

?

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 10 cuestiones, cuya valoración se indica en cada una de ellas.
- Recuerde que debe justificar sus cálculos o respuestas para obtener buena calificación

1. Describa los eventos numerados del 3 al 6 de la figura, relativos a la solicitud de una llamada al sistema por parte de un proceso de usuario, e indique en “Modo:” el modo de funcionamiento de la CPU que corresponde a la realización de cada uno de los eventos numerados.

(0,9 puntos)

1	<p>El diagrama ilustra la arquitectura de software y hardware involucrada en una llamada al sistema. Se muestran cuatro capas principales: Application Interface (verde), System Calls (naranja), Operating System (naranja claro) y HAL (Hardware Abstraction Layer) (naranja oscuro). El hardware incluye CPU y MEMORIA. Se muestran eventos numerados del 1 al 6 con líneas de flujo que indican la secuencia de la llamada al sistema. El evento 1 ocurre en la Application Interface, el evento 2 en System Calls, el evento 3 en el Operating System, el evento 4 en el HAL, el evento 5 en el Operating System y el evento 6 en System Calls.</p>	<p><b>EVENTO 1</b> Descripción: Programa de usuario en ejecución Modo: <b>Usuario</b></p> <p><b>EVENTO 2</b> Descripción: Llamada al sistema Modo: <b>Usuario</b></p> <p><b>EVENTO 3</b> Descripción: Identificación de la llamada Modo: <b>Núcleo</b></p> <p><b>EVENTO 4</b> Descripción: Ejecución del código de servicio Modo: <b>Núcleo</b></p> <p><b>EVENTO 5</b> Descripción: Transferencia de resultados Modo: <b>Núcleo</b></p> <p><b>EVENTO 6</b> Descripción: Ejecución de la instrucción siguiente Modo: <b>Usuario</b></p>
---	--	--

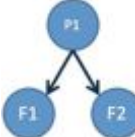
2. Dado el siguiente código cuyo fichero ejecutable es *example1*.

```

1  /** example1***/
2  #include <all_needed.h>
3
4  int main()
5  { int i=0, delay1=1, delay2=1;
6    pid_t pid, pid_x;
7
8    for (i=0; i<2; i++)
9    { pid=fork()
10     if (pid==0)
11     { sleep(delay1);
12       if(execlp("wc","wc","-l","example1.c",NULL)<0)
13         {printf("Error running wc\n");
14           exit(0);}
15       pid_x=fork();
16     }
17   }
18   sleep(delay2);
19   while (wait(NULL) != -1);
20   exit(0);
21 }
```

Suponga que *example1* se ejecuta sin errores e indique de forma justificada:

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

2	<p>a) El número de procesos que se generan al ejecutarlo y dibuje el esquema de parentesco entre procesos.</p> <p>El número de procesos creados al ejecutar ejemplo1 son tres, un padre y dos hijos. Ambos hijos ejecutarán el <code>execlp("wc","wc","-l","example1.c",NULL)&lt;0</code>. El esquema sería el siguiente</p>  <pre> graph TD     P1((P1)) --&gt; F1((F1))     P1 --&gt; F2((F2))   </pre>
	<p>b) Indique de forma justificada la posibilidad de que se generen procesos zombies o huérfanos para cada uno de los siguientes pares de valores de <i>delay1</i> y <i>delay2</i>:</p> <p>b1) <i>delay1</i>=1000; <i>delay2</i>=10;</p> <p>Funcionamiento sin zombies ni huérfanos. Cuando los hijos finalizan el <code>sleep(delay1)</code>, y por tanto su estado de suspendido, el padre ya debe estar esperándolos en el bucle del <code>wait()</code>.</p> <p>b2) <i>delay1</i>=10; <i>delay2</i>=1000;</p> <p>Se generan zombies. Cuanta mayor sea la diferencia más tiempo podrán permanecer los hijos en estado zombie, ya que <i>delay2</i> retrasa al proceso padre para realizar la llamada <code>wait()</code>.</p> <p>Tal y como está estructurado el código, sólo en el caso de algún error o la llegada de una señal que afecte al proceso padre (antes de que acaben los procesos hijos) se podría dar la circunstancia de que los procesos hijos se queden huérfanos.</p>

