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 7 cuestiones, cuya valoración se indica en cada una de ellas.**

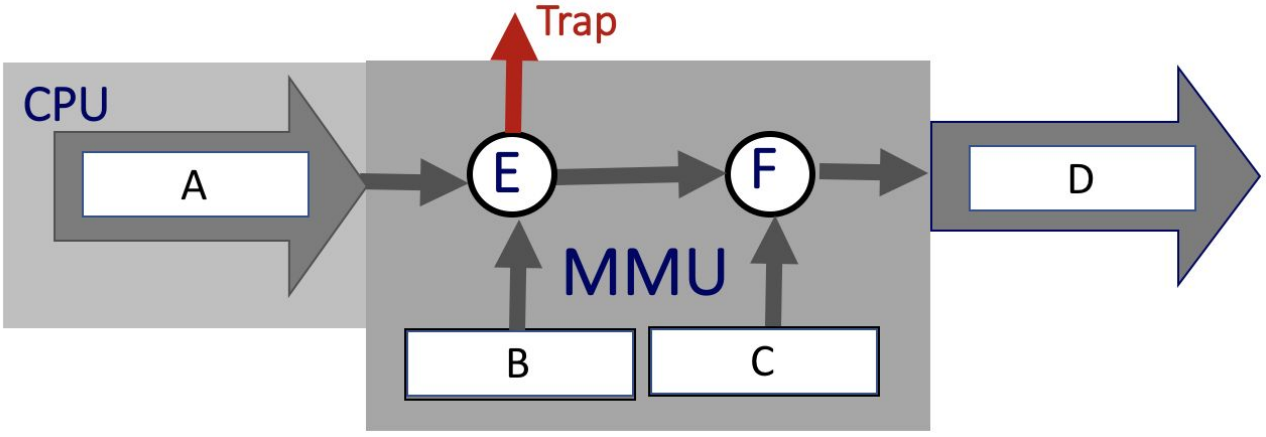
1. Indique, cuáles de las siguientes afirmaciones sobre el mapa de memoria de un proceso en un sistema Linux de 32 bits son verdaderas y cuáles son falsas. (Nota: Un error penaliza un respuesta correcta)

(1,0 puntos)

1	**Afirmaciones**	**V/F**
	La región de datos con valor inicial se puede identificar comprobando que tiene permisos de rw-p y que tiene soporte en el fichero ejecutable.	V
	Al crear una variable local, y para poder alojar la variable en memoria, se crea o incrementa el tamaño de la región de heap.	F
	Si se utiliza un enlace de librería estático, la información de las bibliotecas aparece en regiones diferenciadas del mapa de memoria del proceso.	F
	Si mostramos el valor del puntero de una variable declarada dentro de la función main, éste estará ubicado en la región de datos no inicializados.	F
	El uso de ficheros mapeados en memoria mejora el tiempo de acceso a la información del fichero.	V

2. La siguiente imagen representa el funcionamiento básico de una MMU (Memory Management Unit), donde se han eliminado el nombre de ciertos elementos, y se han sustituido por las letras de la A a F. Complete en la tabla el nombre de estos elementos en la columna NOMBRE. En la columna VALORES, considerando que el elemento correspondiente al elemento A es 1000, sitúe correctamente los siguiente valores: 200, 1200 y 1500, en las casillas de la B a la D, suponiendo que la dirección emitida es correcta.

(1,0 puntos)



2	

ELEMENTOS	NOMBRE	VALORES
A	Dirección lógica	1000
B	Registro límite	1500
C	Registro base	200
D	Dirección física	1200
E	<	---
F	+	---

3. Un sistema de gestión de memoria basado en paginación de doble nivel que permite hasta 16 descriptores de primer nivel y tablas de 256 descriptores de página de segundo nivel. El tamaño de página es de 4Kbytes, cada descriptor de página (de 1º o 2º nivel) ocupa 16 bytes. Se puede direccionar una memoria principal de 16Mbytes. Responda de forma razonada a los siguientes apartados:

(1.5 puntos = 0.5 + 0.25 + 0.75)

3

a) Formato de la dirección lógica y física con nombre y número de bits de cada campo o elemento

DL de 24 bits → compuesta por 12 bits para desplazamiento, 8 bits para el nº de página de segundo nivel y 4 bits para el identificar el descriptor de primer nivel.16Mbytes → 24 bits para DF:24-12 = 12 bits para núm. de marco

Dirección lógica (nombre y número de bits de cada elemento)

Nº pag.1ºn (4bits)	*2ºn (8 bits)*	*Desplazamiento (12 bits)*

b23 *b0*

Dirección física(nombre y número de bits de cada elementos)

