Cada uno de los dos ejercicios tiene el mismo valor.
- Se dispondrá de 1 hora para realizar este ejercicio.
- Se contestará cada ejercicio en su mismo enunciado, en los espacios habilitados para ese fin.
- Las calificaciones se publicarán el día 5 de julio y la revisión de examen tendrá lugar el 8 de julio, a las 10:00 en el EL5.

## Ejercicio 1 Memoria y Memoria Virtual

Un determinado sistema bajo estudio emplea paginación con doble nivel de indirección en la tabla de páginas (paginación paginada). Se sabe que, para un proceso concreto, el directorio de páginas se encuentra en la posición física 0x12345000 y tiene 1024 entradas de cuatro bytes cada una. Cada una de las entradas contiene, en sus 20 bits más significativos, el marco donde se aloja la tabla de páginas correspondiente, y en los bits menos significativos banderas para indicar si el marco está presente, si ha sido referenciado y si ha sido modificado.

Tras analizar una de las tablas de páginas, se comprueba que ésta tiene una estructura similar al directorio de páginas, que tiene el mismo número de entradas y el formato de cada una de las entradas es así mismo idéntico.

Por su parte, el tamaño de las páginas y de los marcos de página es el mismo para ambas e igual a 4KiB.

1. Con estos datos, ¿cuál es el formato de la dirección tanto física como virtual? Es posible que no se puedan deducir por completo todos los componentes de esta a partir de los datos del enunciado. Si este fuera el caso, explíqué qué necesitaría saber.

2,5 puntos

Sea otro sistema con una capacidad de direccionamiento de 32bit, con 10bit para indicar el número de entrada en la tabla de directorio, 10bit para indicar el número de entrada en la tabla de páginas, y los 12bit restantes para indicar el desplazamiento.

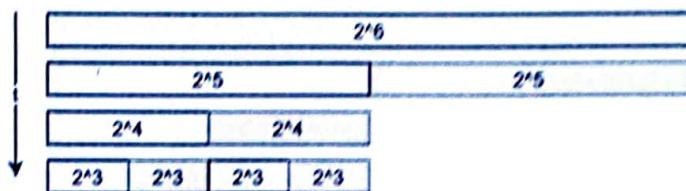
2. ¿Cuánta cantidad de memoria como máximo se puede reservar de forma que se utilice solo una entrada del directorio de páginas?
3. En el peor de los casos, ¿cuántas entradas de la tabla de directorio serán necesarias para alojar 5KiB?
4. Suponiendo que el espacio de direccionamiento virtual de un proceso estuviera absolutamente vacío ¿cuánta memoria se asignaría para soportar dos escrituras de un único byte, en las posiciones de memoria virtual 0x00000001 y 0x00000002?
5. ¿Y si las dos escrituras fueran en las posiciones de memoria 0x10000000 y 0x20000000?. Indique además cuánta memoria sería necesaria para las estructuras típicas de este esquema de gestión de memoria.

5 puntos

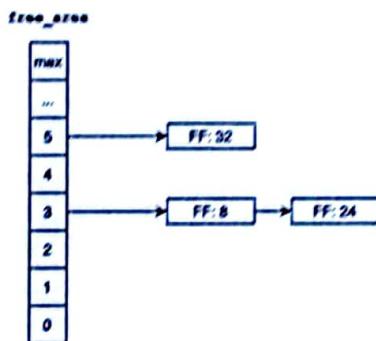
Se sabe que un sistema operativo lleva la contabilidad de marcos libres utilizando el **sistema Buddy**. Para esto mantiene una estructura denominada **free\_area** en la que se agrupan los buddies disponibles. Así, **free\_area[n]** permite acceder a la lista de buddies de  $2^n$  marcos contiguos.

El sistema en cuestión se ejecuta con una **CPU** que tiene una capacidad de **direcciónamiento de 16 bits**, con un **tamaño de marco de 1KiB**.

La memoria libre ha ido evolucionando como muestra el siguiente diagrama hasta llegar al momento actual. Los recuadros más oscuros indican que ese buddy se encuentra libre.



El siguiente diagrama muestra el estado de **free\_area** en ese momento:



Se puede ver que hay un único buddy de  $2^5 = 32$  marcos consecutivos, empezando en el marco número 32, apuntado por la entrada 5 de **free\_area** ( $2^5$ ). De la misma forma, en **free\_area[3]** tendremos dos buddies de  $2^3 = 8$  marcos consecutivos, uno de esos bloques comprende los marcos numerados desde 8 a 15, y el otro desde el 24 al 31.

Con esta información, responda a las siguientes preguntas:

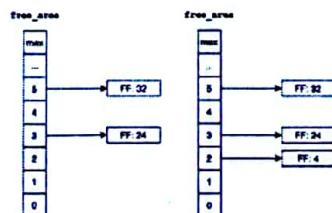
6. ¿Cuál es el valor de **max** en el diagrama anterior de **free\_area**?
7. ¿Cuánta memoria libre hay en el sistema en este momento, expresada en KiB?. Razoné brevemente su respuesta.
8. ¿Qué pasaría si el núcleo del sistema operativo solicitara 7 marcos de página? ¿y si solicitara 4 marcos de página? Muestre el estado del sistema buddy usando un diagrama como el mostrado anteriormente para ilustrar ambos casos.

**2,5 puntos**

## Solución ejercicio 1

### Memoria y Memoria Virtual

1. Para la dirección física, si el tamaño del marco es de 4KiB, serán necesarios 12 bits para el offset. Pero de los datos del enunciado no se puede deducir la capacidad de direccionamiento de la CPU, por lo tanto no se puede determinar el tamaño de la dirección ni el número de marcos. En el caso de la dirección virtual, siguiendo el mismo razonamiento anterior, se puede deducir que se necesitan 12 bits para el offset. Como la tabla de directorio tiene 1024 entradas necesitaremos 10 bits para el número de entrada de directorio. Como las tablas de páginas son iguales serán necesarios otros 10 bits para el número de páginas. En resumen **10 bits para la entrada del directorio, 10 para el número de página y 12 para el offset.**
2. Habría que contar solo con las 1024 entradas de la única tabla de páginas que podríamos tener. Cada tabla de páginas tiene  $2^{10}$  entradas, cada una con un posible marco de  $2^{12}$  bytes o 4KiB. En total serían  $2^{10} * 2^{12} = 2^{22}$ bytes o 4MiB.
3. 5KiB son dos marcos de 4KiB, en los que se produce fragmentación interna. Podría darse el caso de que el primer marco se alojara en la última página de la tabla de páginas correspondiente a la entrada de directorio n, y el segundo marco en la primera página de la tabla de páginas correspondiente a la entrada de directorio n+1.
4. En el primer caso, como las dos escrituras caen en la misma página, las necesidades serían las siguientes:
  - 1 marco para el directorio de páginas
  - 1 marco para la tabla de páginas
  - 1 marco para los datos en sí
 En total 3 marcos, 12KiB
5. En el segundo caso, las dos escrituras caen en dos páginas diferentes, así es que las necesidades serían:
  - 1 marco para el directorio de páginas
  - 1 marco para la primera tabla de páginas
  - 1 marco para la segunda tabla de páginas
  - 1 marco para los datos en sí de la primera escritura
  - 1 marco para los datos en sí de la segunda escritura
 En total 5 marcos, 20KiB
6. Si la capacidad máxima del sistema es 64KiB, entonces  $\log_2[n] \llbracket n=64 \rrbracket$ , donde n=6.
7. Ahora mismo hay 3 buddies libres, uno de  $2^5 = 32$  y dos de  $2^3 = 8$ , que sumados son 48 marcos. Cada marco ocupa 1KiB, así es que el tamaño total será de 48KiB.
8. En un buddy de  $2^3$  marcos consecutivos caben los 7 marcos solicitados, pero así en uno de orden inferior, con lo cual se asignaría un buddy de 8 marcos. En el caso de que se solicitaran 4 marcos se dividiría el buddy de  $2^3$  en dos de  $2^2$  y se asignaría uno de ellos.





NOMBRE y APELLIDOS:

AULA:

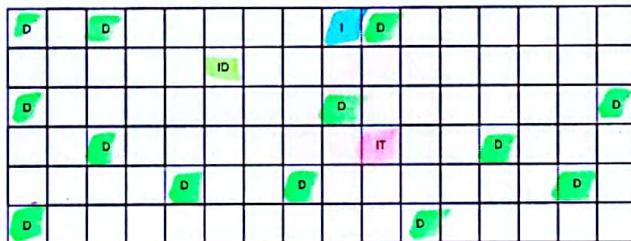
PUESTO:

- Durante el examen solo se podrán utilizar libros (no apuntes)
- Este ejercicio supone un 20 % de la calificación de la asignatura.
- Cada uno de los dos ejercicios tiene el mismo valor.
- Se dispondrá de 1 hora para realizar este ejercicio.
- Se contestará cada ejercicio en su mismo enunciado, en los espacios habilitados para ese fin.
- Las calificaciones se publicarán el día 5 de julio y la revisión de examen tendrá lugar el 8 de julio, a las 10:00 en el ELS.

## Ejercicio 2

### Sistemas de Archivos

Un particular sistema de archivos de tipo i-node opera con tamaño de bloque considerable de 1GiB. Para indicar los bloques que contiene un archivo, su i-node dispone de cuatro entradas de tipo puntero: uno directo, otro indirecto, otro indirecto doble y por último uno indirecto triple. El siguiente diagrama muestra el estado de los primeros bloques de datos, ordenados por orden creciente de numeración, de izquierda a derecha y de arriba abajo.



Los bloques que aparecen sombreados indican que dicho bloque está ocupado, y las letras significan: D, bloque de datos del archivo, I, bloque que contiene punteros indirectos, ID, bloque que contiene punteros indirectos dobles, IT, bloque que contiene punteros indirectos triples. Además, por restricciones externas, los bloques de punteros solo pueden almacenar a su vez cuatro punteros.

- Rellene en la plantilla adjunta cual sería la ocupación final de los bloques tras una escritura de 14,5GiB. Indique para cada bloque cuál sería el contenido correspondiente utilizando la nomenclatura descrita anteriormente.

4 puntos

- ¿Cuántos bloques se necesitan para almacenar el archivo más grande posible? Se define una métrica de eficiencia como la relación entre el número de bloques que ocupan los datos útiles del archivo y el número total de bloques que es necesario utilizar para almacenar el archivo. ¿Cuál es esta relación en el SdA de este ejercicio?

3 puntos

- ¿Qué se podría hacer para mejorar la eficiencia definida en el apartado anterior? ¿Reducir el tamaño del bloque? ¿Aumentar el número de bloques? ¿Aumentar la cantidad de punteros permitidos en los bloques de punteros?

3 puntos



1. La ocupación de los bloques sería la que muestra el siguiente diagrama:

	D		I	D	D	D	D			ID	I	D	D	D	D
I	D	D	D	D		I	D	D							

2. El archivo más grande sería el que usa toda la capacidad de direccionamiento que proporcionan los punteros del i-node correspondiente:

- Un puntero directo = 1 bloque D
- Un puntero indirecto = 1 bloques I , 4D
- Un puntero indirecto doble = 1 bloque ID , 4 bloques I, 16D
- Un puntero indirecto triple = 1 bloque IT + 4 bloques ID + 16 bloques I, 64D =

Total  $1+1+4+1+4+16 = 27$

Ratio  $85 / 85+27 = 75.8\%$

3. Aumentar la cantidad de punteros permitidos en los bloques de punteros. Por ejemplo, si aumentamos de 4 punteros a 8.

- Un puntero directo = 1 bloque
- Un puntero indirecto = 8 bloques
- Un puntero indirecto doble =  $8^2 = 16$  bloques
- Un puntero indirecto triple =  $8^3 = 512$  bloques

Total 537 bloques de datos.

- Un puntero directo = 0 bloque
- Un puntero indirecto = 1 bloques I
- Un puntero indirecto doble = 1 bloque ID + 8 bloques I
- Un puntero indirecto triple = 1 bloque IT + 8 bloques ID + 64 bloques I

Total  $1+1+8+1+8+64=83$

Ratio  $537/537+83 = 86\%$



## Solución ejercicio 2

### Sistemas de Archivos

Sea un disco duro con dos unidades lógicas, cada una con su correspondiente sistema de archivos.

La Partición 1 está formateada con un sistema tipo Unix y tiene 256 GB de capacidad, según las características que se reflejan el siguiente esquema:

Sector arranque	Super-bloque	Mapa bits bloques libres	Mapa bits de nodos-i libres	Tabla nodos-i	Bloques de datos
1bloque	512 bloques	511 bloques	2048 bloques		

El superbloque proporciona la siguiente información:

- Tamaño del bloque de datos: 4 KB.
- Un directorio ocupa 1 bloque de datos.
- Un bloque de datos se direcciona con 32 bits (4 B).
- En los mapas de bits, un bit a 1 indica que está libre dicho bloque o nodo-i.

Cada entrada en la tabla de nodos-i mantiene la siguiente información:

- Tipo de archivo: regular (REG), directorio (DIR), enlace (SLNK) y tubería (FIFO).
- Contador de enlaces fuertes (*hardlinks*).
- Tamaño del archivo en bytes.
- 1 puntero directo, 1 puntero indirecto simple, 1 puntero indirecto doble y 1 puntero indirecto triple.
- Otros: atributos tales como la fecha de modificación, propietario, permisos, etc.

Cada entrada de directorio consta de: nombre de archivo y nodo índice.

Se sabe que los nodos índice y los bloques de datos se han asignado de forma ascendente, a medida que se han creado los archivos. De tal modo, en el instante t se tiene lo siguiente:

Tabla de nodos-i:

Nodo-i	2	3	4	5	6	7
Tipo archivo	DIR	DIR	REG	DIR	DIR	DIR
Nº enlaces	5	2	2	2	3	2
Tamaño (bytes)	4096	4096	32.000	4096	4096	4096
Ptr. directo	0	1	2	10	11	12
Ptr. ind. simple	-	-	3	-	-	-
Ptr. ind. doble	-	-	-	-	-	-
Ptr. ind. triple	-	-	-	-	-	-

## Bloques de datos:

0	1	2	3	4	5-9	10	11	12
.	2	.	3	a)	b)	c)	d)	.
..	2	..	2				5	6
bin	3						..	7
project	4					p1.pdf	2	6
tmp	5						ecm	
home	6						7	

1. Indique qué información debería aparecer en: (1 punto)

- a) Primeros 4KB del contenido del archivo de nodo-i 4, del byte 0 al byte 4.095.
- b) Punteros a los bloques 4, 5, 6, 7, 8, 9.
- c) y d) Siguientes 4KB del contenido del archivo de nodo-i 4, del byte 8.191 al byte 31.999.

2. Suponga que el directorio actual es el raíz. Indique las modificaciones del sistema de archivos (solo las modificaciones) tras ejecutarse las órdenes siguientes: (3 puntos)

```
$ cp /tmp/p1.pdf p1.pdf.bak
$ cd tmp
$ ln -s p1.pdf ../p1.pdf.slnk
$ rm p1.pdf
```

Mapa de nodos-i libres: --00 0000 0011 1111 1111 1111 1..1

Mapa de bloques libres: 0000 0000 0000 0000 0000 0011 1..1

## Tabla de nodos-i:

Nodo-i	2	3	4	5	6	7	8	9
Tipo archivo	DIR	DIR	REG	DIR	DIR	DIR	REG	SLNK
Nº enlaces	5	2	2-1	2	3	2	1	1
Tamaño (bytes)	4096	4096	32.000	4096	4096	4096	32.000	8
Ptr. directo	0	1	2	10	11	12	13	21
Ptr. ind. simple	-	-	3	-	-	-	14	-
Ptr. ind. doble	-	-	-	-	-	-	-	-
Ptr. ind. triple	-	-	-	-	-	-	-	-

## Bloques de datos:

0	1	2	3	4-9	10	11
.	2	.	3	Cont. nodo-i	4	.
..	2	..	2		5	5
bin	3		4		6	6
Project	4				7	
tmp	5				8	
home	6				9	
p1.pdf.bak	8					
p1.pdf.slnk	9					

12	13	14	15 – 20	21
.	7	Cont. nodo-i	15	Cont. nodo-i
..	6	8	16 17 18 19 20	./p1.pdf

3. Calcule cuántos bytes podría almacenar el archivo de mayor tamaño (Justifique su respuesta y muestre las operaciones realizadas). **(2 puntos)**

El archivo de mayor tamaño, utilizará tanto bloques de datos, como bloques de punteros.

