APELLIDOS		NOMBRE		Grupo
DNI		Firma		

- **No desgrape las hojas.**
- **Conteste exclusivamente en el espacio reservado para ello.**
- **Utilice letra clara y legible. Responda de forma breve y precisa.**
- **El examen consta de 8 cuestiones, cuya valoración se indica en cada una de ellas.**

---

1. Sea un sistema que gestiona una memoria de 1200KBytes con asignación contigua con particiones variables. La política de asignación de huecos es Worst Fit (peor hueco) con compactación. El algoritmo de compactación minimiza desplazamientos en memoria hasta encontrar un hueco suficiente para ubicar el proceso que solicita memoria en cada instante, optando siempre por desplazar procesos hacia las direcciones más altas. Considere la ocupación que se muestra (Estado Inical) en la siguiente tabla:

0 1200KB-1

SO 80KB	P1 160KB	HUECO 400KB	P2 120KB	HUECO 440KB

(1,0 puntos = 0,6 +0,4)

a)Gestione la memoria, aplicando compactación si es necesario, e indique la dirección base asignada a cada proceso al producirse de manera secuencial, una tras otra, las nuevas solicitudes descritas a continuación:

Dirección Base	Estado Inicial	Llega P3 (280K)	Llega P4 (200K)	Llega P5 (280K)	Llega P6 (80K)
Base de P1	80K	80K	80K	80K	80K
Base de P2	640K	640K	640K	1040K	1080K
Base de P3	----	760K	760K	760K	800K
Base de P4	----	---	240K	240K	240K
Base de P5	----	----	----	440K	440K
Base de P6	----	----	----	…...	720K

b) Indique de forma justificada qué tipo de fragmentación aparece en particiones variables y si se da en algún momento del apartado a) estime la cantidad

Fragmentación externa, ya que el tamaño de la partición se ajusta al tamaño del bloque dejando huecos en memoria, cuando los procesos finalizan, que pueden no ser lo suficientemente grandes para ubicar un proceso contiguo pero que tras la compactación el proceso puede ser ubicado.
Al intentar ubicar P5 en memoria hay dos huecos en memoria, ninguno con capacidad suficiente:
Hueco 200Bytes en el Byte 440 y Hueco 160Bytes en la Byte 1040→ Total Fragmentación externa 360Bytes.
Al intentar ubicar P6 en memoria hay dos huecos en memoria, ninguno con capacidad suficiente: Hueco 40Bytes en el Byte 720 y Hueco de 40Bytes en la Byte 1160→ Total Fragmentación externa 80Bytes.

2. Respecto a la compartición de páginas en un sistema con paginación, indique si son verdaderas (V) o falsas (F) cada una de las siguientes afirmaciones: (Nota: Un error penaliza una respuesta correcta)

(0,8 puntos)

2	SENTENCIA	V/F
	Existen descriptores de página pertenecientes a procesos diferentes que contienen el mismo número de marco los cuáles corresponde a páginas compartidas	V
	La técnica de copy-on-write, permite compartir páginas con permisos de escritura	F
	Los procesos que comparten páginas ubicadas en memoria, comparten también la Tabla de páginas	F
	Una página cargada en memoria con permisos de su región r-xp puede ser compartida por varios procesos	V
	La probabilidad de llegar a una situación de hiperpaginación se incrementa al aumentar el número de páginas compartidas entre procesos	F

3. Sea un sistema con direcciones físicas y lógicas de 24 bits y páginas de 4KB que gestiona la memoria virtual mediante paginación. Los marcos libres se asignan en orden creciente y utiliza un algoritmo de reemplazo LRU con ámbito LOCAL.

(1.2 puntos = 0,3 + 0,9)

3

a) Calcule la serie de referencias para la siguiente secuencia de direcciones lógicas (en hexadecimal) emitidas por los procesos A y B durante su ejecución: A:40000, A:40014, B:80000, B:80024, B:60030, A:60034, B:80280, B:4060c, A:20000, A:40c10, B:60f24

A:40 B:80 B:60 A:60 B:80 B:40 A:20 A:40 B:60

b) Teniendo en cuenta que el sistema asigna los marcos 0 y 1 al proceso A y los marcos 2, 3, 4 al proceso B y que inicialmente dichos marcos están vacíos. Complete la siguiente tabla con la evolución del contenido de la memoria principal para la serie de referencias obtenida en el apartado anterior e indique el número de fallos de página que produce

Marco	A:40	B:80	B:60	A:60	B:80	B:40	A:20	A:40	B:60
0 (A)	A:40	A:40	A:40	A:40	A:40	A:40	A:20	A20	A:20
1 (A)				A:60	A:60	A:60	A60	A:40	A:40
2 (B)		B:80	B:80	B:80	B:80	B:80	B:80	B:80	B:80
3 (B)			B60	B:60	B:60	B:60	B:60	B:60	B:60
4 (B)						B:40	B:40	B:40	B:40

Número de Fallos de página: 7 (2 con reemplazo)

Nota: los fallos de página están indicados con el texto subrayado, el guión indica que la página se mantiene, y la página sin subrayar un acceso.