Nº Marco (12bits)	*Desplazamiento (12 bits)*

b23 *b0*

b) Indique en hexadecimal el rango de direcciones lógicas asociado a las primeras 32 páginas de un proceso en dicho sistema. Utilice para su respuesta el formato [primera dirección, última dirección]

[0x000000,0x01FFFF] La primera dirección lógica corresponde al inicio de la página 0 con lo que el valor en hexadecimal será 0x000000.La última página en el rango solicitado será la nº 31 con lo que habrá que obtener su valor equivalente en hexadecimal (1F) y la última dirección de dicha página corresponderá a valor máximo de desplazamiento. En definitiva tendremos 0x01FFFF como última dirección lógica en el rango solicitado.

c) Justifique cuántos descriptores de 1º y 2º nivel requiere un proceso de 2K páginas. Asimismo, calcule el número total de páginas ocupadas por las tablas de 2º nivel de dicho proceso.

Nº de descriptores de 1º nivel:

Para determinar el nº de descriptores de primer nivel dividiremos las 2K páginas asignadas al proceso entre el número de descriptores de 2º nivel (256) con el resultado de 8 descriptores.

Nº de descriptores de 2º nivel:

El nº de descriptores de 2º nivel corresponderá a las 2K páginas, es decir 2048

Nº de páginas ocupadas por las tablas de 2do nivel:

Cada uno de estos descriptores ocupa 16 bytes con lo que tenemos 32Kbytes de espacio requerido y que dividido por el tamaño de página (4K) necesitará 8 páginas para su almacenamiento.

4. En un sistema con memoria virtual y paginación bajo demanda se ejecutan actualmente los procesos A y B. Para dichos procesos, el sistema dispone de un total de 5 marcos. La siguiente tabla describe la evolución de la asignación de marcos a las páginas demandadas por ambos procesos desde el instante t=120 hasta t=130. Cada celda detalla el proceso implicado y el número de página accedida (en hexadecimal), así como el valor del bit de referencia. En la primera columna, para el estado de partida, se indican entre paréntesis los tiempos de la última referencia a la página.

	t= 120		**t= 121**		**t= 122**		**t= 123**		**t= 124**		**t= 125**	
↓marco / pág.→	**B:d34**		**B:d34**		**B:c08**		**B:c08**		**A:002**		**A:555**	
c0	A:36a (t=25)	1	A:36a	1	A:36a	1	A:36a	1	A:36a	1	A:36a	1
c1	A:450 (t=50)	0	A:450	0	A:450	0	A:450	0	A:450	0	A:555	1
c2	B:d34	1	B:d34	1	B:d34	1	B:d34	1	B:d34	1	B:d34	1
c3					B:c08	1	B:c08	1	B:c08	1	B:c08	1
c4									A:002	1	A:002	1

	t= 126		**t= 127**		**t= 128**		**t= 129**		**t= 130**	
↓marco / pág.→	**A:555**		**B:3af**		**B:4b0**		**B:d00**		**A:0fe**	
c0	A:36a	1	A:36a	1	A:36a	1	A:36a	1	A:36a	0
c1	A:555	1	A:555	1	A:555	1	A:555	1	A:555	0
c2	B:d34	1	B:3af	1	B:3af	1	B:d00	1	B:d00	1
c3	B:c08	1	B:c08	0	B:4b0	1	B:4b0	0	B:4b0	0
c4	A:002	1	A:002	1	A:002	1	A:002	1	A:0fe	1

Partiendo de la información de la tabla, conteste cada apartado:

(2 puntos = 0.5 + 0.5 + 0.5 + 0.5)

4

a) Señale si el algoritmo de reemplazo usado ha sido el de 2ª oportunidad o el LRU y su ámbito (local/global). En caso de descartar alguno de los dos, detalle el primer instante de tiempo en el que la evolución no se corresponde con el algoritmo descartado. Detalle también el primer instante de tiempo que le ha permitido discriminar el ámbito local/global.