Un bloque de punteros puede referenciar,  $\frac{2^{12} \text{ bytes/bloque}}{2^2 \text{ bytes/ptr}} = 2^{10} \text{ punteros/bloque.}$

Realicemos varias consideraciones que nos permitirán calcular cuál es la estimación correcta:

(1) El archivo de mayor tamaño almacenará la información mantenida por:

- 1 puntero directo 1 bloque de datos.
- 1 puntero indirecto simple 1 bloque de punteros +  $2^{10}$  bloques de datos.
- 1 puntero indirecto doble 1 bloque ptrs +  $2^{10}$  bloques de ptrs +  $2^{20}$  bloques datos.
- 1 puntero indirecto triple 1 bq ptrs +  $2^{10}$  bqs ptrs +  $2^{20}$  bqs ptrs +  $2^{30}$  bqs. datos.

De este modo, si consideramos únicamente los bloques de datos que almacenan información, tendríamos capacidad para un archivo de tamaño:

- $(1 \text{ bloque datos} + 2^{10} \text{ bloques datos} + 2^{20} \text{ bloques datos} + 2^{30} \text{ bloques datos}) \cdot 2^{12} \text{ bytes/blq} \sim 4 \text{ TB.}$

(2) Por otro lado, observemos el mapa de bits de bloques libres. Este ocupa 512 bloques, con ello se tiene que:

- $2^9 \text{ bloques} \cdot 2^{12} \text{ bytes/bloque} \cdot 2^3 \text{ bits/byte} = 2^{24} \text{ bits ocupa del mapa de bits de bloques libres.}$   
Con ello, se pueden mantener  $2^{24} \text{ bloques} \cdot 2^{12} \text{ bytes/bloque} \sim 2^{36} \text{ bytes} \sim 64 \text{ GB.}$

(3) Finalmente, observemos el caso de un puntero, que tiene 32 bits. Esto quiere decir que pueden direccionarse:

- $2^{32} \text{ bloques} \cdot 2^{12} \text{ bytes/bloque} \rightarrow \sim 2^{44} \text{ bytes} \sim 16 \text{ TB.}$

Con todo ello, y debido a las restricciones del sistema, caso (2), el archivo de tamaño máximo ocuparía:

- $2^{24} \text{ bloques, es decir, } 2^{24} \text{ bloques} \cdot 2^{12} \text{ byte/bloque} \sim 2^{36} \text{ bytes} = 64 \text{ GB.}$

La Partición 2 está formateada con un sistema de archivos tipo FAT32 y tiene 126 GB de espacio de almacenamiento, con las siguientes características:

MBR	Reservada	FAT	Copia FAT	Clústeres
	Sector 0..31	Sector 32..231	Sector 232..431	Sector 432..

En la zona reservada se mantiene la BS\_Structure con la siguiente información:

- Nº de bytes por sector: 512 B.
- Nº de sectores por clúster: 8.
- Clúster del directorio raíz: 2.
- Otros.

Cada directorio contiene 15 entradas con:

- Nombre, que ocupa 8 + 3 caracteres.
- Atributos del archivo, que puede adoptar los valores: VOL, DIR o ARCHIVO.
- 1<sup>er</sup> clúster.
- Tamaño.

Cada entrada de la FAT es de 32 bits.

A continuación, se muestra el nombre de cada ítem, su tamaño y fecha de creación:

Etiqueta volumen: PRACTICAS

⌘ config.sys	500 B	25-09-2020
└ SOA	512 B	01-01-2021
└ ej1.c	1024 B	22-01-2021
└ libro.txt	3000 B	25-01-2021
└ Otros	512B	05-01-2021

Suponga que la información se almacena en el primer clúster libre, según su fecha de creación.

4. Rellene, en la siguiente tabla, la información mantenida en la zona de clústeres.

(2 puntos)

Nombre	1 <sup>er</sup> clúster	Offset	Atributo
Dir. raíz	2	0x36000	DIR
config.sys	3	0x37000	ARCHIVO
SOA	4	0x38000	DIR
ej1.c	6	0x3A000	ARCHIVO
libro.txt	7	0x3B000	ARCHIVO
Otros	5	0x39000	DIR

Muestre los cálculos realizados:

$$\begin{aligned}
 \text{Offset (2)} &= ((2-2) * 8 + 432) * 512 = 0x36000 \\
 \text{Offset (3)} &= ((3-2) * 8 + 432) * 512 = 0x37000 \\
 \text{Offset (4)} &= ((4-2) * 8 + 432) * 512 = 0x38000 \\
 \text{Offset (5)} &= ((5-2) * 8 + 432) * 512 = 0x39000 \\
 \text{Offset (6)} &= ((6-2) * 8 + 432) * 512 = 0x3A000 \\
 \text{Offset (7)} &= ((7-2) * 8 + 432) * 512 = 0x3B000
 \end{aligned}$$

5. Rellene en la siguiente tabla el contenido de la FAT del sistema de archivos propuesto, sabiendo que cada entrada es de 32 bits. **(2 puntos)**

La región de FAT tendría el siguiente contenido::

Clúster Offset	0 1 2 3	4 5 6 7	8 9 A B	C D E F
0x00000400	Clúster 0 F8 FF FF 0F	Clúster 1 FF FF FF 0F	Clúster 2 F8 FF FF 0F	Clúster 3 FF FF FF 0F
0x00000401	Clúster 4 F8 FF FF 0F	Clúster 5 F8 FF FF 0F	Clúster 6 FF FF FF 0F	Clúster 7 FF FF FF 0F
0x00000402	Clúster 8 00 00 00 00	Clúster 9 00 00 00 00	...	...

Sabiendo que el clúster 0 y clúster 1 están reservados, y que los archivos del ejemplo solo ocupan un clúster, el significado de las entradas de la FAT es:

- 0x?FFFFFF8: Primer clúster de la región de datos. El ultimo byte se corresponde con el BPB\_Media.
- 0x?FFFFFF: Último clúster del fichero o directorio, es decir, que el fichero finaliza en este clúster.
- 0x00000000: Clúster libre.



## Solución ejercicio 1

### Memoria y Memoria Virtual

Sea un microcomputador que gestiona un esquema de gestión de memoria que soporta paginación pura, con direccionamiento a nivel de byte y las siguientes características:

- El rango de direccionamiento de la memoria física es 20 bits mientras que el rango de direccionamiento virtual es de 32 bits.
- El tamaño de la página es de 64 KB.
- Cada entrada de la tabla de páginas contiene, además del marco de página, 4 bits de estado, los bits P, R, D, W. Cuando están activos, señalan que la página está presente (P), ha sido referenciada (R), ha sido escrita (Dirty) y si está permitida la escritura (Write).

1. Con todo ello, se pide (2,5 puntos):

(a) ¿Cantidad máxima de memoria física que el computador puede direccionar?

El rango de direccionamiento es 20 bits:  $2^{20}$  direcciones = 1MB.

(b) ¿Formato de una dirección física?

4 bits para el marco y 16 bits para el desplazamiento

(c) ¿Cuál es el tamaño del mayor proceso que pueda cargarse en memoria?

Capacidad de direccionamiento virtual ( $2^{32}$ ) = 4GB.

(d) ¿Formato de una dirección virtual?

16 bits para la página y 16 bits para el desplazamiento

(e) ¿Tamaño en bytes de la tabla de páginas de un proceso?

$2^{16}$  entradas, cada entrada 1 byte, la tabla ocupa  $2^{16}$  bytes = 64KB.

En un instante dado se crea un proceso que requiere 1 MB de memoria. Si la asignación se realiza de forma contigua desde el comienzo del espacio virtual.

2. ¿Cuántas páginas necesita el proceso y cuáles serían estos números de página? (1 punto)

Tamaño del espacio virtual del proceso:  $2^{20}$

Tamaño de una página:  $2^{16}$

Cantidad de páginas:  $2^{20} / 2^{16} = 2^4 = 16$  páginas.

Si la asignación se realiza de forma contigua, números de página de la 0 a la 15.

En un instante posterior, la tabla de páginas del proceso contiene la siguiente información, siendo la página 7 la más antigua:

Página	PRDW (Control página)	Marco
...		
3	0111 ->	0x3
4	1111 ->	0x3
5	1001 -> 1111	0x2
6	0111 -> 1111	0x2 -> 0x4
7	1001 -> 0001	0x4
...		

Y el proceso ejecuta el siguiente código:

```
char *p = 0x5FFFA;
strcpy( p, "EXAMEN SOA" );
```

5. Indique el contenido de la memoria física (byte a byte) después de la copia. Si la copia no fuera posible, indique qué acciones realizaría el S.O. Asigne el marco 4 a la página errónea para poder finalizar la copia. Indique también sobre la tabla de páginas anterior, las modificaciones que se puedan producir en los bits de control y los marcos (1,5 puntos)

Dirección virtual --> Dirección física	Contenido
0x5FFFA --> 0x2FFFA	'E'
0x5FFFB --> 0x2FFFB	'X'
0x5FFFC --> 0x2FFFC	'A'
0x5FFFD --> 0x2FFFD	'M'
0x5FFE --> 0x2FFE	'E'
0xFFFF --> 0xFFFF	'N'
0x60000 --> Fallo de página!!! Se asigna el marco 4: 0x4000	..
0x60001 --> 0x40001	'S'
0x60002 --> 0x40002	'O'
0x60003 --> 0x40003	'A'

El sistema de memoria virtual realiza paginación bajo demanda y usa un algoritmo de reemplazo FIFO con segunda oportunidad. Al proceso se le han asignado 3 marcos de página y se parte de la siguiente situación inicial de la tabla de marcos y el puntero del algoritmo FIFO, segunda oportunidad, estará posicionado en el marco M<sub>2</sub>:

	<b>Bit R</b>	<b>Página</b>
M <sub>0</sub>	1	0x00
M <sub>1</sub>	1	0x01
M <sub>2</sub> *	0	0x02

3. Sabiendo que la cadena de referencias es: 3, 0, 2, 3, 0, 1, 5, 1, 2, 4, 3, indique la secuencia de accesos y cuantos fallos de página se producen (5 puntos).

Cad refs	3	0	2	3	0	1	5	1	2	4	3
Page Fault?	PF		PF		PF	PF	PF	PF	PF	PF	
Marco 0											
	0	0	2	2	2	2	5	5	5	3	
Marco 1											
	1	1	1	1	0	0	0	0	2	2	
Marco 2											
	3	3	3	3	3	1	1	1	1	4	
Page out											
	2	0							1	3	2
									0	1	5

## Examen de Sistemas Operativos Avanzados

## Parte Práctica

Apellidos, nombre: \_\_\_\_\_ DNI: \_\_\_\_\_

## Ejercicio 1

Sea un computador que gestiona un esquema de gestión de memoria que soporta paginación pura, con direccionamiento a nivel de byte y las siguientes características:

- Una dirección virtual está formada por un campo de 16 bits para el número de página y 16 bits para el desplazamiento.
- Una dirección física está formada por 8 bits para el marco de página y 16 bits para el desplazamiento.
- Cada entrada de la tabla de páginas contiene, además del marco de página, 8 bits de estado de los cuales se usan 6, los bits P, S, D, R, W y X. Cuando están activos, señalan que la página está presente, está compartida (shared), ha sido escrita (dirty) ha sido referenciada, está permitida la escritura (write) y si es ejecutable.

1. Con todo ello, se pide:

(a) ¿Cantidad máxima de memoria física que el computador puede direccionar?

$$8+16 = 24 \text{ bits, } 2^{24} \text{ direcciones} = 16\text{Mb}$$

(b) ¿Tamaño de una página de memoria?

16 bits para el desplazamiento dentro de la página:  $(2^{16}) \rightarrow 64\text{Kb}$

(c) ¿Cuál es el tamaño del mayor proceso que pueda cargarse en memoria?

Capacidad de direccionamiento virtual ( $2^{32}$ )

(d) ¿Tamaño en bytes de la tabla de páginas de un proceso?

$2^{16}$  entradas, cada entrada 2 bytes, la tabla ocupa  $2^{17}$  bytes

(0,5 puntos)

En un momento dado, la tabla de páginas de un proceso contiene la siguiente información:

Página	PSDRWXxx	Marco
3	011000	0x01
4	110100	0x55
5	101001	0x03
6	011000	0x27
7	101010	0x27

2. ¿Comparte el proceso memoria con otro proceso?, en caso afirmativo indique las direcciones virtuales y físicas.

(0,2 puntos)

Si, la página 4 está presente y compartida.

Dir. Virtual: 0004 XXXX → Dir. Física: 55 XXXX

3. Realice la traducción de las siguientes direcciones virtuales. Si no es posible la traducción indique la causa.

(0,3 puntos)

Dirección Virtual	Dirección física
0x0001ABCD	Error, dirección no válida
0x0003EFAB	Página no presente, fallo de página
0x0005CDEF	03 CDEF
0x0007ABCD	27 ABCD

Al proceso se le asignan 4 marcos de página y durante el arranque se genera la siguiente secuencia de accesos a páginas: 1, 0, 2, 3, 0, 1, 5, 1, 2, 4, 3.

4. Sabiendo que el sistema de memoria virtual realiza paginación bajo demanda y usa un algoritmo de reemplazo FIFO con segunda oportunidad, indique la secuencia de accesos y cuantos fallos de página se producen.

(1 punto)

\*: Candidata a salir

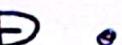
R: Página referenciada

Cad refs	1	0	2	3	0	1	5	1	2	4	3
Page Fault?	sí	sí	sí	sí	no	no	si	si	no	si	sí
Marco 0	1	1	1	1	1	1	5	5	5	5	5
Marco 1	x	0	0	0	0	0	*	*	*	*	*
Marco 2	x	x	2	2	2	2	2	2	2	2	3
Marco 3	x	x	x	x	x	x	x	x	x	x	x
Page out		x	x	x	x	x	x	x	x	x	x

SOA(10)

..

Teoría(11)



Unidad 1. pdf(12)

Unidad 2. pdf(13)



Hardlink

stuff(14)

UD1. pdf(15)

UD2. pdf(13)

UD3. pdf(16)

UD4. pdf(17)

Nodo-i	10	11	12	13	14	15	16	17
tipo	DIR	DIR	SLNK	REG	DIR	REG	REG	REG
Galaxias	3	3	1	2	2	1	4	1
Tamaño	1024	1024	14	1436	1024	456	348	3000
Ptn. dir(2)	10	11	12	13	15	16	17	18
Ptn. dir(2)	-	-	-	14	-	-	14	19
Ptn. Ind. Simp	-	-	-	-	1	-	-	20
Miscel.	-	-	-	2	1	-	-	-

### Bloque de datos

BD	10	11	12	13	14	15	16	17	18
cont.	10 -- Teoria Unidad1.pdf	10 00 stuff 11	11 10 14	Teoria/UD1.pdf	contenido del i-modo	contenido del i-modo	14 11	contenido del i-modo	contenido del i-modo
	11 12	12 13	13 14				15	16	17
	12 13	13 14	14 15						
	13 14	14 15	15 16						
	14 15	15 16	16 17						
	15 16	16 17	17 18						
	16 17	17 18	18 19						
	17 18	18 19	19 20						
	18 19	19 20	20 21						
contenido del i-modo	puntero Bloque	contenido del i-modo							
17	21	17							

## Examen de Sistemas Operativos Avanzados

## Parte Práctica

Apellidos, nombre: \_\_\_\_\_ DNI: \_\_\_\_\_

## Ejercicio 2:

Sea una partición formateada con un sistema de archivos de tipo UNIX que presenta la siguiente organización:

- Bloque de autoarranque, tamaño 512 bytes
- Superbloque que contiene la descripción del sistema de archivos.
  - Tamaño del bloque de datos: 1 KB.  $512 = 1024$
  - Número de nodos índice:  $1.048.576 = 2^{20}$
- Lista de nodos índice, en la que cada entrada ocupa 32Bytes y tiene la siguiente estructura:
  - Tipo de archivo: regular (REG), directorio (DIR), enlace simbólico (SLNK) y tubería (FIFO).
  - Contador de enlaces duros.
  - Tamaño de archivo en bytes.
  - 2 punteros directos.
  - 1 puntero indirecto simple.
  - Otros atributos tales como la fecha de modificación, permisos, id. de dispositivo, etc.
- Las referencias a bloques de datos se realizan mediante punteros de 32 bits (4 bytes).
- Las entradas de un directorio constan de: nombre de archivo y nodo índice.