4. Sea un sistema con paginación por demanda, dos niveles de paginación con 16 descriptores de página de primer nivel, direcciones lógicas y físicas de 24 bits, y páginas de 4Kbytes. Dicho sistema asigna 3 marcos a cada proceso y está ejecutando el proceso A. En el instante t=10, el proceso A tiene ocupado los marcos 0xf00, 0xf01 y 0xf02, por las páginas 0x001, 0xf2f y 0x11a respectivamente.

(1,6 puntos = 0,4 + 0,4 + 0,8)

4

a) Describa la estructura de las direcciones lógicas y direcciones físicas de este sistema, así como el tamaño en bits de cada uno de los elementos que la componen.

Direcciones lógicas:

- 4 bits para el número de página de primer nivel (16 descriptores)
- 8 bits (24-12-4) para referenciar la entrada a la tabla de segundo nivel
- 12 bits para el desplazamiento (páginas de 4KB).

```
-b23--------------------b19---------b12-b11----------------b0-
|  P1 1er nivel (4 bits) |  p2 (8 bits) | desplazam. (12 bits) |
-------------------------------------------------------------
```

Direcciones físicas: 12 bits para el número de marco + 12 bits para el desplazamiento

```
-b23---N° marco (12 bits)---b12-b11---Desplazam. (12 bits)---b0-
```

b) Indique razonadamente el contenido de los descriptores de página con el bit de validez a válido (v=1) para el proceso A en el instante t=10.

Con dos niveles de paginación son necesarios dos tablas de páginas, una de primer nivel y otra de 2º nivel, para traducir direcciones lógicas a físicas. Todas las páginas tendrán su bit de validez a 0 excepto las que están en memoria en t=10.

páginas 0x01→ descriptor 0 de primer nivel , descriptor 1 de segundo nivel, marco f00, v=1

páginas 0xf2f→ descriptor f de primer nivel , descriptor. 2f de segundo nivel,marco f01, v=1

páginas 0x11a→ descriptor 1 de primer nivel , descriptor 1a de segundo nivel, marco f02, v=1

Hay tres descriptores de página con el bit de validez a 1, cada uno de ellos pertenece a una página diferente de 2º nivel: en concreto pertenecen a las páginas 0,f y 1.

c) Complete la siguiente tabla indicando de forma razonada las correspondientes direcciones lógicas o físicas, que podrían haber sido emitidas o accedidas para el proceso A en el instante T=10

Direc. Lógica	Dirección física	¿Es posible?, Justificación
f2ff02	0xf01f02	Si es posible, corresponde a un num. marco f01 y desplazamiento f02
11af01	0xf02f01	Si es posible, corresponde a un num. marco f02 y desplazamiento f01
0x11a13b	f0213b	La dirección lógica 11a13b corresponde a a páginas 11a que se encuentra ubicada en el marco f02
0x13b11a	Fallo de página	La página 13b no está en memoria y por tanto al referenciarla daría fallo de página y tendría que aplicarse un algoritmo de búsqueda de víctima

5. Complete el programa en código en C del apartado a) con las primitivas POSIX necesarias (una en cada línea con número subrayado) para que un proceso padre lea el contenido del archivo "message.txt" y envíe 2 copias concatenadas de este a su proceso hijo a través de un tubo. El proceso hijo reenviará el contenido que reciba de este tubo a la salida estándar mediante el programa cat (Nota: la orden cat sin argumentos escribe en la salida estándar todo lo que lee de la entrada estándar).

(1,4 puntos= 0,8 + 0,6)

5 **a)**

```
1   #include <all_needed>
2   #define SIZE 80
3   #define MODE O_RDONLY
4   int main( int argc, char **argv ){
5     int fd, fdp[2], readbytes, ncopy;
6     char buffer[SIZE];
7     pipe(fdp);
8     if (fork() == 0){ // child
9          dup2(fdp[0],0);
10         close(fdp[0]);
11         close(fdp[1]);
12         execlp("cat","cat", NULL);
13    }
14    else { // parent
15         close(fdp[0]);
16         fd= open("message.txt", MODE);
17         for(ncopy=0; ncopy<2; ncopy++){
18           while((readbytes= read(fd, buffer, SIZE)) > 0){
19             write(fdp[1], buffer, readbytes        ); /*complete*/
20           }
21          lseek(fd, 0, 0     ); /*complete*/
22         }
23         close(fdp[1]);
24         close(fd);
25    }
26    wait(NULL);
27    exit(0);
28  }
```

b) Rellene la tabla de descriptores de archivos del proceso hijo, en el momento en que se ejecuta la línea 12. Haga lo mismo para el proceso padre en la línea 17. Las tablas deben ser correctas para cumplir con los requisitos y la implementación de la sección a)