3. Sea un sistema de tiempo compartido con un planificador de procesos a corto plazo Round Robin, con quantum  $q=2\text{ut}$  y una única cola de preparados. Cuando se producen varios eventos en un mismo instante el orden de inserción en cola de los procesos es: nuevo, procedente de E/S y fin de quantum. La tabla muestra una planificación para los procesos, A, B y C, que llegan en los instantes  $t=0$ ,  $t=3$  y  $t=8$  respectivamente y cuyas ráfagas son:

A ( $t=0$ )	4CPU + 2E/S + 4CPU + 3E/S + 1CPU	B ( $t=3$ )	6CPU + 1E/S + 2CPU	C( $t=8$ )	3CPU+2E/S+1CPU
-------------	----------------------------------	-------------	--------------------	------------	----------------

Por un fallo de implementación, el planificador no funciona según las especificaciones propuesta cometiendo errores. Detecte en la tabla el instante de tiempo (T) en el que ocurre un error por primera vez, indique el motivo de dicho error y rehaga la planificación a partir de ese punto en las columnas paralelas de dicha tabla.

(1,2 puntos=0,9+0,3)

T	Preparados	CPU	Cola E/S	E/S	Preparados	CPU	Cola E/S	E/S	Evento
0	(A)	A							Llega A
1		A							
2		A							
3	B	A							Llega B
4	(B)	B	(A)	A					
5		B		A					
6	B (A)	A							
7	B	A							
8	C A (B)	B			A C (B)	B			Llega C
9	C A	B			A C	B			
10	B C (A)	A			B A (C)	C			
11	B C	A			B A	C			
12	B (C)	C	(A)	A	C B (A)	A			
13	B	C		A	C B	A			
14	C (B)	B		A	C (B)	B	(A)	A	
15	A C	B			C	B		A	
16	A (C)	C	(B)	B	(C)	C	B	A	
17	B (A)	A	(C)	C	(A)	A	C (B)	B	
18	(B)	B		C	(B)	B	(C)	C	Fin A / Fin A
19	C	B				B		C	
20	(C)	C			(C)	C			Fin B / Fin B
21									Fin C / Fin C

	Instante T del error = 8 ó 10 Motivo: La inserción en cola no se hace correctamente en el instante 8, C (proceso nuevo) debe insertarse antes que A (proceso que finaliza el quantum), pero las consecuencias se ven al planificar en el instante 10.			
b)	Indique tiempo de Retorno y Tiempo de Espera para cada proceso			
		A	B	C
	Tiempo Retorno	18	17	13
	Tiempo Espera	4	7	6

4. Dada la siguiente cadena de 52 caracteres correspondiente a un gen:

CACTCAGCACGAA GGGCAGAGGAATG CTTACCGTCCTGA GCCACCCACCAGC

Se desea buscar en qué posiciones se referencia al aminoácido CAG, con un diseño basado en 4 hilos concurrentes. Cada hilo analiza una única fracción de 13 caracteres del gen, de forma que un hilo analiza los 13 primeros caracteres, otro los 13 siguientes, etc. La función “*find\_amino*” recibe un puntero a la posición del primer carácter de la fracción del hilo que tiene asignada y en caso de encontrar el aminoácido en esa fracción, marca el primer carácter donde aparece sobrescribiendo “C” por el carácter “\*”. El programa principal, tras asegurarse de que se han marcado todas las ocurrencias del aminoácido, localiza las posiciones marcadas y las envía a la salida estándar en orden, separadas por un espacio. **(1,0 puntos = 0,5 + 0,25 + 0,25 )**

<pre> 1  #include &lt;all_needed.h&gt; 2  #define NFRAC 4 3  #define GENSIZE 52 4 5  char gen[] = "CACTCAGCACGAAGGGCAGAGGAATG 6  CTTACCGTCCTGAGCCACCCACCAGC"; 7  char amino[] = "CAG"; 8 9  void *find_amino(void *ptr){ 10     char *pgen= (char*) ptr; 11     int i; 12     for (i=0; i&lt;GENSIZE/NFRAC-2; i++) { 13         if (pgen[i] == amino[0] &amp;&amp; 14             pgen[i+1] == amino[1] &amp;&amp; 15             pgen[i+2] == amino[2]) 16             pgen[i] = '*'; 17     } </pre>	<pre> 18 int main() { 19     int i; 20     char *pfrac; 21     pthread_attr_t attrib; 22     pthread_t thread[NFRAC]; 23     pthread_attr_init(&amp;attrib); 24 25     for (i = 0; i&lt;NFRAC; i++) { 26         pfrac= gen + i * GENSIZE/NFRAC; 27 28     /**complete**/ 29 30     for (i = 0; i&lt;GENSIZE; i++) { 31         if (gen[i]=='*') 32             printf("%d ", i); 33     } </pre>
--	---