Sabiendo que el nodo-i número 10 representa el directorio actual "/SOA" y que la ejecución de la orden "ls -al" muestra la siguiente salida por pantalla:

```
[sauron@Mordor ~] $ ls -la
total 16
drwxr-xr-x 3 sauron root 1024 ene 01 12:00 .
drwxr-xr-x 24 sauron root 1024 ene 01 12:00 ..
drwxr-xr-x 3 sauron root 1024 ene 01 12:05 Teoria
lrwxrwxrwx 1 sauron root 14 ene 01 12:15 unidad1.pdf->Teoria/UD1.pdf    enlace simbólico
-rw-r--r-- 2 sauron root 1436 ene 01 12:20 unidad2.pdf

[sauron@Mordor /] $ ls -la Teoria
total 28
drwxr-xr-x 3 sauron root 1024 ene 01 12:10 .
drwxr-xr-x 3 sauron root 1024 ene 01 12:10 ..
drwxr-xr-x 2 sauron root 1024 ene 01 12:10 stuff
-rw-r--r-- 1 sauron root 456 ene 01 12:00 UD1.pdf
-rw-r--r-- 2 sauron root 1436 ene 01 12:00 UD2.pdf
-rw-r--r-- 1 sauron root 348 ene 01 12:00 UD3.pdf
-rw-r--r-- 1 sauron root 3000 ene 01 12:00 UD4.pdf

[sauron@Mordor /] $ ls -la Teoria/stuff
total 8
drwxr-xr-x 2 sauron root 1024 ene 01 12:30 .
drwxr-xr-x 3 sauron root 1024 ene 01 12:30 ..
```

Se pide:

- Sabiendo que el espacio de disco asignado a bloques de datos es de 1 Gbyte ( $2^{30}$ ), ¿cuál es el tamaño máximo que puede tener un fichero en este sistema?

(0,5 puntos)

Cada entrada de puntero indirecto puede almacenar:

$$(\text{Tamaño bloque})/(\text{tamaño puntero}) \rightarrow 1\text{Kb}/4 = 256 \text{ punteros a bloques de datos}$$

Un inode puede definir  $2+256=258$  bloques de datos (2 punteros directos + uno indirecto)

El tamaño máximo de un fichero será de 258 bloques de datos  $\rightarrow 258\text{Kb}$

- ¿Cuántos ficheros de tamaño máximo se pueden almacenar en el sistema de ficheros?

(0,5 puntos)

Cada fichero de tamaño máximo necesita  $258+1$  bloques de datos, incluyendo el bloque con los punteros.

La cantidad máxima de ficheros sería (Número de bloques) / (bloques por fichero)  
Número de bloques de datos =  $2^{30} / 2^{10} = 2^{20}$

Atendiendo a la cantidad de bloques de datos:  $2^{20} / 259 = 4.048$  ficheros de tamaño máximo.

- Dibuje la estructura jerárquica del directorio de trabajo, identificando nombres de cada archivo y su nodo-i. A continuación, rellene las siguientes tablas:

(1 punto)

INODE	PATH
10	/SOA
10	---- .
2	---- ..
12	---- unidad1.pdf -> Teoria/UD1.pdf
13	---- unidad2.pdf
11	---- Teoria
11	---- .
10	---- ..
15	---- UD1.pdf
13	---- UD2.pdf
16	---- UD3.pdf
17	---- UD4.pdf
14	---- stuff
14	---- .
11	---- ..

## Examen de Sistemas Operativos Avanzados

## Parte Práctica

Apellidos, nombre: \_\_\_\_\_ DNI: \_\_\_\_\_

Nodo-i	10	11	12	13	14
Tipo	DIR	DIR	SLNK	REG	DIR
Enlaces duros	3	3	1	2	2
Tamaño (bytes)	1024	1024	14	1436	1024
PtrDir1	0	1	2	3	5
PtrDir2	NULL	NULL	NULL	4	NULL
PtrInd1	NULL	NULL	NULL	NULL	NULL

Nodo-i	15	16	17		
Tipo	REG	REG	REG		
Enlaces duros	1	1	1		
Tamaño (bytes)	456	348	3000		
PtrDir1	6	7	8		
PtrDir2	NULL	NULL	9		
PtrInd1	NULL	NULL	10		

Bloque de datos	0	1	2	3		
Contenido	. .. Teoría unidad1.pdf unidad2.pdf	10 2 11 12 13	. .. stuff UD1.pdf UD2.pdf UD3.pdf UD4.pdf	11 10 14 15 13 16 17	Teoria/UD1.pdf	Contenido fichero unidad2.pdf (primer kb)
Bloque de datos	4	5	6	7		
Contenido	Contenido fichero unidad2.pdf (último kb)	.	14 11	Contenido fichero UD1.pdf	Contenido fichero UD3.pdf	
Bloque de datos	8	9	10	11		
Contenido	Contenido fichero UD4.pdf (primer kb)	Contenido fichero UD4.pdf (segundo kb)	(Lista de punteros) 11	Contenido fichero UD4.pdf (último kb)		
Bloque de datos						
Contenido						

## Examen de Sistemas Operativos Avanzados

### Parte Práctica. Memoria virtual

Apellidos, nombre: \_\_\_\_\_ DNI: \_\_\_\_\_

Sea un computador que gestiona un esquema de gestión de memoria que soporta paginación pura, con direccionamiento a nivel de byte, y las siguientes características:

- El tamaño máximo de la memoria física es de 1 MByte.
- El tamaño máximo de la memoria virtual es de 4 GBytes.
- El tamaño de página y de marco de página es de 256 bytes.
- Los bloques de control de procesos (BCP) se ubican en el marco de página 0.
- Las tablas de páginas de los procesos (TDP) se ubican en el marco de página 1.

Cada bloque BCP ocupa ocho bytes según la siguiente estructura:

```
struct BCP {  
    unsigned char PID; /* 1 byte. Id. proceso, siempre distinto de 0x00 */  
    short int RPBTP; /* 3 bytes. Dirección de inicio de la TDP */  
    unsigned char RLTP; /* 3 bytes. N° de páginas del proceso */  
    unsigned char extra; /* 1 byte. Información extra */
```

Cada entrada en la tabla de páginas tiene el siguiente formato:

- Los bits de mayor peso, PRMC, indican respectivamente si una página está presente, y/o ha sido referenciada, modificada o compartida en memoria principal.
- Los doce bits de menor peso guardan el número de marco de memoria principal.

Según los datos de este sistema, responda a las siguientes preguntas:

1. Calcule el formato de una dirección de memoria virtual y de una dirección de memoria principal. (1 punto).

Respuesta:

Dirección de memoria virtual: Página (24 bits) — Desplazamiento (8 bits)

Dirección de memoria principal: Marco (12 bits) — Desplazamiento (8 bits)

2. A continuación, se muestra el mapa de memoria principal en el instante  $t$ . Identifique los procesos que hay en el sistema y muestre la información que mantiene cada BCP y TDP. (3 puntos).

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000	03	00	01	18	00	00	09	xx	02	00	01	10	00	00	04	xx
0x0001	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000x	...															
0x0010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0011	80	0A	80	0B	90	0C	80	0D	90	0C	80	02	80	03	80	04
0x0012	80	05	80	06	80	07	80	08	80	09	00	00	00	00	00	00
0x001x	...															

**Respuesta:**

Según muestra el marco 0, existen dos procesos en el sistema con las siguientes características:

PID	RBTP	RLTP	Extra
0x03	0x000118	0x000009	xx
0x02	0x000110	0x000004	xx

Observando el marco 1, se deduce que la TDP del proceso de pid 2 es:

PRMC(base2) (4 bits)	Marco(base16) (12 bits)
1000	0x00A
1000	0x00B
1001	0x00C
1000	0x00D

Del mismo modo, se deduce que la TDP del proceso de pid 3 es:

PRMC(base2) (4 bits)	Marco(base16) (12 bits)
1001	0x00C
1000	0x002
1000	0x003
1000	0x004
1000	0x005
1000	0x006
1000	0x007
1000	0x008
1000	0x009

3. Según los resultados del apartado anterior, indique si existe algún marco compartido por ambos procesos. Razoné su respuesta (1 punto).

**Respuesta:**

Tanto el proceso de pid 2, como el proceso de pid 3, comparten probablemente su código, ya que se trata de las páginas de solo lectura, pagina 2 para el proceso de pid=2 y pagina 1 para el pid=3, cuyo marco es el 0x00C.

4. Tras un periodo de tiempo, el proceso P2 realiza las referencias a memoria virtual: 0x00000000, 0x00000200, 0x00000400. Realice la traducción de dichas direcciones, considerando que la tabla de páginas en ese instante es la siguiente: (2 puntos).

Página	PMRC(base 2)	Marco(base 16)
0	1101	0x00C
1	1100	0x007
2	0001	0x0A0
3	1100	0x009

**Respuesta:**

a) 0x00000000.

- 1) Comprueba si la dirección virtual está dentro de los límites considerando el valor del RLTP = 0x4.
- 2) Accede a la tabla de páginas para localizar la entrada correspondiente a la página 0x000000.
- 3) Comprueba el valor del bit de presencia (P=1).
- 4) Genera la dirección física, a partir de la tabla de páginas, concatenando el número de marco con el desplazamiento de la dirección virtual.

La dirección virtual 0x00000000 se corresponde con la dirección física 0x00C00.

b) 0x00000200.

- 1) Comprueba si la dirección virtual está dentro de los límites del RLTP = 0x4.
- 2) Accede a la tabla de páginas para localizar la entrada correspondiente a la página 0x000002.
- 3) Comprueba que el bit de presencia (P=0) muestra que la página no está cargada en memoria principal.
- 4) Se produce una excepción de fallo de página.

c) 0x00000400.

- 1) Comprueba si la dirección virtual está dentro de los límites del RLTP = 0x4.
- 2) Se produce una excepción por exceder los límites de memoria del proceso.

5. La gestión de la memoria virtual implementa un algoritmo FIFO de segunda oportunidad, con 3 marcos (A, B y C) y política de reemplazo local. Indique los aciertos y fallos de página que se generan, suponiendo que el proceso P4 acaba de iniciarse y tiene la siguiente cadena de referencias: (3 puntos).
- 0x005000, 0x002002, 0x000000, 0x004004, 0x003000, 0x004004, 0x000000, 0x001001,  
0x004004, 0x002002

Respuesta:

Refs.	005000	002002	000000	004004	003000	004004	000000	001001	004004	002002
A/F	F	F	F	F	F	A	A	F	A	F
MarcoA	+5000	+5000	+5000	+4004	+4004	+4004	+4004	*-4004	*+4004	-4004
MarcoB		+2002	+2002	*-2002	+3000	+3000	+3000	-3000	-3000	+2002
MarcoC			+0000	-0000	*-0000	*-0000	*+0000	+1001	+1001	*+1001

**Examen de Sistemas Operativos Avanzados**  
**Parte Práctica. Sistema de archivos**  
**(Elija resolver un único ejercicio sobre el sistema de archivos)**

Apellidos, nombre: \_\_\_\_\_ DNI: \_\_\_\_\_

Sea una partición formateada con un sistema de archivos de tipo UNIX que presenta la siguiente organización:

- Bloque de autoarranque.
- Superbloque que contiene la descripción del sistema de archivos.
  - Tamaño del bloque de datos: 1 KB.
  - Tamaño de un directorio: 1 KB.
  - Número de nodos índice:  $1.048.576 = 2^{20}$ .
  - Mapa de bits de nodos índice libres ocupa 128 bloques.
  - Mapa de bits de los bloques libres ocupa 2.048 bloques.
  - El bit a 1 indica libre, en ambos mapas de bits.
- Lista de nodos índice, en la que cada entrada ocupa 32 B y tiene la siguiente estructura:
  - Tipo de archivo: regular (REG), directorio (DIR), enlace (SLNK) y tubería (FIFO).
  - Contador de enlaces.
  - Tamaño de archivo en bytes.
  - 2 punteros directos.
  - 1 puntero indirecto simple.
  - 1 puntero indirecto doble.
  - Otros atributos tales como la fecha de modificación, permisos, id. de dispositivo, etc.
- Un puntero referencia un bloque de datos con 32 bits (4 bytes).
- Los nodos índice y los bloques de datos se asignan de forma ascendente, a medida que se crean los archivos.
- Las entradas de un directorio constan de: nombre de archivo y nodo índice.

1. Dibuje el árbol con la estructura de dicho directorio, identificando el nombre de cada archivo y su nodo-i. (2 puntos):

Cuadro 5: Información del superbloque.

Mapa de bits de nodos-i libres:	--00 0000 1111 1111 1...
Mapa de bits de bloques libres:	0000 0000 0000 0111 1...

Cuadro 6: Lista de nodos índice y bloques de datos.

Nodo-i	2	3	4	5	6	7
Tipo	DIR	DIR	DIR	REG	REG	DIR
Nº enlaces	4	3	2	1	3	2
Tamaño (bytes)	1024	1024	1024	2000	6144	1024
PtrDir1	0	1	2	3	5	12
PtrDir2	NULL	NULL	NULL	4	6	NULL
PtrIndSim1	NULL	NULL	NULL	NULL	7	NULL
PtrIndDob1	NULL	NULL	NULL	NULL	NULL	NULL

0	1	2	3	4
. 2 .. 2 apuntes 3 normas.pdf 6 practical 7	. 3 .. 2 normasap.pdf 6 temal 4	. 4 .. 3 ej1.pdf 5	Contenido de nodo-i 5	Cont. contenido de nodo-i 5
5	6	7	8	9
Contenido de nodo-i 6	Cont. contenido de nodo-i 6	8, 9, 10, 11	Cont. contenido de nodo-i 6	Cont. contenido de nodo-i 6
10	11	12	13	14
Cont. contenido de nodo-i 6	Cont. contenido de i-nodo 6	. 7 .. 2 normaspr.pdf 6		

Respuesta:

/ (i-nodo: 2)  
..... (.....)

Respuesta:

/ (2)  
- apuntes (3)  
- . - normasap.pdf (6)  
- . - temal (4)  
- . - ej1.pdf (5)  
- normas.pdf (6)  
- practical (7)  
- . - normaspr.pdf (6)

2. Verifique la consistencia de dicho sistema de archivos, tanto a nivel de bloque como de archivo, e indique si contiene algún error. Justifique su respuesta. (2 puntos)

**Respuesta:**

El sistema de archivos es consistente tras realizar el análisis de los mapas de bits correspondientes a los nodos índice y bloques de datos; así como de la lista de nodos-i y de los bloques de información.

3. Sabiendo que desde el directorio raíz se ejecutan las siguientes órdenes:

```
$ cat "Hola mundo" > hola.txt
$ mkdir practica2
$ cd practica2
$ ln -s ../../normas.pdf normas.slnk
$ rm ../../normas.pdf
```

Muestre únicamente los cambios en el superbloque, lista de nodos índice y bloques de datos que se producirían tras la ejecución de las órdenes del apartado anterior. (4 puntos)

Cuadro 7: Información del superbloque modificada.

Mapa de bits de nodos-i libres:	
Mapa de bits de bloques libres:	

Cuadro 8: Lista de nodos índice modificados y bloques de datos modificados.

Nodo-i					
Tipo					
Nº enlaces					
Tamaño (bytes)					
PtrDir1					
PtrDir2					
PtrIndSim1					
PtrIndDobl1					


**Respuesta:**

Cuadro 9: Información del superbloque.

Mapa de bits de nodos-i libres:	--00 0000 0001 1111 11...
Mapa de bits de bloques libres:	0000 0000 0000 0000 11...

Cuadro 10: Lista de nodos índice y bloques de datos.

Nodo-i	2	6	8	9	10
Tipo	DIR	REG	REG	DIR	SLNK
Nº enlaces	5	2	1	2	1
Tamaño (bytes)	1024	6144	11	1024	14
PtrDir1	0	5	13	14	15
PtrDir2	NULL	6	NULL	NULL	NULL
PtrIndSim1	NULL	7	NULL	NULL	NULL
PtrIndDob1	NULL	NULL	NULL	NULL	NULL

0	13	14	15
. 2 .. 2 apuntes 3  practical 7 hola.txt 8 practica2 9	Hola mundo	. 9 .. 2 normas.slnk 10	./normas.pdf

4. Calcule el número de bloques de punteros y de datos que necesitaría el archivo de tamaño máximo en esta partición. Asimismo indique la cantidad de bytes que podría almacenar dicho archivo. (2 puntos)

**Respuesta:**

Según la estructura indicada para este sistema de archivos:

- Nº de punteros que contiene un bloque es:  $\frac{2^{10} \text{bytes}/\text{bloque}}{2^2 \text{bytes}/\text{ptr}} = 2^8 \text{ptrs}/\text{bloque}$ .