Tabla del Poceso. hijo en 12	
0	"fdp[0]"
1	STDOUT
2	STDERR
3	
4	
5	

Tabla del Proceso. padre en 17	
0	STDIN
1	STDOUT
2	STDERR
3	"mensaje.txt"
4	"fdp[1]"
5	

6. Dado el siguiente listado de un directorio en un sistema POSIX:

```
i-nodo   permisos  enlaces usuario  grupo tamaño     fecha     nombre
37093377 drwxr-xr-x  3       marta   disca   4096 dic 11 11:57 .
32448485 drwxr-xr-x  3       marta   disca   4096 dic 11 12:02 ..
32448767 -rwsr-xr-x  1       marta   disca 141528 dic 11 10:31 cp2
33373385 dr-xrwxr-x  2       marta   disca   4096 dic 11 12:02 dir1
32448804 lrwxrwxrwx  1       marta   disca      4 dic 11 10:35 dir2 -> dir1
32448793 -r--r--r--  1       marta   disca    337 dic 11 10:33 f1
32448802 -rw-r--r--  3       marta   disca    402 dic 11 10:33 f2
32448802 -rw-r--r--  3       marta   disca    402 dic 11 10:33 f3
32448803 lrwxrwxrwx  1       marta   disca      2 dic 11 10:34 f4 -> f3
```

(1,4 puntos = 0,8 + 0,6)

6

a) El programa `cp2` es similar a la orden cp de linux. Cuando el último parámetro de cp2 es un directorio copia los archivos indicados en dicho directorio (creándose si no existen). Asuma que el directorio dir1 está vacío, conteniendo sólo las entrada "." y "..", e indique en caso de éxito cuáles son los permisos que se van comprobando y, en caso de error, cuál es el permiso que falla y por qué.

(UID,GID)	ORDEN	¿FUNCIONA?	JUSTIFICACIÓN
(pepe, disca)	cp2 f1 dir1	No	El proceso, al ejecutar cp2 (pepe,disca) pasa a ser (marta, disca) propietaria de f1 y de dir1, como tal puede leer de f1 (ok) pero no puede escribir en dir1 por lo que no podrá crear el archivo dentro (error)
(juan, fso)	cp2 f1 dir2	No	dir2 es un enlace simbólico a dir1 por lo que es necesario permisos de escritura en dir1 para copiar el archivo dentro. Cuando el proceso con identidad (juan,fso) ejecuta cp2 pasa a ser (marta, fso) propietaria de f1 y de dir1, y podrán leer de f1 (ok) pero no puede escribir en dir1 (error)
(ana, alum)	cp2 f1 f2	SI	Cuando el proceso con identidad (ana, alum) ejecuta cp2 pasa a ser (marta, alumn) propietaria de f1 y de f2, como tal puede leer de f1 (ok) y puede escribir en f2 (ok)

b) Justifique el valor de la columna enlaces de las siguientes entradas, indicando cuál es o deduciendo cuál puede ser la razón para que dicha entrada tenga ese número de enlaces

Nombre	Enlaces	Justifique
.	3	El directorio . tiene 3 enlaces que son: la referencia desde su directorio padre, su autorreferencia de la entrada . y la referencia .. del directorio dir1
dir1	2	Este directorio no tiene ningún subdirectorio ya que solo tiene dos enlaces, el de su padre y el propio.
f2	3	Este archivo tiene 3 enlaces: la propia entrada f2, f3 que se puede apreciar que tiene el mismo nodo-i y habrá una tercera entrada en un directorio no mostrado.

7. Un sistema de archivos está organizado en bloques de 512 Bytes, con punteros a bloque de 16 bits. En dicho sistema, existe un archivo "*ejemplo.txt*" de 32KBytes. Para cada uno de los métodos de asignación indicados, calcule el número de accesos a bloques de datos necesarios para leer el Byte 16900 del archivo *ejemplo.txt*. Justifique su respuesta

(1,0 puntos = 0,5 + 0,5)