a) Utilice las variables ya declaradas en el programa y complete las líneas de código que faltan en el espacio marcado “**/\*\*complete\*\*/**”, de forma que se realice la creación y espera de los 4 hilos “*find\_amino*” para que se resuelva el problema de la forma propuesta.

**NOTA.** Definición de funciones: *int pthread\_join(pthread\_t thread, void \*\*retval);*

*int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*(\*start\_routine) (void \*), void \*arg);*

```

25 for (i = 0; i<NFRAC; i++){
26     pfrac= gen + i * GENSIZE/NFRAC;
27     pthread_create(&thread[i], &attrib, find_amino, pfrac);
28 }
29 for (i = 0; i<NFRAC; i++)
30     pthread_join(thread[i], NULL);

```

b) Justifique cuál es el máximo número de hilos pertenecientes a este proceso que llegarán a ejecutarse concurrentemente.

Cinco, el principal (main) + los 4 hilos “*find\_amino*”

c) Razone si la ejecución de este programa puede dar lugar a alguna condición de carrera.

No es posible, porque los 4 hilos acceden a posiciones diferentes de la cadena *gen*, que es la única variable que comparten. Por otro lado, el hilo principal tampoco accede concurrentemente a esta cadena, sino que espera a que los otros 4 hilos hayan terminado sus accesos.

5. Sea el siguiente código de tres procesos concurrentes cuyos valores iniciales de la variable X y semáforos S1, S2 son los siguientes:  $X=1$ ,  $S1=1$ ,  $S2=1$ .

A	B	C
$P(S1);$ $X = X + 1;$ $V(S1);$	$P(S1);$ $V(S1);$ $P(S2);$ $X = X - 1;$ $V(S2);$	$P(S1);$ $P(S2);$ $X = X + 1;$ $V(S2);$

Estos tres procesos se ejecutan en un sistema donde el orden en que se asigna por primera vez el procesador está SIEMPRE determinado por el orden de llegada a la cola de preparados. Tras conseguir la CPU por primera vez SIEMPRE se ejecuta la primera instrucción ( $P(S1)$ ), pero a partir de ese punto pueden producirse cambios de contexto en cualquier momento. Indique de forma justificada: i) Si los tres procesos pueden finalizar; ii) Si es posible que ocurra una condición de carrera y iii) El valor final de X, si los procesos acaban.

Considera las siguientes tres posibles secuencias de llegada:

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

5	<p><b>a) A, B, C</b></p> <p>Los tres procesos terminan.</p> <p>NO hay condición de carrera y <math>X=2</math>.</p> <p>SECUENCIA: Se le asigna la CPU a A que decreuenta S1, con lo que B y C quedarían suspendidos en S1 si ejecutasen <math>P(S1)</math>. A continuación A incrementa X y ejecuta <math>V(S1)</math> incrementado el contador de S1. A continuación B pasa a ejecución B y ejecuta <math>V(S1)</math>, que activa a C. Al estar B y C en preparados y el semáforo S2 inicializado a 1, ambos podrán realizar su operación sobre X en exclusión mutua. Por lo tanto, el valor final será <math>X=2</math>.</p> <p>NO se producen interbloqueos, ya que mientras se está ejecutando A, aunque se produzca un cambio de contexto en la línea <math>X=X+1</math>, los procesos B y C se suspende al ejecutar su primera instrucción <math>P(S1)</math>, y después está garantizada la exclusión mutua en el acceso a la sección crítica en B y C con el semáforo S2.</p>
	<p><b>b) B, A, C</b></p> <p>Los tres procesos terminan.</p> <p>SI es posible condición de carrera.</p> <p>SECUENCIA: Se le asigna la CPU a B que ejecuta <math>P(S1)</math> dejando el semáforo S1. A partir de este momento tanto A como C se quedarían bloqueados si ejecutasen <math>P(S1)</math>. Con lo que B continúa su ejecución con <math>V(S1)</math> poner a uno S1. Con lo que A puede entrar cuando se produzca un posible cambio de contexto ejecutando su sección crítica (operaciones sobre X) y no garantizado el acceso a X de B y A en exclusión mutua.</p> <p>Dado que se puede producir condición de carrera el valor de X no se puede asegurar que sea finalmente 2.</p>

