UNIVERSITAT POLITÈCNICA DE VALÈNCIA
Departamento de Informática de Sistemas y Computadoras (DISCA)

fSO

Ejercicio de Evaluación EEE2
16 de Diciembre de 2013

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

APELLIDOS		**NOMBRE**		Grupo
DNI		**Firma**		

- **No desgrape las hojas.**
- **Conteste exclusivamente en el espacio reservado para ello.**
- **Utilice letra legible. Responda de forma breve, precisa y justificando sus respuestas.**
- **El examen consta de 10 cuestiones, en cada una de ellas se indica su puntuación.**

1. Considere un programa, que contiene una función f que imprime la dirección de los objetos visibles (variables locales, globales y dinámicas o funciones) y el mapa de memoria del proceso. Durante su ejecución se visualiza un mapa de memoria que consta de 12 regiones como se muestra a continuación:

```
/**Código fuente **/
#include <…
float x;
void * p;

void f(int k){
 int L;
/*Mostramos Mapa*/
  ...
  }
main() {
  p=malloc(8);
  f(100);
}
```

```
Mapa  dememoria en /* Mostramos Mapa*/
a 08048000-08049000 r-xp 00000000 00:14 5374410   /home/m/programa
b 08049000-0804a000 r--p 00000000 00:14 5374410   /home/m/programa
c 0804a000-0804b000 rw-p 00001000 00:14 5374410   /home/m/programa
d 09cbe000-09cdf000 rw-p 00000000 00:00 0         [heap]
e b7524000-b76c8000 r-xp 00000000 08:06 291538    /lib/i386-linux-gnu/libc-2.15.so
f b76c8000-b76ca000 r--p 001a4000 08:06 291538    /lib/i386-linux-gnu/libc-2.15.so
g b76ca000-b76cb000 rw-p 001a6000 08:06 291538    /lib/i386-linux-gnu/libc-2.15.so
h b76e3000-b76e4000 r-xp 00000000 00:00 0         [vdso]
i b76e4000-b7704000 r-xp 00000000 08:06 291528    /lib/i386-linux-gnu/ld-2.15.so
j b7704000-b7705000 r--p 0001f000 08:06 291528    /lib/i386-linux-gnu/ld-2.15.so
k b7705000-b7706000 rw-p 00020000 08:06 291528    /lib/i386-linux-gnu/ld-2.15.so
l bfa0f000-bfa30000 rw-p 00000000 00:00 0         [stack]
```

Indique de forma justificada la letra de la región en la que estará ubicado cada uno de los objetos siguientes:

0,75 puntos

1	**Objeto**	**Región**	**Justificación**
	x	c	x es una variable global → se ubica en la región de datos soportada por el archivo con código ejecutable
	k	l	k es un parámetro → se ubica en la región de pila para la ejecución de la función f. La región de pila no está soportada
	f	a	f es una función del programa → se ubica en la región de código ejecutable soportada por el archivo
	L	l	x es una variable local → se ubica en la región de pila durante la ejecución de la función f. La región de pila no está soportada
	p	c	p es una variable global → se ubica en la región de datos soportada por el archivo con código ejecutable
	***p**	d	p es una variable de tipo puntero→ la función malloc reserva espacio en la región heap. p apunta a dicho espacio.

2. Sea un sistema con 512 MB de espacio de direccionamiento lógico, páginas de 4 KB y 64 MB de memoria física.

a) En una gestión de memoria mediante **paginación**, con un máximo de 4 páginas por proceso se han producido, durante la ejecución del proceso A, las siguientes traducciones de direcciones lógicas a físicas.

Dir. lógica	10185	28	14050	7248
Dir. Física	1845193	102428	5858	27728

Rellene el contenido de la tabla de páginas para dicho proceso A.

b) Indique cuál sería el formato de la dirección lógica para una gestión de memoria mediante **segmentación paginada** con un máximo de 32 segmentos por proceso, en este sistema.

c) En el modelo de **segmentación paginada** expuesto y teniendo en cuenta la tabla de páginas del Segmento 1 de un proceso dado, indique cuáles serán las direcciones lógicas correspondiente a las direcciones físicas 8220 y 25096.

	Tabla Páginas Segmento1	
	marco	validez
0	4	válido
1	5	válido
2	6	válido
3	2	válido

1,25 puntos (0,5+0,5+0,25)

2

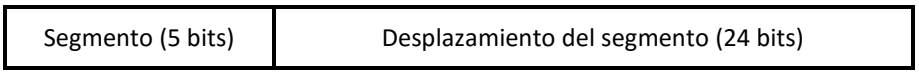
a)

Tabla de páginas del proceso A		
	marco	Bit validez
0	25	válido
1	6	válido
2	450	válido
3	1	válido

b) Formato dirección lógica segmentación paginada

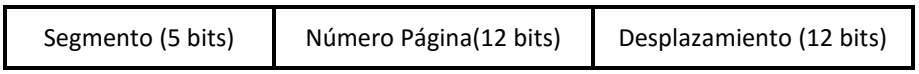
El espacio de direccionamiento lógico es de 512MBytes= 2^{29} bytes→ 29 bits de dirección lógica. Tamaño de página es de 4KBytes=2^{12}, el número de bits para desplazamiento es de 12 (tanto en marco como en página. Dado que hay 32 Segmentos por proceso→ 5 bits para el segmento.