La evolución se corresponde a un algoritmo de 2ª oportunidad con ámbito local. No puede ser un LRU porque en el instante t=125 no se ha elegido como víctima a la página menos recientemente accedida, que era la A:36a con último acceso en t=25 sino a la A:450 que había sido accedida por última vez en t=50. En el instante t=127 se demuestra que el criterio de reempazo es local, en caso de ser global en ese momento habrían quedado a 0 todos los bits de referencia, excepto el de la página reemplazada.

b) Si en los instantes t=131 y t=132 se accediera a las páginas A:36a y A:070 respectivamente, completar los siguientes elementos como continuación de la tabla anterior:

t= 131		**t= 132**	
A:36a		**A:070**	
A:36a	1	A:36a	0
A:555	0	A:070	1
B:d00	1	B:d00	1
B:4b0	0	B:4b0	0
A:0fe	1	A:0fe	1

c) Suponiendo que se maneja un esquema de área activa, con un tamaño de ventana de 4, determine las áreas activas de los procesos A y B tras completarse el acceso del instante t=130.

AreaActiva(A)= {002, 555, 0fe}

AreaActiva(B)= {c08, 3af, 4b0, d00}

d) Justifique si en el instante t=130 se está ante un escenario de hiperpaginación para los procesos A y B, según el criterio de área activa del apartado anterior y suponiendo una gestión global de marcos.

Se considera que se producirá hiperpaginación si no existe un número de marcos suficiente para albergar las áreas activas de los procesos. Para los procesos A y B se dispone de 5 marcos, pero la suma del tamaño de sus áreas activas es TAA(A) + TAA(B) = 3 + 4 = 7, que es mayor de 5, por lo que sí que habrá hiperpaginación.

5. Teniendo en cuenta los mecanismos de herencia de los procesos en Unix y las llamadas POSIX, responda a los siguientes apartados:

(1.5 puntos = 0.5 + 1.0)

5

a) Considere que la orden : `$cat < proc.c | grep "for" > loop.txt`
debe ejecutarse y finalizar correctamente. Dibuje la tabla de descriptores de archivos abiertos para cada uno de los procesos que intervienen en ella, indicado el contenido de los descriptores no vacíos.
Nota. Denominar al tubo como "fdpipe"

Tabla del proceso "cat"	
0	"proc.c"
1	fdpipe[1]
2	STDERR

Tabla del proceso "grep"	
0	fdpipe[0]
1	"loop.txt"
2	STDERR

b) Escriba un programa en C con las instrucciones y llamadas al sistema necesarias para que se llegue a ejecutar la siguiente línea de órdenes: `$ \bin\ls | \bin\grep ".c" > my_programs`
Nota: Suponga que no se producen errores en las llamadas al sistema.

```
/** Prog.c:   $\bin\ls | \bin\grep ".c" > my_programs **/
#include <unistd.h>
#define newfile (O_WRONLY | O_CREAT | O_TRUNC)
#define mode644 (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)

int main() {
    int fdpipe[2];
    int fd;

    pipe(fdpipe);

    if (fork()) {

        dup2(fdpipe[1], STDOUT_FILENO);
        close(fdpipe[0]); close(fdpipe[1]);
        execl("/bin/ls", "ls", NULL);


    } else {
        fd = open("my_programs", newfile, mode644);

        dup2(fdpipe[0], STDIN_FILENO);
        dup2(fd, STDOUT_FILENO);
        close(fdpipe[0]); close(fdpipe[1]); close(fd);
        execl("/bin/grep", "grep", ".c", NULL);
    }

    return 0;
}
```

6. Dado el listado del directorio ***XXXX/*** (cuyo nombre desconocemos) en un sistema POSIX:

```
i-nodos  permisos enlaces usuario grupo  tamaño  fecha     nombre

13504322 drwxrwxr-x  3   aperez  disca   4096 dic 21 19:33 .
12200756 drwx------  5   aperez  disca   4096 dic 21 19:20 ..
13504405 drwxrwxr-x  2   aperez  disca   4096 dic 21 19:29 back
13504425 -rwxr-xr-x  1   aperez  disca 151024 dic 21 19:33 cp
13504425 -rwsr-xr--  1   aperez  disca 151024 dic 21 19:33 cp2
13504417 -rw----r--  1   aperez  disca   6364 dic 21 19:22 dat1
13504421 -rw-rw-r--  3   ana     fso     1134 dic 21 19:25 dat2
13504421 -r--r--r--  3   juan    aic     1134 dic 21 19:25 dat3

./back:
13504405 drwxrwxr-x  2   aperez  disca   4096 dic 21 19:29 .
13504322 drwxrwxr-x  3   aperez  disca   4096 dic 21 19:33 ..
13504420 lrwxrwxrwx  1   manolo  disca      7 dic 21 19:29 old1 -> ../dat1
13504421 -rw-rw-r--  3   maria   fso     1134 dic 21 19:25 old2
```

(1.5 puntos = 1 + 0,5)

6

a) Los programas `cp y cp2` son dos copias idénticas del programa del sistema cp que copia el contenido del archivo que recibe como primer argumento en otro cuyo nombre se indica como segundo argumento. Es decir, "`cp a b`" copia el contenido del archivo `a` al archivo `b`, si `b` no existe lo debe crear y después realizar la copia. Rellene la tabla indicando, si la orden correspondiente funciona, el EUID y EGID (UID y GID efectivos del proceso al ejecutar la orden) y en caso de error, cuál es el permiso que falla.