6. Un sistema con 4 GBytes de Memoria Principal utiliza paginación por demanda con un algoritmo LRU con reemplazo LOCAL. El espacio de direccionamiento lógico es de 4 GBytes y el tamaño de página de 4 KBytes. En un momento dado se inicia la ejecución de los procesos A y B, y el sistema asigna 4 marcos (numerados del 0 al 3) para A y B que se reparten según el esquema equitativo, marcos que inicialmente se encuentran vacíos.

(1,6 puntos = 0,2 + 0,8 + 0,6)

6	<p><b>a)</b> Obtenga la serie de referencias de la siguiente secuencia de direcciones lógicas (en hexadecimal):  (A:1A407125) (A:1A4072C9) (A:4FB30190) (B:00400000) (A:1A407C41) (B:1000458F) (A:24B71AFF)  (B:1E861100) (B:1E861ABC) (A:1A407642) (B:0040093C) (A:4FB30456)  A:1A407; A:4FB30; B:00400; A:1A407; B:10004; A:24B71; B:1E861; A:1A407; B:00400; A:4FB30</p>
---	---

**b)** Indique la evolución del contenido de memoria principal para la serie de referencias obtenida en el apartado a). Los marcos se asignan en orden creciente según demanda. Indique también los fallos de página producidos.

Marco	A:1A407 t=0	A:4FB30 t=1	B:00400 t=2	A:1A407 t=3	B:10004 t=4	A:24B71 t=5	B:1E861 t=6	A:1A407 t=7	B:00400 t=8	A:4FB30 t=9
0	A:1A407 *(0)	A:1A407 (0)	A:1A407 (0)	A:1A407 (3)	A:1A407 (3)	A:1A407 (3)	A:1A407 (3)	A:1A407 (7)	A:1A407 (7)	A:1A407 (7)
1		A:4FB30 0*(1)	A:4FB30 (1)	A:4FB30 (1)	A:4FB30 (1)	A:24B71 *(5)	A:24B71 (5)	A:24B71 (5)	A:24B71 (5)	A:4FB30 * (9)
2			B:00400 *(2)	B:00400 (2)	B:00400 (2)	B:00400 (2)	B:1E861 *(6)	B:1E861 (6)	B:1E861 (6)	B:1E861 (6)
3					B:10004 *(4)	B:10004 (4)	B:10004 (4)	B:10004 (4)	B:00400 * (8)	B:00400 (8)

FALLOS DE PÁGINA TOTALES :

8 ( 4 de ellos con reemplazo)

**c)** Tras la serie de referencias del apartado b) se generan dos nuevas direcciones lógicas: (A:1A407FFF) (B:1E861008). Indique de forma justificada qué direcciones físicas generará la MMU para cada una de ellas:

(A:1A407FFF) -> 00000FFF

(B:1E861008) -> 00002008

7. Teniendo en cuenta el mecanismo de herencia de procesos en Unix y las llamadas POSIX, responda a los siguientes apartados: **(1.3 puntos =0.6+0.7)**

7 a) Rellene la tabla de descriptores de archivos abiertos, indicando el contenido de los descriptores no vacíos, durante la ejecución de cada uno de los procesos que intervienen en la siguiente orden:

```
$ ls -l 2> err | tee f1 | grep ".c" > f2
```

La orden debe finalizar y ejecutarse correctamente. Recuerde que la orden **tee** escribe su entrada estándar tanto en el archivo que se le pasa como parámetro como en la salida estándar.