Se sabe que desde un nodo-i se pueden referenciar:

- 2 ptrs. directos = 2 bloques de datos.
- 1 ptr. indirecto simple = 1 bloque de punteros directos +  $2^8$  bloques de datos.
- 1 ptr. ind. doble = 1 blq. ptrs. inds. simp. +  $2^8$  blqs. ptrs. dirs. +  $2^8 * 2^8$  blqs. datos.

Por tanto:

- El número de bloques de punteros es:  $1 + 1 + 256 = 258$  bloques de punteros.
- El número de bloques de datos es:  $2 + 256 + 4096 = 65.794$  bloques de datos.
- Tamaño máx. de archivo:  $65.794$  bloques de datos \* 1.024 bytes/bloque = 64,25 MB.

**Examen de Sistemas Operativos Avanzados**  
**Parte Práctica. Sistema de archivos**

Apellidos, nombre: \_\_\_\_\_ DNI: \_\_\_\_\_

Se tiene un sistema de archivos basado en FAT llamado Microfat. En este sistema la FAT tiene 256 entradas, y cada una de ellas es de 8 bit. Para cada entrada de la FAT, **0x00** indica que el bloque correspondiente no está siendo utilizado, **0xFF** significa que ese es el **último bloque del archivo** y cualquier otro valor indica el siguiente número de bloque que compone el archivo.

En este sistema de archivos los bloques son de 32 bytes.

En Microfat sólo hay un directorio raíz. Cada entrada de directorio contiene un nombre de 8 bytes, una extensión de 3 bytes, un byte indicando el número total de bloques del archivo y otro byte que indica cuál es el primer bloque.

1. ¿Cuál es el tamaño del archivo más grande que se puede almacenar en Microfat, considerando únicamente la limitación impuesta por el diseño de la FAT?

(2 puntos)

**RESPUESTA:**

Si cada entrada de la FAT es de 8 bit, se pueden referenciar hasta 256 bloques. Cada bloque es de 32 bytes, así es que el tamaño más grande referenciable por la FAT sería  $2^8 * 32 = 8KiB$ . Habría que restar a este valor 64 bytes, ya que el bloque 0 y el 256 no se pueden utilizar de forma efectiva, al ser esos números dos delimitadores especiales de la FAT.

2. Debido a un fallo hardware, el sistema Microfat existente quedó dañado. Concretamente se perdió el contenido de la FAT, aunque se pudo rescatar el contenido de los bloques de datos. A continuación se muestra el volcado hexadecimal de los datos rescatados. Los bloques de datos se numeran comenzando por 1.

Bq1	00000000	0e 6c 61 20 63 6f 6e 63	6c 75 69 61 6e 20 73 61	.la concluian sa
	00000010	79 6f 20 64 65 20 76 65	6c 61 72 74 65 20 28 32	yo de velarte (2
Bq2	00000020	06 20 66 6c 61 63 6f 20	79 20 67 61 6c 67 6f 20	. flaco y galgo
	00000030	63 6f 72 72 65 64 6f 72	2e 20 55 6e 61 20 6f 6c	corredor. Una ol
Bq3	00000040	04 75 6e 20 68 69 64 61	6c 67 6f 20 64 65 20 6c	.un hidalgo de l
	00000050	6f 73 20 64 65 20 6c 61	6e 7a 61 20 65 6e 20 61	os de lanza en a
Bq4	00000060	05 73 74 69 6c 6c 65 72	6f 2c 20 61 64 61 72 67	.stillero, adarg
	00000070	61 20 61 6e 74 69 67 75	61 2c 20 72 6f 63 69 6e	. antigua, rocin
Bq5	00000080	02 79 6f 20 6e 6f 6d 62	72 65 20 6e 6f 20 71 75	.yo nombre no qu
	00000090	69 65 72 6f 20 61 63 6f	72 64 61 72 6d 65 2c 20	iero acordarme,
Bq6	000000a0	0b 61 6c 6f 6d 69 6e 6f	20 64 65 20 61 6e 61 64	.alemico de anad
	000000b0	69 64 75 72 61 20 6c 6f	73 20 64 6f 6d 69 6e 20	idura los domin
Bq7	000000c0	07 6c 61 20 64 65 20 61	6c 67 6f 20 6d 61 73 20	.la de algo mas
	000000d0	76 61 63 61 20 71 75 65	20 63 61 72 6e 65 72 6f	vaca que carnero
Bq8	000000e0	08 2c 20 73 61 6c 70 69	63 6f 6e 20 28 31 29 20	., salpicon (1)
	000000f0	6c 61 73 20 64 61 73 20	6e 6f 63 68 65 73 2c 20	las mas noches,
Bq9	00000100	0d 64 65 20 73 75 20 68	61 63 69 65 6e 64 61 2e	.de su hacienda.
	00000110	20 20 45 6c 20 72 65 73	74 6f 20 64 65 20 65 6c	El resto de el
BqA	00000120	09 64 75 65 6c 6f 73 20	79 20 71 75 65 62 72 61	.duelos y quebra
	00000130	6e 74 6f 73 20 6c 6f 73	20 73 61 62 61 64 6f 73	ntos los sabados
BqB	00000140	0a 2c 20 6c 65 6e 74 65	6a 61 73 20 6c 6f 73 20	., lentejas los
	00000150	76 69 65 72 6e 65 73 2c	20 61 6c 67 75 6e 20 70	viernes, algun p
BqC	00000160	01 45 6e 20 75 6e 20 6c	75 67 61 72 20 64 65 20	.En un lugar de
	00000170	6c 61 20 4d 61 6e 63 68	61 2c 20 64 65 20 63 75	la Mancha, de cul
BqD	00000180	13 33 29 20 64 65 20 6c	6f 20 6d 61 73 20 66 69	.3) de lo mas fil
	00000190	6e 6f 2e 20 20 20 20 20	20 20 20 20 20 20 20 20	no.
BqE	000001a0	0c 67 6f 73 2c 20 63 6f	6e 73 75 6d 69 61 6e 20	.gos, consumian
	000001b0	6c 61 73 20 74 72 65 73	20 70 61 72 74 65 73 20	las tres partes
BqF	000001c0	10 69 65 73 74 61 73 2c	20 63 6f 6e 20 73 75 73	.iestas, con sus
	000001d0	20 70 61 6e 74 75 66 6c	6f 73 20 64 65 20 6c 6f	pantuflas de lo
Bq10	000001e0	11 20 6d 69 73 6d 6f 2c	20 79 20 6c 6f 73 20 64	. mismo, y los d

```

000001f0 69 61 73 20 64 65 20 65 6e 74 72 65 20 73 65 6d |ias de entre sem|
Eq11 00000200 0f 29 2c 20 63 61 6c 7a 61 73 20 64 65 20 76 65 |.), calzas de ve|
00000210 6c 6c 75 64 6f 20 70 61 72 61 20 6c 61 73 20 66 |lludo para las f|
Eq12 00000220 12 61 6e 61 20 73 65 20 68 6f 6e 72 61 62 61 20 |.ana se honraba |
00000230 63 6f 6e 20 73 75 20 76 65 6c 6c 6f 72 69 20 28 |con su vellori (|
Eq13 00000240 03 6e 6f 20 68 61 63 65 20 6d 75 63 68 6f 20 74 |no hace mucho ti|
00000250 69 65 6d 70 6f 20 71 75 65 20 76 69 76 69 61 20 |empo que vivia |
Eq14 00000260

```

Se sabe que el archivo estaba almacenado en un formato propietario llamado Minidoc. Minidoc divide el contenido de sus documentos en segmentos de 32 bytes. Obsérvese que cada bloque de datos contiene en el primer byte su número de orden original, comenzando por 0x01, y le sigue la información del archivo en los siguientes bytes. Con esta información, se pide que reconstruya el contenido de una tabla FAT que permita recuperar el archivo completo, así como una posible entrada de directorio con un nombre arbitrario. Suponga una FAT inicializada tal como se muestra en el siguiente ejemplo:

index	value								
0	FF	5	00	10	00	15	00	20	00
1	00	6	00	11	00	16	00	21	00
2	00	7	00	12	00	17	00	22	00
3	00	8	00	13	00	18	00	23	00
4	00	9	00	14	00	19	00	24	00

(8 puntos)

**RESPUESTA:**

Como los segmentos de Minidoc son del mismo tamaño que los bloques de datos, todos los bloques de datos comenzarán por un número que indica el orden del segmento dentro del archivo. Como los bloques son de 32 bytes (0x20), en el offset 0 comenzará el bloque 1, en el offset 0x20 el bloque 2, en el 0x40 el bloque 3, y en general en el offset n comenzará el bloque  $n/20 + 1$  (en hexadecimal). Buscamos por lo tanto un bloque que comience por 1, que se encuentra en el offset 0x160 (bloque 0x0C). A continuación uno que comience por 2, que se encuentra en el offset 0x80 (bloque 0x05), y así sucesivamente llegamos a la siguiente tabla:

seg#	Offset	Bloque
1	0x160	0x0C
2	0x080	0x05
3	0x240	0x13
4	0x040	0x03
5	0x060	0x04
6	0x020	0x02
7	0x0C0	0x07
8	0x0E0	0x08
9	0x120	0x0A
10	0x140	0x0B
11	0x0A0	0x06
12	0x1A0	0x0E
13	0x100	0x09
14	0x000	0x01
15	0x200	0x11
16	0x1C0	0x0F
17	0x1E0	0x10
18	0x220	0x12
19	0x180	0x0D

La tabla FAT quedaría por lo tanto de la siguiente forma:

index	value								
0	FF	5	13	0A	0B	10	10	14	00
1	11	6	0E	0B	06	10	12	15	00
2	07	7	08	0C	05	11	0F	16	00
3	04	8	0A	0D	FF	12	0D	17	00
4	02	9	01	0E	09	13	03	18	00

La entrada de directorio tendría un nombre de archivo, por ejemplo **qui jots** con extensión **.txt**, que ocupa 19 bloques y cuyo primer bloque sería 0x0C.

## Examen de Sistemas Operativos Avanzados

### Parte Práctica

Apellidos, nombre: \_\_\_\_\_ DNI: \_\_\_\_\_

Sea un computador que gestiona un esquema de gestión de memoria que soporta paginación pura, con direccionamiento a nivel de byte, y las siguientes características:

- El tamaño máximo de la memoria física es de 1 MByte.
- El tamaño máximo de la memoria virtual es de 4 GBytes.
- El tamaño de página y de marco de página es de 256 bytes.
- Los bloques de control de procesos (BCP) se ubican en el marco de página 0.
- Las tablas de páginas de los procesos (TDP) se ubican en el marco de página 1.

Cada bloque BCP ocupa ocho bytes según la siguiente estructura:

```
struct BCP {  
    unsigned char PID; /* 1 byte. Id. del proceso, distinto de 0x00 */  
    short int RPBTP; /* 3 bytes. Dirección de inicio de la TDP */  
    unsigned char RLTP; /* 3 bytes. Nº de páginas del proceso */  
    unsigned char extra;} /* 1 byte. Información extra */
```

Cada entrada en la tabla de páginas tiene el siguiente formato:

- Los bits de mayor peso, PRMC, indican respectivamente si una página está presente, y/o ha sido referenciada, modificada o compartida en memoria principal.
- Los doce bits de menor peso guardan el número de marco de memoria principal.

Cuando se lanza un proceso, el sistema de memoria realiza las siguientes acciones:

1. Crea el correspondiente BCP, en la primera posición libre del marco 0.
2. Comprueba si hay suficientes marcos de página libres para alojar la TDP del proceso, y crea la correspondiente tabla de páginas, en la primera posición libre del marco 1.

Cuando un proceso finaliza, el sistema de memoria realiza de manera ordenada las acciones:

1. Se marca con pid = 0 su entrada del BCP.
2. Se inician a 0 las posiciones de memoria ocupadas por tabla de páginas.

Según los datos de este sistema, responda a las siguientes preguntas:

1. Calcule el formato de una dirección de memoria virtual y de una dirección de memoria principal. (1 punto)  
Respuesta:  
Dirección de memoria virtual: Página (24 bits) — Desplazamiento (8 bits)  
Dirección de memoria principal: Marco (12 bits) — Desplazamiento (8 bits)
2. Suponiendo el instante inicial, en que toda la memoria principal está a 0, indique las posiciones de memoria modificadas en los marcos de página 0 y 1, tras la ejecución de los eventos siguientes:

Se lanza el proceso P1 (pid = 1), que requiere de 2.040 bytes de memoria. (1,5 puntos)  
 Respuesta:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000																
0x0001																
...																
0x0010																
0x0011																
...																

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Marco 0:	0x0000	01	00	01	00	00	00	08	xx	00	00	00	00	00	00	00
Marco 1:	0x0010	80	02	80	03	80	04	80	05	80	06	80	07	80	08	80

3. Se lanza el proceso P2 (pid = 2), que requiere de 1.020 bytes de memoria. (1,5 puntos)

Respuesta:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000																
0x0001																
0x0002																
...																
0x0010																
0x0011																
0x0012																
...																

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Marco 0:	0x0000	01	00	01	00	00	00	08	xx	02	00	01	10	00	00	04
Marco 1:	0x0010	80	02	80	03	80	04	80	05	80	06	80	07	80	08	80
	0x0011	80	0A	80	0B	80	0C	80	0D	00	00	00	00	00	00	00

4. Finaliza P1 (pid=1) y se lanza P3 (pid = 3), de 2.060 bytes, que comparte su página 0 con la página 2 del proceso P2. (3 puntos)

Respuesta:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x0000																
0x0001																
0x0002																
...																
0x0010																
0x0011																
0x0012																
...																

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Marco 0:	03	00	01	18	00	00	09	xx	02	00	01	10	00	00	04	xx

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Marco 1:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
	0x0010	80	0A	80	0B	90	0C	80	0D	90	0C	80	02	80	03	80
	0x0011	80	05	80	06	80	07	80	08	80	09	00	00	00	00	00

5. Tras un periodo de tiempo, el proceso P2 realiza las referencias a memoria virtual: 0x00000000, 0x00000200, 0x00000400. Realice la traducción de dichas direcciones, considerando que la tabla de páginas en ese instante es la siguiente: (1 puntos)

Página	PMRC(base 2)	Marco(base 16)
0	1101	0x00C
1	1100	0x007
2	0001	0x0A0
3	1100	0x009

Respuesta:

a) 0x00000000.

- 1) Comprueba si la dirección virtual está dentro de los límites considerando el valor del RLTP = 0x4.
- 2) Accede a la tabla de páginas para localizar la entrada correspondiente a la página 0x000000.
- 3) Comprueba el valor del bit de presencia (P=1).
- 4) Genera la dirección física, a partir de la tabla de páginas, concatenado el número de marco con el desplazamiento de la dirección virtual.

La dirección virtual 0x00000000 se corresponde con la dirección física 0x00C00.

b) 0x00000200.

- 1) Comprueba si la dirección virtual está dentro de los límites del RLTP = 0x4.
- 2) Accede a la tabla de páginas para localizar la entrada correspondiente a la página 0x000002.
- 3) Comprueba que el bit de presencia (P=0) muestra que la página no está cargada en memoria principal.
- 4) Se produce una excepción de fallo de página.

c) 0x00000400.

- 1) Comprueba si la dirección virtual está dentro de los límites del RLTP = 0x4.
- 2) Se produce una excepción por exceder los límites de memoria del proceso.