(UID,GID)	**ORDEN (desde directorio XXXX/)**	**¿FUN-CIONA?**	**(EUID,EGID)**	**Si falla: permisos(s) que FALTAN**
(ana,fso)	cp dat1 dat4	NO	(ana,fso)	W en XXXX/
(ana, disca)	cp2 dat1 back/old3	SI	(aperez,disca)	
(ana,disca)	cp back/old1 dat2	NO	(ana,disca)	R de old1->../dat1
(maria,fso)	cp dat3 ..	NO	(maria,fso)	W en ..

b) Si se ejecuta la orden: `ls -li ..`
y obtenemos:

```
i-nodos  permisos enlaces usuario grupo  tamaño  fecha      nombre
12200756 drwx------   5 aperez aperez  4096 dic 26 11:52 .
10223617 drwxr-xr-x 149 aperez aperez 12288 dic 25 01:10 ..
13764791 drwxrwxr-x   2 aperez aperez  4096 dic 26 11:51 alumnos
13504322 drwxrwxr-x   3 aperez aperez  4096 dic 26 11:51 organizacion
13636461 drwxrwxr-x   2 aperez aperez  4096 dic 26 11:50 practicas
```

¿Cúal es el nombre del directorio ***XXXX***?

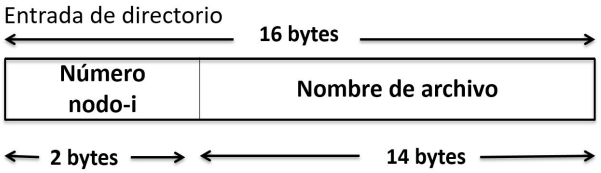
Según el listado inicial, el directorio actual (`./`) está en el nodo-i 13504322. Buscando qué directorio en el nuevo listado tiene asignado este node-i, comprobamos que es el directorio: ***organizacion***

7. Las siguientes figuras hacen referencia a los tamaños y estructuras de los elementos utilizados para formatear una partición MINIX, con la siguiente configuración:

- Nº de nodos-i : 3424
- Tamaño de bloque : 1 KByte
- 1 zona = 1 bloque
- Tamaño de la partición : 10 MBytes

Obsérvese que todos los campos del nodo-i son de 16 bits salvo "Nº de enlaces" y GID que son de 8 bits.

La entrada de directorio tiene el siguiente formato:



(1,5 puntos = 0,5 + 0,2 + 0,8)

a) Número de bloques que ocupa los siguientes elementos de la cabecera: mapa de bits nodos-i, mapa de bits zonas y nodos-i.

```
#bloques mapa bits nodos-i = 3424/(1024*8) < 1 -> 1 bloque
#bloques mapa bits zonas = 10*1024/(1024*8) = 10/8 -> 2 bloques
#bloques nodos-i = 3424*32/1024 = 3424/32 = 107 bloques
```

b) Índice de bloque del primer bloque de la zona de datos.

```
Bloques ocupados = 1 + 1 + 1 + 2 + 107 = 112 -> del 0 al 111
El primer bloque de la zona de datos es el 112
```

c) Justo después de formatear la partición se crea un archivo "message.txt" de 64 KBytes. Sabiendo que el primer bloque asignado al archivo es el 113, y que los bloques se asignan en orden, obtener en qué bloque se encuentra el byte 62459 del archivo "message.txt"

```
Un archivo de 64 KBytes ocupa 64 zonas de datos. Para direccionar dichos
bloques se utilizarán los 7 punteros directos y el puntero indirecto (el
doble indirecto no se usa ya que el indirecto tiene una capacidad de
direccionamiento de 512 KBytes). Al utilizar el puntero indirecto se
utiliza una zona para punteros directos. Así pues el orden de asignación
de las zonas es:

7 zonas datos + 1 zona punteros + 57 zonas de datos

El byte 62459 está en la zona relativa: 62459 / 1024 = 60,995 -> 60
Está pues después de la zona de punteros, será pues la zona relativa
60 + 1 = 61. Considerando que el primer bloque del archivo es el 113, el
bloque absoluto en el que se encuentra el byte 62459 es el:

113 + 61 = 174
```