**Nota.** Denomine al primer tubo "pipeA" y al segundo "pipeB"

ls	tee	grep
0 <u>STDIN</u>	0 <u>pipeA[0]</u>	0 <u>pipeB[0]</u>
1 <u>pipeA[1]</u>	1 <u>pipeB[1]</u>	1 <u>"f2"</u>
2 <u>"err"</u>	2 <u>STDERR</u>	2 <u>STDERR</u>
3	3 <u>"f1"</u>	3

b) Suponga que no se producen errores en las llamadas al sistema y complete el siguiente programa en C con las instrucciones y llamadas al sistema necesarias (una en cada línea con número subrayado) para que se llegue a ejecutar la siguiente línea de órdenes: `$ ls -l 2> err | tee f1 | grep ".c" > f2`

```

1. #include <unistd.h>
2. #include <fcntl.h>
3. #define readfile O_RDONLY
4. #define newfile (O_RDWR | O_CREAT | O_TRUNC)
5. #define mode644 (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)
6. int main() {
7.     int pipeA[2], pipeB[2];
8.     int fd;
9.     pipe(pipeA);
10.    if (fork()) {
11.        fd = open("err", newfile, mode644);
12.        dup2(fd, STDERR_FILENO);
13.        dup2(pipeA[1], STDOUT_FILENO);
14.        close(pipeA[0]); close(pipeA[1]); close(fd);
15.        execlp("ls", "-l", NULL);
16.    } else {
17.        pipe(pipeB);
18.        if (fork()) {
19.            dup2(pipeA[0], STDIN_FILENO);
20.            dup2(pipeB[1], STDOUT_FILENO);
21.            close(pipeA[0]); close(pipeA[1]);
22.            close(pipeB[0]); close(pipeB[1]);
23.            execlp("tee", "tee", "f1", NULL ); /*complete*/
24.        } else {
25.            close(pipeA[0]); close(pipeA[1]);
26.            fd = open("f2", newfile, mode644 ); /*complete*/
27.            dup2(fd, STDOUT_FILENO);
28.            dup2(pipeB[0], STDIN_FILENO);
29.            close(pipeB[0]); close(pipeB[1]); close(fd);
30.            execlp("grep", "grep", ".c", NULL);
        }
    }
    return 0;
}

```

8. Dado el siguiente listado del contenido de un directorio en un sistema POSIX:

i-nodo	permisos	enlaces	usuario	grupo	tamaño	fecha	nombre
32448485	drwxrwxr-x	2	pep	alumne	4096	ene 8 11:57	.
1	drwxr-xr-x	11	pep	alumne	236871	ene 8 12:02	..
32448793	-rw-rw----	1	pep	alumne	310	ene 9 10:37	f1
32448802	--w-rw-r--	3	pep	alumne	343	ene 9 11:15	f3
32448805	-rw----r--	3	pep	alumne	343	ene 9 10:33	f4
32448752	-rwxrwxrwx	1	pep	alumne	5824	ene 9 15:17	f5
33373385	-r-sr-xr-x	1	ana	alumne	706	ene 9 10:35	cp1
32448804	lrwxrwxrwx	1	ana	alumne	8	ene 9 10:36	cp2 ->cp1
32448803	lrwxrwxrwx	1	ana	profes	8	ene 9 10:40	f2 ->f1

Donde los programas cp1 y cp2 requieren el nombre de dos archivos como argumentos. Las órdenes cp1 y cp2 fich1 fich2, tienen como resultado que el contenido de fich1 se copia en fich2 y si fich2 no existe entonces se crea..

(1,0 puntos = 0,75+ 0,25)

8

Rellene la tabla e indique en caso de éxito cuales son los permisos que se comprueban y, en caso de error, cuál es el permiso que falla y porqué.