6. La gestión de la memoria virtual implementa un algoritmo FIFO de segunda oportunidad, con 3 marcos (A, B y C) y política de reemplazo local. Indique los aciertos y fallos de página que se generan, suponiendo que el proceso P4 acaba de iniciarse y tiene la siguiente cadena de referencias: (2 puntos)

0x005000, 0x002002, 0x000000, 0x004004, 0x003000, 0x004004, 0x000000, 0x001001, 0x004004, 0x002002

Respuesta:

Refs.	005000	002002	000000	004004	003000	004004	000000	001001	004004	002002
A/F	F	F	F	F	F	A	A	F	A	F
MarcoA	+5000	+5000	+5000	+4004	+4004	+4004	+4004	*-4004	*+4004	-4004
MarcoB		+2002	+2002	*-2002	+3000	+3000	+3000	-3000	-3000	+2002
MarcoC			+0000	-0000	*-0000	*-0000	*+0000	+1001	+1001	*+1001

• • 7  
• (11)

normas.pdf (15)

apuntes 12

Práctica L 16

• 16

• • 11

normaspr.pdf (15)

• • 12

• • 11

normasap.pdf (15)

tema L → ejer L.pdf 14  
• 13  
• 12

Modo-P	11	12	13	14	15	16	
Tipo	Din	Din	Din	Reg	Reg	Din	
Nº enlaces	4	3	2	1	3	2	
Tamaño	1KB	1KB	1KB	2000	6144	1KB	
Pt Din L	11	12	13	14	16	23	
Pt Din 2	-	-	-	15	17	-	
Pt Indi L	-	-	-	-	18	-	
Pt Indi 2 doble	-	-	-	-	-	-	

11	12	13	14	15	16	17	18	19	20	21	22
• 11	• 12	• 13	cont	cont	cont	cont	19				
• 17	• 11	• 12	i-mod	mod	15	15	20				
normas.pdf 15	normasap.pdf 15	ejer L.pdf 14	14	14	15	15	21				
apuntes 12	temas 13						22				
Práctica L 16							Cont				

23
• 16
• 11
normas 15

~~2048~~

$$6144 - 2048 = \frac{4096}{1024} = 4$$

## Examen de Sistemas Operativos Avanzados

### Parte Práctica

Apellidos, nombre: \_\_\_\_\_ DNI: \_\_\_\_\_

Sea una partición formateada con un sistema de archivos de tipo UNIX que presenta la siguiente organización:

- Bloque de autoarranque.
- Superbloque que contiene la descripción del sistema de archivos.
  - Tamaño del bloque de datos: 1 KB.
  - Tamaño de un directorio: 1 KB.
  - Número de nodos índice:  $1.048.576 = 2^{20}$ .
  - Mapa de bits de nodos índice libres ocupa 128 bloques.
  - Mapa de bits de los bloques libres ocupa 2.048 bloques.
  - El bit a 1 indica libre, en ambos mapas de bits.
- **Lista de nodos índice**, en la que cada entrada ocupa 32 B y tiene la siguiente estructura:
  - Tipo de archivo: regular (REG), directorio (DIR), enlace (SLNK) y tubería (FIFO).
  - Contador de enlaces.
  - Tamaño de archivo en bytes.
  - 2 punteros directos.
  - 1 puntero indirecto simple.
  - 1 puntero indirecto doble.
  - Otros atributos tales como la fecha de modificación, permisos, id. de dispositivo, etc.
- Un puntero referencia un bloque de datos con 32 bits (4 bytes).
- Los nodos índice y los bloques de datos se asignan de forma ascendente, a medida que se crean los archivos.
- Las entradas de un directorio constan de: nombre de archivo y nodo índice.

Sabiendo que el nodo-i 11 es el directorio actual, el nodo-i 15 tiene asignados los nombres ./normas.pdf, ./apuntes/normasap.pdf y ./practica1/normaspr.pdf, y que la ejecución de la orden ls -Rgoz<sup>2</sup> muestra la siguiente salida por pantalla del directorio de trabajo:

```
drwxr-xr-x 1024 17 ene 08:31 apuntes
-rw-r--r-- 6144 17 ene 08:37 normas.pdf
drwxr-xr-x 1024 17 ene 09:37 practica1
./apuntes
-rw-r--r-- 6144 17 ene 08:37 normasap.pdf
drwxr-xr-x 1024 17 ene 08:33 temal
./apuntes/temal
-rw-r--r-- 2000 17 ene 08:34 ej1.pdf
./practica1
-rw-r--r-- 6144 17 ene 08:37 normaspr.pdf
```

1. Dibuje la estructura del directorio de trabajo, identificando nombres de cada archivo y su nodo-i. A continuación rellene las siguientes tablas: (5 puntos):

<sup>2</sup>Listado recursivo mostrando: tipo de archivo, permisos, tamaño en bytes, fecha y nombre del archivo.

Cuadro 5: Información del superbloque.

Mapa de bits de nodos-i libres:	
Mapa de bits de bloques libres:	

Cuadro 6: Lista de nodos índice y bloques de datos.

Nodo-i	11	12	13	14	15
Tipo	DIR				
Nº enlaces					
Tamaño (bytes)					
PtrDir1	11				
PtrDir2	-				
PtrIndSim1	-				
PtrIndDob1	-				
Nodo-i					
Tipo					
Nº enlaces					
Tamaño (bytes)					
PtrDir1					
PtrDir2					
PtrIndSim1					
PtrIndDob1					

Bloque 11 nombre archivo	nodo-i	Bloque 12		
.	11			
..	7			
apuntes	12			
normas.pdf	15			
practical	16			

**Respuesta:**

Cuadro 7: Información del superbloque.

Mapa de bits de nodos-i libres:	--00 0000 0000 0000 0111 1111 1...
Mapa de bits de bloques libres:	0000 0000 0000 0000 0000 0000 1...

Cuadro 8: Lista de nodos índice y bloques de datos.

Nodo-i	11	12	13	14	15	16
Tipo	DIR	DIR	DIR	REG	REG	DIR
Nº enlaces	4	3	2	1	3	2
Tamaño (bytes)	1024	1024	1024	2000	6144	1024
PtrDir1	11	12	13	14	16	23
PtrDir2	NULL	NULL	NULL	15	17	NULL
PtrIndSim1	NULL	NULL	NULL	NULL	18	NULL
PtrIndDob1	NULL	NULL	NULL	NULL	NULL	NULL
PtrIndTrip1	NULL	NULL	NULL	NULL	NULL	NULL

11	12	13	14	15
. 11 .. 7 apuntes 12 normas.pdf 15 practical 16	. 12 .. 11 normasap.pdf 15 temal 13	. 13 .. 12 ej1.pdf 14	Contenido del de nodo-i 14 (ej1.pdf)	archivo
16	17	18	19	20
Contenido del de nodo-i 15 (./normas.pdf...)	archivo (sigue)	19, 20, 21, 22	Continuación del del archivo de nodo-i 15	contenido (sigue)
21	22	23	24	25
Continuación del archivo nodo-i 15 (fin)	contenido	. 16 .. 11 normaspr.pdf 15		

2. Sabiendo que se ejecutan las siguientes órdenes:

```
$ cd ./apuntes/temal
$ rm ej1.pdf
$ cat "Hola mundo" > hola.txt
$ cd ../../practical
$ mkdir practical
$ cd practical
$ ln -s ../../apuntes/temal/ej1.pdf apuntestemal.slnk
```

Muestre únicamente los cambios en el superbloque, lista de nodos índice y bloques de datos que se producirían tras la ejecución de las órdenes del apartado anterior. (3 puntos)

Cuadro 9: Información del superbloque modificada.

Mapa de bits de nodos-i libres:	
Mapa de bits de bloques libres:	

Cuadro 10: Lista de nodos índice modificados y bloques de datos modificados.

Nodo-i					
Tipo					
Nº enlaces					
Tamaño (bytes)					
PtrDir1					
PtrDir2					
PtrIndSim1					
PtrIndDob1					


**Respuesta:**

Cuadro 11: Información del superbloque.

Mapa de bits de nodos-i libres:	--00 0000 0000 0000 0001 1111 11...
Mapa de bits de bloques libres:	0000 0000 0000 0000 0000 01...

11	12	13	14	15
. 11 .. 7 apuntes 12 normas.pdf 15 practical 16	. 12 .. 11 normasap.pdf 15 temal 13	. 13 .. 12 hola.txt 14	Contenido del de nodo-i 14 (hola.txt)	. 17 .. 16 apuntestema1.slnk 18
16	17	18	19	20
Contenido del de nodo-i	archivo 15 (sigue)	19, 20, 21, 22	Continuación del archivo de	del contenido nodo-i 15 (sigue)
21	22	23	24	25
Continuación archivo nodo-i 15 (fin)	del contenido	. 16 .. 11 normaspr.pdf 15 practical 17	../../apuntes/ /temal/ej1.pdf	

Cuadro 12: Lista de nodos índice y bloques de datos.

Nodo-i	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>
Tipo	DIR	DIR	DIR	REG	REG
Nº enlaces	4	3	2	1	3
Tamaño (bytes)	1024	1024	1024	11	6144
PtrDir1	11	12	13	14	16
PtrDir2	NULL	NULL	NULL	NULL	17
PtrIndSim1	NULL	NULL	NULL	NULL	18
PtrIndDob1	NULL	NULL	NULL	NULL	NULL
Nodo-i	<b>16</b>	<b>17</b>	<b>18</b>	<b>19</b>	<b>20</b>
Tipo	DIR	DIR	SLNK		
Nº enlaces	3	2	1		
Tamaño (bytes)	1024	1024	28		
PtrDir1	23	15	24		
PtrDir2	NULL	NULL	NULL		
PtrIndSim1	NULL	NULL	NULL		
PtrIndDob1	NULL	NULL	NULL		

3. Calcule el número de bloques de punteros y de datos que necesitaría el archivo de tamaño máximo en esta partición. Asimismo indique la cantidad de bytes que podría almacenar dicho archivo. (2 puntos)

**Respuesta:**

Según la estructura indicada para este sistema de archivos:

- Nº de punteros que contiene un bloque es:  $\frac{2^{10} \text{bytes}/\text{bloque}}{2^2 \text{bytes}/\text{ptr}} = 2^8 \text{ptrs}/\text{bloque}$ .

Se sabe que desde un nodo-i se pueden referenciar:

- 2 ptrs. directos = 2 bloques de datos.
- 1 ptr. indirecto simple = 1 bloque de punteros directos +  $2^8$  bloques de datos.
- 1 ptr. ind. doble = 1 blq. ptrs. inds. simp. +  $2^8$  blqs. ptrs. dirs. +  $2^8 * 2^8$  blqs. datos.

Por tanto:

- El número de bloques de punteros es:  $1 + 1 + 256 = 258$  bloques de punteros.
- El número de bloques de datos es:  $2 + 256 + 4096 = 65.794$  bloques de datos.
- Tamaño máx. de archivo:  $65.794$  bloques de datos \* 1.024 bytes/bloque = 64,25 MB.

Ejercicio final de evaluación continua  
 Sistemas Operativos Avanzados  
 Parte Práctica 1

## Ejercicio 1

Considere el siguiente programa escrito en lenguaje C:

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define PAGE_SIZE_BYTES 4096
5
6 int main() {
7     char* data;
8     int i;
9
10    data = malloc(300*PAGE_SIZE_BYTES);
11
12    printf("start data pointer = %p\n", data);
13    printf("end data pointer = %p\n", data+(300*PAGE_SIZE_BYTES));
14
15    getchar();
16
17    for (i=0; i< 32*PAGE_SIZE_BYTES; i++) {
18        data[i]=55;
19    }
20
21    printf("end data loop = %p\n", &data[i]);
22    getchar();
23
24    free(data);
25
26    return 0;
27 }
```

En informática, el tamaño del conjunto residente o RSS, del inglés Resident Set Size, es la cantidad de memoria de un proceso que se encuentra realmente en memoria principal (RAM). El resto de la memoria del proceso puede estar en el área de intercambio (swap, o sistema de archivos), bien porque haya sido desalojada (paged out) o bien porque nunca haya sido cargada. Por su lado, el tamaño de memoria virtual o VSZ, del inglés Virtual Memory Size, incluye toda la memoria a la que el proceso puede acceder, incluyendo aquella que se encuentre en espacio de intercambio y aquella procedente de bibliotecas compartidas.

La salida por pantalla del programa anterior muestra el siguiente resultado: start data pointer = 0x170000, end data pointer = 0x29c000 y end data loop = 0x190000.

Consideré así mismo que el sistema tiene un procesador que gestiona su memoria mediante paginación, y sobre él corre un sistema operativo similar a Linux. Con esta información, responda a las siguientes preguntas.

1. Durante la ejecución del programa, se ejecuta la orden `ps ax -o rss,vsz,command | grep sample` para obtener el RSS y el VSZ del proceso que lo ejecuta. Después de repetir la ejecución varias veces se comprueba que el valor de VSZ es siempre 4268932, mientras que el valor del RSS es unas veces 800KiB y otras veces 928KiB. ¿Cuál puede ser la explicación de esta variación? ¿Podría estar relacionado con el punto que había alcanzado la ejecución del programa cuando se invoca la orden `ps`? Explique su respuesta brevemente.

(0.5 puntos)

**Answer:** El resultado depende de si la ejecución de la orden `ps` se hace antes o después del bucle que escribe datos en la memoria asignada de forma dinámica, ya que esa escritura fuerza a que se asigne memoria de forma efectiva, mediante la asignación de marcos de página, lo que hace que se incremente el RSS.

2. Si el programa se ejecuta en una máquina con paginación, con un tamaño de página de 4096 bytes, ¿cuál será la cadena de referencia resultante de la ejecución del bucle de las líneas 17-19? Consideré únicamente los accesos al heap, es decir, las escrituras a la zona de memoria apuntadas por el puntero **data** (no considere código, ni otro tipo de variables).

(0.5 puntos)

**Answer:** The first address is 0x170000, and the last address 0x29c000. If we make a logic shift to the right, thus eliminating the 12 least significative bits, the remainder are the page number. The reference string will be 0x170, 0x171 ..., 0x18F.

3. Si el programa se ejecuta en una máquina con paginación paginada, con un tamaño de página de 4096 bytes, y con directorios de  $2^{10}$  entradas, ¿cuál será la cadena de referencia resultante de la ejecución del bucle de las líneas 17-19? Consideré únicamente los accesos al heap, es decir, las escrituras a la zona de memoria apuntadas por el puntero **data** (no considere código, ni otro tipo de variables).

(0.5 puntos)

**Answer:** The reference string now has to include the directory entry along with the page number. Given that page number can be expressed with 10 bits, if we do another logic shift to the right in the last address 0x29c, the result will be directory entry 0 for all the pages. Thus, the resulting reference string will be (0,0x170), (0,0x171) ... (0,0x18F).

4. Si el sistema de memoria virtual necesitara liberar tres marcos de página y tuviera que elegir entre los asignados para el puntero **data** cuando ya se hubiera alcanzado la ejecución de la línea 21, ¿qué páginas serían las víctimas si el algoritmo de reemplazo fuera FIFO con segunda oportunidad? Suponga que hay marcos de página suficientes y que no ha sido necesario realizar anteriormente ningún tipo de reemplazo.

(0.5 puntos)

**Answer:** All the pages allocated for the data pointer will be referenced, so FIFO second chance will degenerate into FIFO, and therefore the three victims will be the three first paged allocated, that is 0x170, 0x171 and 0x172.

**Ejercicio final de evaluación continua****Sistemas Operativos Avanzados****Parte Práctica 2****Ejercicio 2**

Sea un disco SATA con tres particiones lógicas. Se sabe que la primera partición es la activa y está formateada con un sistema de archivos tipo UNIX, que se organiza en:

- El sector de arranque (MBR), que ocupa 1 bloque.
- El superbloque mantiene información acerca del sistema de archivos. Esto es, el tamaño del bloque de datos, 1.024 bytes, el máximo número de bloques de datos,  $2^{10}$ , el tamaño de un directorio, 1.024 bytes, el mapa de bits de nodos-índice libres, etc. El superbloque ocupa 2 bloques.
- La tabla de nodos índice ocupa 17 bloques y almacena, entre otras, la siguiente información:
  - Tipo de archivo: archivo regular (REG), directorio (DIR), enlace (SLNK) y tubería con nombre (FIFO).
  - Contador de enlaces.
  - Tamaño del archivo en bytes.
  - 1 puntero directo: PtrDir.
  - 1 puntero indirecto simple: PtrIndSim.
  - 1 puntero indirecto doble: PtrIndDob.
- Los bloques de datos.

La ejecución de la orden `$ ls -Rli /` muestra un listado recursivo desde el directorio raíz, indicando para cada archivo: nodo índice, permisos, número de enlaces fuertes, tamaño en bytes y nombre.