7	**a)** Asignación Enlazada En asignación enlazada los punteros a bloques se encuentran junto con los datos del propio archivo. Así cada bloque de datos contiene en sus últimos 2 Bytes un puntero al siguiente bloque Con bloques de 512 Bytes contiene: 510 Bytes de Datos y 2 Bytes del siguiente puntero. El Byte 16900, se encuentra en el bloque 33 (trigésimo tercero) de dicho archivo.--> 16900 / 510 = 33 ; 16900 mod 510 =77 Como en en asignación enlazada en cada bloque hay un puntero al siguiente, serán necesarios 34 (bloque 0 al 33) accesos a bloques para leer el Byte 16900
	b) Asignación indexada En asignación indexada se dispone de un bloque que contiene los punteros a bloques de datos. Un bloque de 512 Bytes con punteros de 2 Bytes, pueden contener hasta 256 punteros a bloque Con bloques de 512 Bytes, el Byte 16900, se encuentra en el bloque 33 (trigésimo tercero) de dicho archivo En el caso de asignación indexada, sólo hace falta realizar 2 accesos, uno al bloque índice donde están los punteros a bloque y otro para acceder al bloque 33 que es donde está el byte 16900.

8. Un disco con capacidad de 64 MBytes, se formatea con una versión de MINIX, cuyas especificaciones son las siguientes:

- El bloque de arranque y el superbloque ocupan 1 bloque cada uno
- Tamaño del nodo-i de 32 Bytes (7 punteros directos, 1 indirecto,1 doble indirecto).
- Punteros a zona de 16 bits
- Entrada de directorio de 16 Bytes: 2 Bytes para nodo-i, 14 Bytes para el nombre
- 1 zona = 1 bloque = 1 KByte

(1,6 puntos = 0,3 + 0,8 + 0,5)

8	**a)** Calcule el número máximo de nodos-i que puede tener este sistema. Justifique su respuesta. El máximo número posible de nodos-i está determinado por el tamaño del campo de la entrada de directorio reservada para indicar el nodo-i que son 2 bytes, o sea 16 bits. Por lo tanto el número máximo posible de nodos-i es $2^{16} = 65536$

b) Suponga que el disco se formatea reservando espacio en la cabecera para un total de 32768 nodos-i (32 K nodos-i). Calcule el número de bloques que ocupa cada uno de los elementos de la cabecera y el área de datos. Justifique su respuesta

Arranque	Super bloque	Mapa de bits de Nodos-i	Mapa de bits de Zonas	Nodos- i	Zonas de datos

1 Bloque de arranque → bloque 0
1 Bloque de Super bloque → bloque 1
4 Bloques para el Mapa de Nodos-i → bloque 2,3,4 y 5
32K de nodos-i requieren 32K bits; 32Kbits/ 1KByte= 32Kbits/8KBits= 4 bloques
8 Bloques para el Mapa de Zonas → bloque 6, 7, 8, 9, 10, 11, 12, 13 y 14
Partición de 64 MBytes= 64MBytes/1KByte= 64K zonas → son necesarios 64Kbits
64Kbits/8Kbits= 8 Bloques
1024 Bloques para los Nodos-i→ bloque 16 al bloque 1040
Nodos-i = 32 K nodos-i; 32Knodos-i * 32 Bytes = 1MBytes para almacenar los nodos-i
1MByte/1KByte =1024 bloques para almacenar nodos-i
La zona de datos comienza en el bloque 1041
La cabecera ocupa: 1+1+4+8+1024=1038 bloques
La partición es de 64MBytes → 64 MBytes/1KByte 64KBloque =65536 Bloques
65536-1038 de la cabecera= 64498 zonas de datos

c) Suponga que, en un momento dado, hay un total 32768 nodos-i (32K nodos-i) ocupados en el disco con un único directorio, el directorio raíz, y archivos regulares todos de 1KByte de tamaño. Indique de forma justificada el número de zonas del área de datos que estarán ocupadas en dicha partición.

El nodo-i 0 está reservado para el borrado de archivos.
El nodo- i 1 corresponde al directorio raíz, quedan pues 32766 nodos-i para los archivos regulares.
Cada archivo regular lleva asociada una entrada de directorio, además son necesarias las entradas "." y ".." del directorio raíz → Esto implica que el directorio raíz será un archivo con 32768 entradas.
Cada entrada de directorio ocupa 16 Bytes → Directorio raíz ocupa 32K entradas*16 Bytes = 512KBytes. Este directorio necesitará 512 ptos a Zonas→ utilizará los 7 punteros directos del nodo-i y el puntero indirecto → 1 Zona con punteros a Zonas
Cada archivo ocupa 1 zona, es decir 1KByte, estos archivos no necesitan bloques de punteros ya que se direccionan con el primer puntero directo del nodo-i, la ocupación de los archivos es pues: 32766*1K
Número de zonas ocupas: 512 zonas del raíz con entradas de directorios + 1 para el directorio raíz con punteros a Zonas + 32766 zonas de archivos regulares=33279 zonas