Usuario	Grupo	Orden	¿Funciona?
ana	profes	./cp1 f1 f3	NO
<b>Justifique</b> El proceso con identidad (ana, profes) tiene permiso de ejecución para ./cp1. El proceso mantiene su identidad (ana,profes) por lo que no tendrá permiso para leer el fichero f1, ya que consulta la tercera tripleta de permisos.			
pep	alumne	./cp2 f3 f2	SI
<b>Justifique</b> Como ./cp2 es un enlace a ./cp1 los permisos que mira el proceso (pep,alumne) son los de ./cp1. Puede ejecutarlo pasando a tener la identidad de (ana,alumne). Consulta la segunda tripleta de f3 y tiene permiso de lectura. Y para escribir en f2, como es un enlace a f1 se consulta la segunda tripleta de f1 y tiene permiso para escribir.			
pep	disca	./cp1 f4 f6	NO
<b>Justifique</b> El proceso con identidad (pep,disca) consulta los permisos de la tercera tripleta y sí puede ejecutar ./cp1, pasando a tener la identidad (ana,disca). Tiene, por tanto, permisos de lectura en f4 (tercera tripleta) pero no tiene permisos de escritura en el directorio, para crear el fichero f6 (tercera tripleta)			

b) Justifica adecuadamente qué tipo de archivo es **f2** e indique la lista de i-nodos a los que será necesario acceder para leer el contenido de dicho archivo.

El fichero f2 es un enlace simbólico a f1 ya que no comparten el mismo i-nodo. Por ser un enlace simbólico, para leer el contenido de f2 tenemos que acceder primero al bloque de datos de f2, donde nos dice la ruta para llegar al contenido de f1. Los i-nodos consultados serán: 32448803, 32448485, 32448793 (podría acceder también al 1 si la ruta fuese absoluta)



9. Un sistema de archivos Minix de 512MBytes tiene las siguientes características:

- Nodos-i con un tamaño de 32 Bytes con 7 punteros directos a zonas, 1 indirecto y 1 doblemente indirecto
- Punteros a zona de 16 bits (2Bytes)
- Entradas de directorio de 16 Bytes (14 Bytes para el nombre y 2 Bytes para el nodo-i)
- **1 Bloque = 1 Zona = 2 KBytes**

(1,0 puntos = 0.6+0.4)

9 a) Indique de forma justificada como quedarían los tamaños de cada elemento de la cabecera al formatearlo para almacenar 16384 nodos-i (16K nodos).

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

1 Bloque de Arranque, 1 Super Bloque, 1Bloque Mapa Nodos-i, 16 Bloque Mapa zonas y 256 Bloques para los Nodos-i.

Calculos:

MapaNodos-i  $\rightarrow 16384 / 2K * 8 = 2^4 2^{10} / 2^1 2^{10} 2^3 = 1 \rightarrow 1$  Bloque

Nº máximo de zonas =  $512 \text{ MB} / 2K = 2^9 2^{20} / 2^1 2^{10} = 256 \text{ K}$  zonas

Mapa Zonas  $\rightarrow \text{Nº de Zonas} / \text{Tamaño en bits de la zona} = 256 \text{ K} / 2K * 8 = 2^8 2^{10} / 2^1 2^{10} 2^3 = 2^4 = 16$  Bloque

Bloques Nodos-i  $\rightarrow 16384 * 32 \text{ Bytes} / 2K\text{Bytes} = 2^4 2^{10} 2^5 / 2^1 2^{10} = 256$  Bloques

b) Calcule y justifique el tamaño mínimo de zona con que se debería formatear una partición de 512MBytes si se diseña el nodo-i con únicamente 8 punteros directos con punteros a zona de 32 bits y se quiere disponer de archivos de hasta 256 MBytes. Indique de forma justificada el tamaño de zona y qué ventajas y desventajas tiene este diseño para archivos grandes de 256MBytes y archivos pequeños de 1 Byte.

Con punteros a zona de 32 bits podrían direccionarse particiones de hasta 4 GZonas. Al disponer de únicamente 8 punteros en el nodo-i sólo podremos direccionar 8 Zonas de datos

Si únicamente se dispone de 8 punteros directos en cada nodo-i, cada archivo sólo podrá tener asignados 8 zonas.

Si queremos archivos de 256MBytes, cada zona tendrá que ser de 32Mbytes ( $2^8 2^{20} / 2^3 = 2^5 2^{20}$ )

Archivos grandes: Acceso rápido ya que todos los punteros a zona están en el nodo-i y son directos

Archivos de 1 Byte: Hay mucha fragmentación interna ya que si se le asigna como mínimo una zona se está desperdiciando una gran cantidad de espacio de la partición  $\rightarrow 32\text{MBytes}-1\text{Byte}$