```
3 drwxr-xr-x 2 1024 Sistema
6 -rwxr-xr-x 2 3000 prueba.c
7 drwxr-xr-x 2 1024 Documentos
10 -rwxr-xr-x 1 526000 ejemplo.txt
./Documentos:
6 -rwxrwxrwx 2 3000 pract1.c
8 lrwxr-xr-x 1 14 pract2.slnk ->../ejemplo.txt
9 lrwxr-xr-x 1 12 pract3.slnk ->../datos.txt
./Sistema:
4 -rwxr-xr-x 1 1000 cd
5 -rwxr-xr-x 1 5000 bash
```

1. Complete las tablas de la plantilla adjunta indicando el contenido de la tabla de nodos índice y de los bloques de la partición.

(0.8 puntos)

RESPUESTA:

Nodo-i	2	3	4	5	6	7	8	9	10
Tipo	DIR	DIR	REG	REG	REG	DIR	SLNK	SLNK	REG
Enlaces	4	2	1	1	2	2	1	1	1
Tam.bytes	1024	1024	1000	5000	3000	1024	14	12	526000
PtrDir	0	1	2	3	8	11	12	13	14
PtrIndSim	NULL	NULL	NULL	4	9	NULL	NULL	NULL	15
PtrIndDob	NULL								

Cuadro 2: Tabla de nodos índice.

0	1	2	3	4	5
.	2	.	3	Archivo cd	Archivo bash (inicio)
..	2	..	2		Ptr.dir. 5 Ptr. dir. 6 Ptr. dir. 7
Sistema	3	cd	4		
prueba.c	6	bash	5		
Documentos	7				
ejemplo.txt	10				
6	7	8	9	10	11
	Archivo bash (cont.)	Archivo prueba.c o pract1.c (inicio)	Ptr.dir. 10	Archivo prueba.c o pract1.c (cont.)	. 7 .. 2 pract1.c 6 pract2.slnk 8 pract3.slnk 9
12	13	14	15	16 ..527	
../ejemplo.txt	../datos.txt	Archivo ejemplo.txt (inicio)	Ptrs. dirs. 16 .. 527	Archivo ejemplo.txt (cont.)	

Cuadro 3: Bloques de datos.

El número de bloques que requiere el archivo de nodo-i 10, es:  $526.000 \text{ B} / 1.024 \text{ B/bloque} = 513,7 = 514$  bloques.

Sabiendo que un bloque se referencia con 16 bits/puntero, cada bloque de datos tendrá 1024 bytes/bloque / 2 bytes/puntero  $\Rightarrow 512$  punteros.

Por tanto, para este archivo se requiere:

1 bloque de datos referenciado por el puntero directo + 1 bloque de punteros (que a su vez, referenciará a 512 bloques de datos).

2. Indique el tamaño total, en bloques, que tendría el archivo de mayor tamaño que podría almacenarse en dicha partición.

(0.3 puntos)

RESPUESTA:

Tamaño del puntero a bloque de datos: 16 bits/puntero = 2 bytes/puntero.

En cada bloque de datos caben 1024 bytes/bloque / 2 bytes/puntero  $\Rightarrow 512$  punteros.

Para calcular el tamaño máximo de archivo, se consideran los bloques referenciados por:

- Puntero directo: 1 bloque de datos.
- Puntero indirecto simple: 1 bloque de punteros directos + 512 bloques de datos.

- Puntero indirecto doble: 1 bloque ptrs. ind. simples + 512 bloques ptrs. directos +  $512^2$  bloques de datos.

Por tanto, el número de bloques para el archivo de mayor tamaño, sería: 1 datos + (1 ptrs. + 512 datos) + (1 ptrs. + 512 ptrs. +  $512^2$  datos)=  
 514 bloques de punteros + 262.657 bloques de datos = 263.171 bloques.

Según la información que proporciona el superbloque, el máximo número de bloques direccionable es  $2^{16}$  bloques = 65.536 bloques.

Por ello, el archivo de mayor tamaño tendrá:

$2^{16}$  bloques - 20 bloques (MBR, superbloque y lista de nodos índice) = **65.516 bloques** (aproximadamente 64 MB).

3. Indique qué cambios tendrán lugar en el sistema de archivos descrito (superbloque, lista de nodos-i y bloques de datos) al ejecutar secuencialmente cada una de las órdenes siguientes:

```
$ rm /Documentos/pract1.c
$ rm /prueba.c
$ rm /Documentos/pract3.slnk
```

(0.3 puntos)

RESPUESTA:

```
$ rm /Documentos/pract1.c
```

- Superbloque: No se modifica.
- Lista de nodos indice: Se decrementa el número de enlaces del nodo índice 6, siendo su valor 1.
- Bloques de datos: Se elimina la entrada de directorio **pract1.c** en el bloque de datos 11.

```
$ rm /prueba.c
```

- Superbloque: Se libera el nodo-i 6 y, por tanto, los bloques de datos 8 y 9; así como el bloque 10, que está referenciado desde el bloque 9.
- Bloques de datos: Se elimina la entrada de directorio **prueba.c** en el bloque 0.

```
$ rm /Documentos/pract3.slnk
```

- Superbloque: Se libera el nodo-i 9 y, por tanto, el bloque de datos 13.
- Bloques de datos: Se elimina la entrada de directorio **pract3.slnk** en el bloque 11.

4. Suponga que un proceso ejecuta las siguientes llamadas sobre el archivo **/ejemplo.txt**:

```
fd = open("/ejemplo.txt", O_RDWR); //resuelve la ruta y asigna el fd
lseek(fd, 0, SEEK_END); // posiciona el puntero al final del archivo
write(fd, &buffer, 1024); // escribe 1024 caracteres, contenidos en buffer
```

Detalle, paso a paso, las operaciones que realizará el sistema operativo y las estructuras de datos que se modifican.

(0.6 puntos)

RESPUESTA:

La llamada al sistema **open**:

- Resuelve la ruta a partir del nodo-i 2, buscando la entrada **ejemplo.txt** en el bloque 0, para obtener su nodo-i, 10.
- Comprueba si los permisos del archivo permiten su apertura, según el usuario, grupo o resto.

- Crea un descriptor de archivo en la tabla de descriptores del proceso del bloque de control de procesos, y actualiza la tabla de archivos abiertos en el sistema.

La llamada al sistema **lseek**:

- A partir del descriptor de archivo pasado como parámetro, y tras acceder a la información del puntero indirecto simple del nodo-i 10, posiciona el puntero al final del bloque 527.

La llamada al sistema **write**:

- Selecciona el descriptor de archivo pasado como parámetro.
- Sabiendo que según el paso anterior, el puntero para la escritura está en el bloque 527, y que éste es el último referenciado por el puntero indirecto simple; en el nodo-i 10 se ocupa el puntero indirecto doble con la referencia al bloque de datos 528.
- El bloque de datos 528, contendrá un puntero indirecto al bloque 529, que a su vez, mantendrá el puntero directo al bloque de datos 530.
- Incrementa el tamaño final del archivo en 3\*1024 bytes.

**Ejercicio final de evaluación continua  
Sistemas Operativos Avanzados  
Parte Práctica**

**Ejercicio 1**

La unidad de gestión de memoria de un computador presenta un esquema de paginación, con direccionamiento a nivel de byte y las siguientes estructuras:

- Dirección de memoria virtual formada por el campo página, 16 bits, y el campo desplazamiento, 24 bits.
- Dirección de memoria física formada por el campo marco, 10 bits, y el campo desplazamiento, 24 bits.
- Entrada en la tabla de páginas formada por los bits P, S, R, L, E y X y el marco en el que se ubica dicha página. El valor 1 para estos bits indica respectivamente si la página está presente en memoria, está compartida, ha sido referenciada y tiene permisos de lectura, escritura y/o ejecución.

1. Según esta información, responda a las siguientes preguntas en la plantilla adjunta:

- a) ¿Cuál sería la máxima cantidad de memoria principal que podría utilizarse?
- b) ¿Cuál es el tamaño de una página de memoria virtual? ¿Y de un marco de memoria principal?
- c) ¿Cuál sería el tamaño del mayor proceso que pueda ejecutarse?
- d) ¿Cuál sería el tamaño ocupado por una tabla de páginas para dicho proceso?

(0,5 puntos)

**RESPUESTA:**

- a) La máxima cantidad de memoria RAM que podría instalarse vendría dada por las líneas de una dirección de memoria principal, 34 bits. De modo que con 34 bits pueden direccionar  $2^{34}B = 16\text{ GB}$ .
- b) Ambos tamaños vienen dados por el desplazamiento que se puede realizar en una página o en un marco de página. De modo que con 24 bits se pueden direccionar  $2^{24}B = 16\text{ MB por página o 16 MB por marco}$ .
- c) El tamaño máximo del mayor proceso vendría dado por el número de líneas de una dirección de memoria virtual, 40 bits. De modo que con 40 bits, se pueden direccionar procesos de  $2^{40}B = 1\text{ TB}$ .
- d) Una entrada en la tabla de páginas tiene 16 bits ( $2B/\text{entrada}$ ) y el número de entradas de un proceso de tamaño máximo es de  $2^{16}B = 64K\text{entradas}$ . Por tanto, la tabla de páginas para el proceso de mayor tamaño sería de  $64K\text{entradas} * 2B/\text{entrada} = 128\text{ KB}$ .

En el instante t, el sistema operativo gestiona únicamente dos procesos. Sabiendo que la tabla de páginas del proceso P1 es:

	PSRL EX <sub>2</sub>	Marco <sub>10</sub>
0	0110 00	0x008
1	1101 00	0x00E
2	1110 00	0x006
3	0110 00	0x004
4	1010 00	0x004

Cuadro 5: Tabla de páginas del proceso P1.

2. ¿Podrían estar compartiendo información ambos procesos? Si es así, indique los marcos y direcciones de memoria principal compartidas.

(0,2 puntos)

**RESPUESTA:**

Sí, ya que el proceso P1 tiene presentes en memoria las páginas 1 y 2 (observar bit P), que además están compartidas (observar bit S). Por tanto, se puede afirmar que, al menos, **se comparten los marcos E y 6**, que corresponden respectivamente a las **direcciones de memoria principal: 0 0E00 0000 .. 0 0EFF FFFF y 0 0600 0000 .. 0 06FF FFFF**.

3. Realice la traducción de las siguientes direcciones virtuales a direcciones de memoria principal. En caso de no ser posible la traducción, identifique el problema.

- 0x00 01AB CDEF
- 0x00 03AB CDEF
- 0x00 04AB CDEF

(0,3 puntos)

**RESPUESTA:**

- 0x00 01AB CDEF → 0 0EAB CDEF
- 0x00 03AB CDEF → Excepción, fallo de página, ya que no está presente en memoria principal, bit P = 0.
- 0x00 04AB CDEF → 0 04AB CDEF

A continuación se reinicia el computador y se lanzan dos nuevos procesos. Se sabe que el sistema de memoria gestiona un esquema de paginación por demanda con un algoritmo de reemplazo FIFO segunda oportunidad.

La cadena de referencias del proceso P1 es: 1, 0, 2, 3, 0, 1, 5, 1, 2, 4 y 3; mientras que el proceso P2 tiene la cadena de referencias: 9, 7, 8, 6, 7, 9, 5, 7 y 6.

4. Indique qué referencias generan un fallo de página, suponiendo que existe una política de asignación de marcos local con cuatro marcos de página por proceso. Muestre el estado del bit de referencia en cada paso.

(0,4 puntos)

**RESPUESTA:**

Cad. refs.	1	0	2	3	0	1	5	1	2	4	3
	FP	FP	FP	FP	A	A	FP	FP	A	FP	FP
Marco 0	R=1 1	R=1 1	R=1 1	* R=1 1	=	=	R=1 5	R=1 5	R=1 5	* R=1 5	R=0 5
Marco 1	*	R=1 0	R=1 0	R=1 0			* R=0 0	R=1 1	R=1 1	R=1 1	R=0 1
Marco 2		*	R=1 2	R=1 2			R=0 2	* R=0 2	* R=1 2	R=0 2	R=1 3
Marco 3			*	R=1 3			R=0 3	R=0 3	R=0 3	R=1 4	* R=1 4

Cuadro 6: Ejecución del algoritmo FIFO segunda oportunidad para P1.

Cad. refs.	9	7	8	6	7	9	5	7	6
	FP	FP	FP	FP	A	A	FP	A	A
Marco 0	R=1 9	R=1 9	R=1 9	* R=1 9	=	=	R=1 5	R=1 5	R=1 5
Marco 1	*	R=1 7	R=1 7	R=1 7			* R=0 7	* R=1 7	* R=1 7
Marco 2		*	R=1 8	R=1 8			R=0 8	R=0 8	R=0 8
Marco 3			*	R=1 6			R=0 6	R=0 6	R=1 6

Cuadro 7: Ejecución del algoritmo FIFO segunda oportunidad para P2.

5. Se sabe que el sistema recibe referencias de ambos procesos en forma alternada, comenzando por el proceso P1 (1, 9, 0, 7, ...). Indique qué referencias generan un fallo de página, suponiendo una política de asignación de marcos global con siete marcos de página disponibles para los procesos de usuario. Considere que la política de reemplazo es, al igual que en caso anterior, FIFO de segunda oportunidad, salvo que esta vez su ámbito es global. Muestre el estado del bit de referencia en cada paso.

(0,4 puntos)

**RESPUESTA:**

Cad. refs.	1	9	0	7	2	8	3	6	0	7	1
	FP	FP	A	A	FP						
Marco 0	R=1 1	* R=1 1	R=1 6	R=1 6	R=1 6	R=1 6					
Marco 1	*	R=1 0	R=1 9	* R=0 9	* R=0 9	* R=0 9	R=1 1				
Marco 2	*	R=1 0	R=0 0	R=1 0	R=1 0	* R=1 0					
Marco 3	*	R=1 7	R=0 7	R=0 7	R=1 7	R=1 7	R=1 7				
Marco 4	*	R=1 2	R=0 2	R=0 2	R=0 2	R=0 2	R=0 2				
Marco 5	*	R=1 8	R=0 8	R=0 8	R=0 8	R=0 8	R=0 8				
Marco 6	*	R=1 3	R=0 3	R=0 3	R=0 3	R=0 3	R=0 3				

Cad. refs.	9	5	5 (P2)	1	7	2	6	4	3
	FP	FP	FP	A	A	FP	A	FP	FP
Marco 0	R=1 6	R=1 6	* R=1 6	=	* R=1 6	R=0 6	R=1 6	R=0 6	R=0 6
Marco 1	R=1 1	R=1 1	R=1 1		R=1 1	R=0 1	R=0 1	R=1 4	R=1 4
Marco 2	R=0 0	R=0 0	R=0 0		R=0 0	R=1 2	R=1 2	* R=1 2	R=0 2
Marco 3	R=0 7	R=0 7	R=0 7		R=1 7	* R=1 7	* R=1 7	R=0 7	* R=1 3
Marco 4	R=1 9	R=1 9	R=1 9		R=1 9	R=1 9	R=1 9	R=0 9	R=0 9
Marco 5	* R=0 8	R=1 5	R=1 5		R=1 5	R=1 5	R=1 5	R=0 5	R=0 5
Marco 6	R=0 3	* R=0 3	R=1 5-P2		R=1 5-P2	R=1 5-P2	R=1 5-P2	R=0 5-P2	R=0 5-P2

Cuadro 8: Ejecución del algoritmo FIFO segunda oportunidad para los procesos P1 y P2 con siete marcos de página.

6. Indique cuántos fallos de página se producen en la solución de apartado 4 y en la del apartado 5 y, a la vista de los resultados, indique qué política/s considera más adecuada/s para la implementación del sistema de memoria, suponiendo que en cualquier caso se emplea el algoritmo FIFO segunda oportunidad.

(0,2 puntos)

**RESPUESTA:**

Suponiendo que la política de asignación de marcos es local a cada proceso, y cada uno de ellos dispone de cuatro marcos de página, el proceso P1 genera ocho fallos de página y el proceso P2 genera cinco fallos de página.

En el caso de implementar una política de asignación de marcos es global al sistema, para los procesos de usuario, y sabiendo que se dispone de siete marcos de página; la ejecución alternada de los procesos P1 y P2 generaría 15 fallos de página.

Aparentemente daría mejores resultados el utilizar una política de asignación local. Sin embargo, si se asignan ocho marcos de página de forma global a ambos procesos, el resultado en cuanto al número de fallos de página sería menor. En todo caso, tanto para la política de asignación global como para la local, añadir más marcos de página mejoraría los resultados.

**Ejercicio final de evaluación continua**  
**Sistemas Operativos Avanzados**  
**Parte Práctica**

### Ejercicio 2

Se tiene un sistema de archivos basado en FAT llamado Microfat. En este sistema la FAT tiene 256 entradas, y cada una de ellas es de 8 bit. Para cada entrada de la FAT, 0x00 indica que el bloque correspondiente no está siendo utilizado, 0xFF significa que ese es el último bloque del archivo y cualquier otro valor indica el siguiente número de bloque que compone el archivo.

En este sistema de archivos los bloques son de 32 bytes.

En Microfat sólo hay un directorio raíz. Cada entrada de directorio contiene un nombre de 8 bytes, una extensión de 3 bytes, un byte indicando el número total de bloques del archivo y otro byte que indica cuál es el primer bloque.

8
3
1
1
13

1. ¿Cuál es el tamaño del archivo más grande que se puede almacenar en Microfat, considerando únicamente la limitación impuesta por el diseño de la FAT?

(0,1 puntos)

#### RESPUESTA:

Si cada entrada de la FAT es de 8 bit, se pueden referenciar hasta 256 bloques. Cada bloque es de 32 bytes, así es que el tamaño más grande referenciable por la FAT sería  $2^8 * 32 = 8KiB$ . Habría que restar a este valor 64 bytes, ya que el bloque 0 y el 256 no se pueden utilizar de forma efectiva, al ser esos números los delimitadores especiales de la FAT.

2. Debido a un fallo hardware, el sistema Microfat existente quedó dañado. Concretamente se perdió el contenido de la FAT, aunque se pudo rescatar el contenido de los bloques de datos. A continuación se muestra el volcado hexadecimal de los datos rescatados. Los bloques de datos se numeran comenzando por 1.

Bq1	00000000	0e 6c 61 20 63 6f 6e 63 6c 75 69 61 6e 20 73 61	.la concluian sa
	00000010	79 6f 20 64 65 20 76 65 6c 61 72 74 65 20 28 32	yo de velarte (2
Bq2	00000020	06 20 66 6c 61 63 6f 20 79 20 67 61 6c 67 6f 20	. flaco y galgo
	00000030	63 6f 72 72 65 64 6f 72 2e 20 55 6e 61 20 6f 6c	corredor. Una ol
Bq3	00000040	04 75 6e 20 68 69 64 61 6c 67 6f 20 64 65 20 6c	.un hidalgo de l
	00000050	6f 73 20 64 65 20 6c 61 6e 7a 61 20 65 6e 20 61	os de lanza en a
Bq4	00000060	05 73 74 69 6c 6c 65 72 6f 2c 20 61 64 61 72 67	.stillero, adarg
	00000070	61 20 61 6e 74 69 67 75 61 2c 20 72 6f 63 69 6e	a antigua, rocin
Bq5	00000080	02 79 6f 20 6e 6f 6d 62 72 65 20 6e 6f 20 71 75	.yo nombre no qu
	00000090	69 65 72 6f 20 61 63 6f 72 64 61 72 6d 65 2c 20	iero acordarme,
Bq6	000000a0	0b 61 6c 6f 6d 69 6e 6f 20 64 65 20 61 6e 61 64	.alomino de anad
	000000b0	69 64 75 72 61 20 6c 6f 73 20 64 6f 6d 69 6e 20	idura los domin
Bq7	000000c0	07 6c 61 20 64 65 20 61 6c 67 6f 20 6d 61 73 20	.la de algo mas
	000000d0	76 61 63 61 20 71 75 65 20 63 61 72 6e 65 72 6f	vaca que carnero
Bq8	000000e0	08 2c 20 73 61 6c 70 69 63 6f 6e 20 28 31 29 20	.. salpicon (1)
	000000f0	6c 61 73 20 6d 61 73 20 6e 6f 63 68 65 73 2c 20	las mas noches,
Bq9	00000100	0d 64 65 20 73 75 20 68 61 63 69 65 6e 64 61 2e	.de su hacienda.
	00000110	20 20 45 6c 20 72 65 73 74 6f 20 64 65 20 65 6c	El resto de el
BqA	00000120	09 64 75 65 6c 6f 73 20 79 20 71 75 65 62 72 61	.duelos y quebra
	00000130	6e 74 6f 73 20 6c 6f 73 20 73 61 62 61 64 6f 73	ntos los sabados
BqB	00000140	0a 2c 20 6c 65 6e 74 65 6a 61 73 20 6c 6f 73 20	.. lentejas los
	00000150	76 66 65 72 6e 65 73 2c 20 61 6c 67 75 6e 20 70	viernes, algun p
BqC	00000160	01 45 6e 20 75 6e 20 6c 75 67 61 72 20 64 65 20	.En un lugar de
	00000170	6c 61 20 4d 61 6e 63 68 61 2c 20 64 65 20 63 75	la Mancha, de cu
BqD	00000180	13 33 29 20 64 65 20 6c 6f 20 6d 61 73 20 66 69	.3) de lo mas fil
	00000190	6e 6f 2e 20 20 20 20 20 20 20 20 20 20 20 20 20	no.
BqE	000001a0	0c 67 6f 73 2c 20 63 6f 6e 73 75 6d 69 61 6e 20	.gos, consumian
	000001b0	6c 61 73 20 74 72 65 73 20 70 61 72 74 65 73 20	las tres partes
BqF	000001c0	10 69 65 73 74 61 73 2c 20 63 6f 6e 20 73 75 73	.iestas, con sus
	000001d0	20 70 61 6e 74 75 66 6c 6f 73 20 64 65 20 6c 6f	pantuflas de lo
Bq10	000001e0	11 20 6d 69 73 6d 6f 2c 20 79 20 6c 6f 73 20 64	. mismo, y los d

000001f0	69 61 73 20 64 65 20 65	6e 74 72 65 20 73 65 6d	ias de entre sem
Bq11 00000200	01 29 2c 20 63 61 6c 7a	61 73 20 64 65 20 76 65	.), calzas de ve
00000210	6c 6c 75 64 6f 20 70 61	72 61 20 6c 61 73 20 66	lludo para las f
Bq12 00000220	12 61 6e 61 20 73 65 20	68 6f 6e 72 61 62 61 20	.ana se honraba
00000230	63 6f 6e 20 73 75 20 76	65 6c 6c 6f 72 69 20 28	icos su vellori (
Bq13 00000240	03 6e 6f 20 68 61 63 65	20 6d 75 63 68 6f 20 74	.no hace mucho ti
00000250	69 65 6d 70 6f 20 71 75	65 20 76 69 76 69 61 20	tiempo que vivia
Bq14 00000260			

Se sabe que el archivo estaba almacenado en un formato propietario llamado Minidoc. Minidoc divide el contenido de sus documentos en segmentos de 32 bytes. Obsérvese que cada bloque de datos contiene en el primer byte su número de orden original, comenzando por 0x01, y le sigue la información del archivo en los siguientes bytes. Con esta información, se pide que reconstruya el contenido de una tabla FAT que permita recuperar el archivo completo, así como una posible entrada de directorio con un nombre arbitrario. Suponga una FAT inicializada tal como se muestra en el siguiente ejemplo:

index	value								
0	FF	5	00	10	00	15	00	20	00
1	00	6	00	11	00	16	00	21	00
2	00	7	00	12	00	17	00	22	00
3	00	8	00	13	00	18	00	23	00
4	00	9	00	14	00	19	00	24	00

(0,8 puntos)

**RESPUESTA:**

Como los segmentos de Minidoc son del mismo tamaño que los bloques de datos, todos los bloques de datos comenzarán por un número que indica el orden del segmento dentro del archivo. Como los bloques son de 32 bytes (0x20), en el offset 0 comenzará el bloque 1, en el offset 0x20 el bloque 2, en el 0x40 el bloque 3, y en general en el offset n comenzará el bloque  $n/20 + 1$  (en hexadecimal). Buscamos por lo tanto un bloque que comience por 1, que se encuentra en el offset 0x160 (bloque 0x0C). A continuación uno que comience por 2, que se encuentra en el offset 0x80 (bloque 0x05), y así sucesivamente llegamos a la siguiente tabla:

seg#	Offset	Bloque
1	0x160	0x0C
2	0x080	0x05
3	0x240	0x13
4	0x040	0x03
5	0x060	0x04
6	0x020	0x02
7	0x0C0	0x07
8	0x0E0	0x08
9	0x120	0x0A
10	0x140	0x0B
11	0x0A0	0x06
12	0x1A0	0x0E
13	0x100	0x09
14	0x000	0x01
15	0x200	0x11
16	0x1C0	0x0F
17	0x1E0	0x10
18	0x220	0x12
19	0x180	0x0D

La tabla FAT quedaría por lo tanto de la siguiente forma:

OF	10	
6	10	12
12	0D	
OD	FF	

index	value								
0	FF	5	13	0A	0B	0F	10	14	00
1	11	6	0E	0B	06	10	12	15	00
2	07	7	08	0C	05	11	0F	16	00
3	04	8	0A	0D	FF	12	0D	17	00
4	02	9	01	0E	09	13	03	18	00

La entrada de directorio tendría un nombre de archivo, por ejemplo **quijote** con extensión **.txt**, que ocupa 19 bloques y cuyo primer bloque sería 0x0C.

3. Se dispone así mismo de otro sistema de archivos llamado micronode que, a diferencia del anterior, está basado en nodos-i. En este sistema, un nodo-i contiene, entre otras cosas, un puntero directo y uno indirecto simple. Ambos punteros son de 8 bit. ¿Cuál es el tamaño de archivo más grande que se puede crear con este tipo de nodos-i?.

(0,1 puntos)

**RESPUESTA:**

Un puntero directo permite referenciar un bloque. Un puntero indirecto simple apunta a un bloque de 32 bytes, con capacidad para albergar 32 punteros, ya que según el enunciado, los punteros a los bloques son de 8 bits. Con esta configuración se podrían referenciar  $1 + 1 + 32 = 34$  bloques, y por lo tanto el archivo máximo sería de  $34 \text{ bloques} * 32 \text{ bytes/bloque} = 1,088 \text{ bytes}$

4. ¿Qué valores tendría un nodo-i (solo el valor de los punteros) que permitiera reconstruir el archivo perdido? Si fuera necesario utilizar un puntero indirecto, indique así mismo el contenido del bloque apuntado por dicho puntero.

(0,7 puntos)

**RESPUESTA:**

El puntero directo apuntaría al primer bloque, es decir 0x0C. El puntero indirecto apuntaría a un bloque de punteros. Como el último bloque del archivo es 19, podemos suponer que el 20 está libre. Este bloque contendría los siguientes bytes: 0x05, 0x13, 0x03, 0x04, 0x02, 0x07, 0x08, 0x0A, 0x0B, 0x06, 0x0E, 0x09, 0x01, 0x11, 0x0F, 0x10, 0x12, 0x0D

5. Se piensa en modificar el diseño de nodo-i para poder almacenar archivos más grandes, y se plantean tres alternativas:

- a) Contar con dos punteros indirectos.
- b) Contar con dos punteros indirectos dobles.
- c) Contar con puntero directo y puntero indirecto doble.

Sabiendo que el hardware no podrá soportar sistemas de archivos de más de 48 KiB. ¿Cuál de las tres opciones elegiría? Justifique su respuesta.

(0,3 puntos)

**RESPUESTA:**

Dos punteros indirectos permitirían un tamaño máximo de archivo de  $2 * (1 + 32) = 66$  bloques, es decir  $66 * 32 = 2304 \text{ bytes} = 2,25 \text{ KiB}$ .

Dos punteros indirectos dobles podrían referenciar  $2 * (1 + 32 + 32^2) = 2,114 \text{ bloques}$ , por lo tanto un tamaño máximo de archivo de  $2114 * 32 = 67648 \text{ bytes} = 66,06 \text{ KiB}$ , que supera el máximo de 48 KiB.

Un puntero directo permite referenciar 1 bloque, y otro indirecto doble permite a su vez  $32 + 32^2 = 1056$  bloques. Ambos permitirían obtener un tamaño máximo de archivo de  $(1 + (1 + 32 + 32^2)) \text{ bloques} * 32 \text{ bytes/bloque} = 33856 \text{ bytes}$ .

Si el hardware de almacenamiento de este sistema de archivos no será mayor de 48 KiB, no merece la pena la segunda opción, ya que para archivos pequeños siempre se desperdiciarían bloques.

$$\begin{aligned} \circ C.MV &= 4GB = 2^{32} \text{ bytes} \Rightarrow 32 \text{ bits} \\ \circ C.MP &= 256MB = 2^{28} \text{ bytes} \Rightarrow 28 \text{ bits} \\ \circ \text{desplazamiento} & 12 \text{ bits} \end{aligned}$$

### Ejercicio 3

Sea un sistema con gestión de memoria paginada. La capacidad máxima de direcciónamiento virtual es de 4 GB y la memoria principal es de 256 MB. El desplazamiento dentro de una página y de un marco se realiza con 12 bits.

1. ¿Cuál es el tamaño de una página? ¿Y de un marco?
2. Indique el formato de una dirección de memoria virtual y de memoria principal.
3. ¿Cuál sería el número máximo de páginas que puede tener un proceso?
4. ¿Cuál sería el formato de una entrada en la tabla de páginas?
5. Indique la problemática que se plantea con respecto al tamaño de la tabla de páginas y proporcione una solución.
6. Según la solución del apartado anterior, diseñe una arquitectura para el nuevo sistema de gestión de memoria. Respete el tamaño del campo desplazamiento.

#### SOLUCIÓN:

1. En ambos casos el campo desplazamiento es de 12 bits, con los que se pueden direccionar  $2^{12}$  posiciones de memoria = 4 KB, suponiendo direccionamiento a nivel de byte.

2. Antes de dar respuesta a la cuestión planteada es necesario establecer el tamaño de las direcciones de memoria virtual y de memoria principal.

El tamaño de la memoria virtual para un proceso es de 4 GB =  $2^{32}$  bytes/proceso.

El tamaño de la memoria principal para todos los procesos en el sistema es de 256 MB =  $2^{28}$  bytes.

Si en ambos casos el campo desplazamiento es de 12 bits, el formato es:

- Formato de una dirección de memoria virtual: Página (20 bits) | Desplazamiento (12 bits).
- Formato de una dirección de memoria principal: Marco (16 bits) | Desplazamiento (12 bits).

3. El número máximo de páginas que puede tener un proceso vendrá dado por la capacidad de direccionamiento del campo P de una dirección virtual. Con 20 bits pueden identificarse  $2^{20}$  páginas.

Por tanto el número máximo de páginas que puede tener un proceso es 1 Mpáginas.

4. La tabla de páginas contiene los marcos de memoria principal, 16 bits, que identifican la parte alta de una dirección de memoria.

El formato de una entrada en la tabla de páginas consta al menos del campo Marco (16 bits).

5. Consultar bibliografía sobre el esquema de memoria virtual con paginación paginada.

La tabla de páginas podría tener un tamaño de:  $2^{20}$  entradas (referenciadas por las páginas del proceso) multiplicado por el tamaño de cada entrada, al menos 16 bits (2 bytes). Esto da un valor de, al menos,  $2^{20} * 2 \text{ bytes} = 2 \text{ MB}$ .

6. En el caso de modificar la arquitectura para utilizar un esquema de paginación paginada, la dirección virtual estaría formada por los campos:

Directorio (x bits) | Página (y bits) | Desplazamiento (12 bits), donde  $x+y = 20$  bits.

Además se tendría un tabla de directorio por proceso que apuntaría a tablas de páginas con el siguiente formato:

Formato tabla de directorio: Dir. TMP (28 bits).

Formato tabla de páginas: Marco (16 bits).

#### Ejercicio 4

En el sistema del Ejercicio 1 se ejecuta el proceso P1. La unidad de gestión de memoria utiliza la tabla de segmentos mostrada en la tabla 1, donde se referencia para cada segmento de memoria virtual, la dirección de inicio en memoria principal y el límite de cada segmento.

Tabla 1: Tabla de segmentos del proceso P1.

Segmento	Base <sub>HEX</sub>	Límite <sub>HEX</sub>
0	0F00 0000	FFFF
1	0000 7980	0600
2	0600 2000	0700
3	0300 0000	FFFF
4	0301 0000	C000

Según esta información, responda a las siguientes cuestiones:

1. Dibuje el mapa del espacio de direccionamiento virtual y físico del proceso P1.
  2. ¿Cuál es la correspondencia entre la dirección de un segmento de memoria virtual y una dirección memoria principal?
  3. ¿Qué direcciones de memoria principal se corresponden las siguientes referencias a direcciones virtuales?
 

Segmento 16 bits  $\Rightarrow$  FFFF - - - -

desplazamiento  $\Rightarrow$  - - - - FFFF
- a) 0x0004 0202  
 b) 0x0004 D898  
 c) 0x0003 000A  
 d) 0x0000 0509  
 e) 0x0001 06FF  
 f) 0x0002 0701  
 g) 0x0005 0001

#### SOLUCIÓN:

1. Consultar bibliografía sobre el esquema de segmentación de memoria.
2. Consultar bibliografía sobre el esquema de segmentación de memoria.
3. a) 0x0004 0202  
 Segmento: 0x4 < RLTS: 0x5.  
 La base del segmento es: 0x0301 0000 y el límite 0xC000.

Desplazamiento: 0x0202 < límite del segmento.

Se suma la base del segmento con el desplazamiento para obtener la dirección en memoria principal: 0x0301 0000 + 0202 = 0x0301 0202.

b) 0x0004 D898

Segmento: 0x4 < RLTS: 0x5.

La base del segmento es: 0x0301 0000 y el límite 0xC000.

Desplazamiento: 0xD898 < límite del segmento es FALSO.

Se produce una excepción por intento de acceso a una zona de memoria fuera de sus límites.

c) 0x0003 000A

Segmento: 0x3 < RLTS: 0x5.

La base del segmento es: 0x0300 0000 y el límite 0xFFFF.

Desplazamiento: 0x000A < límite del segmento.

Se suma la base del segmento con el desplazamiento para obtener la dirección en memoria principal: 0x0300 0000 + 000A = 0x0300 000A.

d) 0x0000 0509

La base del segmento es: 0x0F00 0000 y el límite 0xFFFF.

Desplazamiento: 0x0509 < límite del segmento.

Se suma la base del segmento con el desplazamiento para obtener la dirección en memoria principal: 0x0F00 0000 + 0509 = 0x0F00 0509.

e) 0x0001 06FF

Segmento: 0x1 < RLTS: 0x5.

La base del segmento es: 0x0000 7980 y el límite 0x0600.

Desplazamiento: 0x06FF < límite del segmento es FALSO.

Se produce una excepción por intento de acceso a una zona de memoria fuera de los límites del segmento.

f) 0x0002 0701

Segmento: 0x2 < RLTS: 0x5.

La base del segmento es: 0x0600 2000 y el límite 0x0700.

Desplazamiento: 0x0701 < límite del segmento es FALSO.

Se produce una excepción por intento de acceso a una zona de memoria fuera de los límites del segmento.

g) 0x0005 0001

Segmento: 0x5 < RLTS: 0x5. es FALSO.

Se produce una excepción por intento de acceso a una zona de memoria fuera de los límites del espacio de direccionamiento virtual asignado al proceso.

## Ejercicio 5

Siguiendo la arquitectura de la memoria virtual planteada en el ejercicio 3, se rediseña la tabla del mapa de páginas (TMP) añadiendo un conjunto de bits que indican, si dicha página está presente en memoria (bit P), si ha sido modificada (bit M), si ha sido referenciada (R), si es compartida con otro proceso (bit S), si es de lectura (bit R), si es de escritura (bit W) o si es de ejecutable (X).

A continuación se muestra la tabla de páginas del proceso P1 (ver tabla 2).

Tabla 2: Tabla del mapa de páginas del proceso P1.

Página	-PMR	SRWX <sub>BIN</sub>	Marco <sub>HEX</sub>
0	0000	0111	000A
1	0000	1000	0006
2	0100	1110	0007
3	0000	0110	0007
4	0000	0000	0007
5	0100	1101	0006
6	0000	1001	000A
7	0100	0110	000A

Según esta información, responda a las siguientes cuestiones:

1. ¿Qué páginas de memoria virtual están ubicadas en marcos de memoria física? ¿Cuál es la correspondencia?
2. ¿Qué direcciones de memoria física se corresponden las siguientes referencias a direcciones virtuales?  
*20 bits paginación → E F F F F - - -*
  - a) 0x0000 0202
  - b) 0x0000 2898
  - c) 0x0000 500A
  - d) 0x0000 7509
  - e) 0x0000 66FF
  - f) 0x0000 8701
3. El proceso P2 comparte su página 2, que es de datos, y su página 3, que es de código, con el proceso P1. ¿Cuál sería el contenido de las entradas 2 y 3 de la tabla del mapa de páginas del proceso P2?

#### SOLUCIÓN:

1. En memoria principal están ubicadas las páginas cuyo bit de presencia es 1. Estas son, la página 2 ubicada en el marco 0x0007, la página 5 ubicada en el marco 0x0006 y la página 7 ubicada en el marco 0x000A.
2. a) 0x0000 0202  
 Página: 0x0 < RLTP: 0x8 es FALSO.  
 La entrada 0 de la tabla de páginas tiene el bit P (presencia) = 0.  
 Se genera una excepción, fallo de página, ya que la página referenciada no se encuentra ubicada en ningún marco de memoria principal.
- b) 0x0000 2898  
 Página: 0x2 < RLTP: 0x8.  
 La entrada 2 de la tabla de páginas tiene el bit P (presencia) = 1. El marco de página es: 0x0007.  
 Desplazamiento: 0x898.  
 Se concatena el marco de página con el desplazamiento para obtener la dirección en memoria principal: 0x000 7898.

c) 0x0000 500A

Página: 0x5 < RLTP: 0x8.

La entrada 5 de la tabla de páginas tiene el bit P (presencia) = 1. El marco de página es: 0x0006.

Desplazamiento: 0x00A.

Se concatena el marco de página con el desplazamiento para obtener la dirección en memoria principal: 0x000 600A.

d) 0x0000 7509

Página: 0x7 < RLTP: 0x8.

La entrada 7 de la tabla de páginas tiene el bit P (presencia) = 1. El marco de página es: 0x000A.

Desplazamiento: 0x509.

Se concatena el marco de página con el desplazamiento para obtener la dirección en memoria principal: 0x000 A509.

e) 0x0000 66FF

Página: 0x6 < RLTP: 0x8.

La entrada 6 de la tabla de páginas tiene el bit P (presencia) = 0.

Se genera una excepción, fallo de página, ya que la página referenciada no se encuentra ubicada en ningún marco de memoria principal.

f) 0x0000 8701

Página: 0x8 < RLTP: 0x8 es FALSO.

Se genera una excepción por intento de acceso a una zona de memoria fuera de los límites del espacio de direccionamiento virtual asignado al proceso.

3. Las entradas 2 y 3 de la tabla del mapa de páginas del proceso P2 son:

2 | 0100 1110 | 0007, coincide con la página de datos compartida con el proceso P1.

3 | 0100 1101 | 0006, coincide con la página de código compartida con el proceso P1.

# Sistemas Operativos Avanzados

## Ejercicios del tema 1: Gestión de memoria

UAH. Departamento de Automática. ATC-SOL  
<http://atc1.aut.uah.es>

### Ejercicio 1

Sea un sistema con gestión de memoria segmentada y capacidad máxima de direccionamiento, virtual y real, de 4 GB. El número de segmento se indica con 16 bits.

1. ¿Cuál es el tamaño de una dirección virtual? ¿Y de una dirección física en memoria principal?
2. ¿Cuál es el número de bits en el campo desplazamiento? ¿Cuál es el formato de una dirección virtual?
3. ¿Cuál es el número máximo de segmentos que puede tener un proceso?
4. ¿Cuál es el tamaño máximo de un segmento?
5. ¿Cuál es el formato de una entrada en la tabla de segmentos?

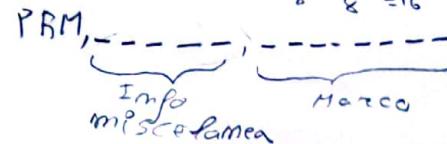
SOLUCIÓN:

1. El espacio de direccionamiento virtual es de 4 GB =  $2^{32}$  bytes. Por tanto, la dirección virtual tendrá 32 bits, formada por los campos: número de segmento y desplazamiento (en ese segmento).  
El espacio de direccionamiento real es de 4 GB =  $2^{32}$  bytes. Por tanto, una dirección física tendrá 32 bits.
2. Si el campo segmento es de 16 bits y una dirección de memoria virtual tiene 32 bits, el desplazamiento también tendrá 16 bits.

El formato de una dirección virtual es: Segmento (16 bits) | Desplazamiento (16 bits).

- C. MP = 4 GB =  $2^{32}$  bytes, 32 bits
- C. MV = 4 GB =  $2^{32}$  bytes, 32 bits
- N° de segmentos 16 bits

[S P]

**EJERCICIO GESTIÓN DE MEMORIA**

El sistema operativo ValOS utiliza un **sistema** de gestión de memoria con **paginación pura** y las siguientes características:

- La unidad de gestión de memoria tiene **direcciónamiento a nivel de byte**.
- La **dirección virtual** cuenta con 8 bits para el campo página y 8 bits para el desplazamiento.
- La dirección física cuenta con 8 bits para el campo marco y 8 bits para el desplazamiento.
- Una entrada en la **tabla de páginas** tiene el siguiente formato:
  - 3 bits más significativos indican si la página está **presente** en memoria (P), si ha sido **referenciada** (R) y si ha sido **modificada** (M).
  - Siguen 5 bits de **información miscelánea**.
  - 8 bits para **referenciar el marco** en memoria principal.
- El **mapa de bits** identifica qué marcos están libres en memoria principal, con un 0 indica si está **ocupado** y con 1 si está **libre**.

En el instante t, el **mapa de bits** es el siguiente (el número de marco se expresa en hexadecimal):

M0..M3	M4..M7	M8..M0B	M0C..M0F	M10..M13	M14..M17	M18..M1B	M1C..M1F	M20..M23	M24..M27	...	MFF
1011	1111	0111	1110	0010	0100	0100	0011	1111	1000	...	0
z				11	15	16	17	22	23	12	

En dicho **instante** se inicia la tarea Ta, que contiene los hilos H1 y H2, que **requiere** de 4 KB de memoria de usuario. El sistema operativo realiza las siguientes operaciones:

- i. Crea un **bloque de control de procesos** en memoria principal, a partir de la posición 0x0100.
- ii. Crea la **tabla de páginas** en memoria principal, a partir de la posición 0x0800.
- iii. Asigna un espacio en memoria física para todo el proceso.

Responda a las siguientes preguntas:

1. Indique cómo quedaría el mapa de memoria principal.
2. Indique cuál sería el contenido de la tabla de páginas resultante.
3. Indique cuál sería el mapa de bits resultante.

En el instante t+10, el H1 escribe "Hola hilo 2" a partir de la dirección 0x00F9 de memoria virtual.

4. Indique el contenido y las posiciones de memoria virtual y de memoria principal que se ven modificadas.

Suponga que, tras un tiempo, la tarea se ha descargado de la memoria principal.

Al activarse el H2, el sistema carga de nuevo la tarea en memoria principal, pero en esta ocasión se asignan a la tarea únicamente 3 marcos de página vacíos.

El H2 genera la siguiente cadena de referencias:

0xA 0xB 0x06 0xA 0xC 0x06 0xA 0xB 0xC 0x01

Considerando que la política de reemplazo es FIFO de segunda oportunidad, local a la tarea,

5. Realice la ejecución paso a paso del algoritmo.

## SOLUCIÓN

- 1 y 2. Indique cómo quedaría el mapa de memoria principal.

La memoria principal está compartida por todos los procesos del sistema

Tal como puede observarse en el mapa de bits, están ocupados los marcos cuyo bit está a 0: 1 8 F 10 11 13 14 16 17 18 1A 1B 1C 1D 25 ... FF.

Los marcos libres son aquellos cuyo bit es 1, es decir marcos: 0 2 3 4 5 6 7 9 A B C D E 12 15 19 1E 1F 20 21 22 23 24.

En esta situación, se crea un proceso con dos hilos que requiere de 4 KB de memoria principal. Sabiendo que el tamaño de página es  $2^8$  bytes, el número de páginas que necesita el proceso es:

$$\frac{2^{12}}{2^8} = 2^4 = 16 \text{ páginas}$$

El sistema operativo realizará las siguientes operaciones:

- Crea un bloque de control de procesos en memoria principal, a partir de la posición 0x0100. Esta posición de memoria se corresponde con el marco 1, desplazamiento 0.  
· Es a partir de esta posición donde se ubicará el bloque de control de procesos de la tarea que acaba de crearse, que contiene los dos hilos a nivel de usuario.
- Crea la tabla de páginas en memoria principal, a partir de la posición 0x0800.  
Esta tabla de páginas contendrá las 16 entradas correspondientes a cada una de las páginas del proceso.

Cada entrada tiene los bits P, R, M y el marco donde se ubica cada página. De este modo la tabla de páginas del proceso queda de la siguiente manera:

PRM---	Marco	PRM-----	Marco	PRM----	Marco
0 110x xxxx	0x00	5 110x xxxx	0x06	A 110x xxxx	0x0C
1 110x xxxx	0x02	6 110x xxxx	0x07	B 110x xxxx	0x0D
2 110x xxxx	0x03	7 110x xxxx	0x09	C 110x xxxx	0x0E
3 110x xxxx	0x04	8 110x xxxx	0x0A	D 110x xxxx	0x12
4 110x xxxx	0x05	9 110x xxxx	0x0B	E 110x xxxx	0x15
				F 110x xxxx	0x19

La tabla de páginas de este proceso se ubica en el marco 8, desplazamiento 0. Por tanto, un volcado de la memoria principal sería el siguiente: (Suponemos que las x son 0).

Dir.MP	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x080	CO	00	CO	02	CO	03	CO	04	CO	05	CO	06	CO	07	CO	09
0x081	CO	0A	CO	0B	CO	0C	CO	0D	CO	0E	CO	12	CO	15	CO	19

- Asigna un espacio en memoria física para todo el proceso.

Obsérvese cómo queda la distribución de la memoria principal con los marcos ocupados con las páginas del proceso.

M00	P00	M07	P06	M0E	POC	M15	POE
M01	BCP	M08	TDP	M0F		M16	
M02	P01	M09	P07	M10		M17	
M03	P02	M0A	P08	M11		M18	
M04	P03	M0B	P09	M12	POD	M19	POF
M05	P04	M0C	POA	M13		M1A	
M06	P05	M0D	POB	M14		M1B	...

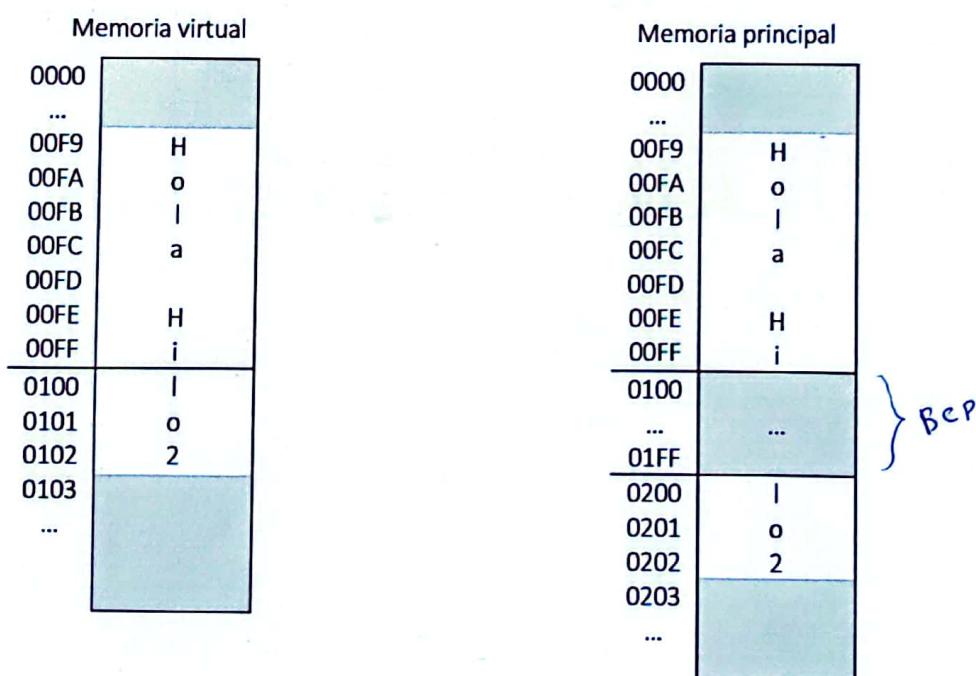
(Los marcos rayados, son marcos ocupados por otros procesos)

3. Indique cuál sería el mapa de bits resultante.

M0..M3	M4..M7	M8..M0B	M0C..M0F	M10..M13	M14..M17	M18..M1B	M1C..M1F	M20..M23	M24..M27	...	MFF
0000	0000	0000	0000	0000	0000	0000	0011	1111	1000	...	0

En el instante t+10, el H1 escribe "Hola hilo 2" a partir de la dirección 0x00F9 de memoria virtual.

4. Indique el contenido y las posiciones de memoria virtual y de memoria principal que se ven modificadas.



5. Realice la ejecución paso a paso del algoritmo.

Aplicando FIFO 2<sup>a</sup> oportunidad.

0x0A FP	0x0B FP	0x06 FP	0x0A A	0x0C FP	0x06 A	0x0A FP	0x0B FP	0x0C A	0x01
'0A →	'0A →	→'0A '0B →	→'0A '0B →	'0C →'0B '06 →	'0C →'0B '06 →	'0C →'0A '06 →	→'0C '0A '0B →	→'0C '0A '0B →	'0C →'01 '0B →