29 bits



B28 B24 B23 B0

29 bits



B28 B24 B23 B0

c)

Dirección Física	Cálculo para obtener dir. lógica	Dirección lógica
8220	S1,3*4096 + 28	S1,12316
25096	S1, 2*4096 + 520	S1,8712

---

3. Respecto a los fallos de página en un sistema con memoria virtual, indique si son verdaderas (V) o falsas (F) cada una de las siguientes afirmaciones: (Nota: Un error penaliza una respuesta correcta)

0,75 puntos

3	**V/F**	
	V	Se produce fallo de página, cuando un proceso solicita una nueva página para expandir la pila, con independencia de que haya o no marcos libre para ubicarla.
	V	Se produce fallo de página cuando se solicita acceso a una página que todavía está en la reserva de marcos.
	V	Se produce fallo de página cuando se realiza el primer acceso a una página de código o de datos del proceso que no está cargada en memoria.
	F	Los fallos de página sólo se producen cuando no hay marcos libres en memoria para ubicar una nueva página.
	F	Los fallos de página siempre implican llevar una página de memoria a disco y traer una nueva página del disco a memoria.

4. Sea un sistema de memoria virtual con **dos niveles de paginación** y las siguientes características:

- Direcciones lógicas de 24 bits, direcciones físicas de 22 bits
- Tamaño de página de 4KBytes
- Las tablas de páginas de primer nivel disponen de 16 entradas
- El reemplazo de páginas con algoritmo ÓPTIMO y de ámbito LOCAL
- El sistema asigna un máximo de 4 marcos por proceso

Actualmente en memoria hay un proceso P que ocupa los marcos del 0 al 3, como se muestra en la figura, siendo p1 y p2 el índice de la tabla de primer y segundo nivel respectivamente.

Marco	p1, p2
0	(0x8,0x18)
1	(0x8,0xDA)
2	(0x3,0x85)
3	(0xB,0x15)

Nota: La notación 0x, hace referencia al sistema hexadecimal

a) Indique el formato de las direcciones lógicas y físicas, con número de bits y tamaños.

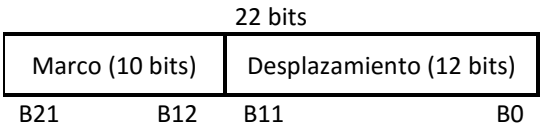
b) Suponga que a partir del instante mostrado en la tabla, la CPU hace referencia a la siguiente secuencia de direcciones de P: 0x154891 0x385F94 0x8DA122 0xB15679 0xB15A8C, 0x3851E9, 0x36C98A, 0x154917 0x1541CB, 0x385A03, 0x385545, 0x2F223C, 0x2F2B21, 0x1546F5. Muestre la evolución del contenido de memoria principal, e indique el número de fallos de página.

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

4 **a)** Estructura de direcciones lógicas y físicas

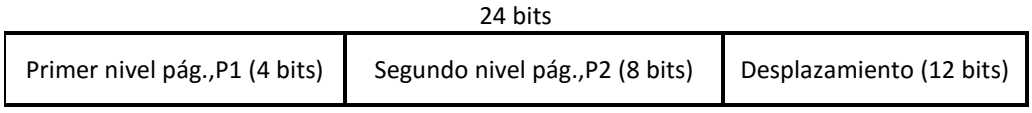
Tamaño marco = Tamaño de página= 4KBytes =2^{12}→ 12 de bits para desplazamiento de página. Dado que hay 22 bits de dirección física, quedan 10 bits para el número de marco y 12 para desplazamiento.

Dirección Física



Direcciones lógicas de 24 bits de los cuales: 4 bits son para índice de la tabla de primer nivel→p1, 8 bits para el índice de la tabla de segundo nivel→ p2 y 12 para el desplazamiento.

Dirección Lógica



b) Serie de referencias (154), (385), (8DA), (B15), (385), (36C), (154), (385), (2F2), (154)

Marco		**(154)**	(385)	(8DA)	(B15)	(385)	**(36C)**	(154)
0	(818)	**(154)**	(154)	(154)	(1,54)	(154)	(154)	(154)
1	(8DA)	(8DA)	(8DA)	(8DA)	(8DA)	(8DA)	**(36C)**	(36C)
2	(385)	(385)	(385)	(385)	(385)	(385)	(385)	(385)
3	(B15)	(B15)	(B15)	(B15)	(B15)	(B15)	(B15)	(B15)

Marco	(385)	**(2F2)**	(154)		
0	(154)	(154)	(154)		
1	(36C)	(36C)	(36C)		
2	(385)	(385)	(385)		
3	(B15)	**(2F2)**	(2F2)		

Dado que no se conocen mas referencias la página 2F2 puede ubicarse indistintamente en los marcos 1,2,3. La solución propuesta la ubica con criterio FIFO

Número total de Fallos de Página = 3 fallos con reeemplazo

5. Sea un sistema dotado de una memoria física de 16MBytes, direcciones lógicas de 20 bits, un tamaño de página de 4KBytes y paginación por demanda. Además utiliza un algoritmo de reemplazo de SEGUNDA OPORTUNIDAD con ÁMBITO LOCAL. Dicho sistema asigna 3 marcos a cada proceso. En el momento actual la asignación de marcos y el estado de la memoria de los procesos A y B que están siendo ejecutados son los mostrados en las tablas.

Marco	**Proceso**	**Nº Página**	**Bit Referencia**
0x003	A	0xA5	1
0x004	A	0x24	0
0x005	A	0x6E	1

Marco	**Proceso**	**Nº Página**	**Bit Referencia**
0x009	B	0x9A	0
0x00A	B	0x27	1
0x00B	B	0x3F	0

A partir de dicho instante se invocan las direcciones lógicas (A, 0x24350) (A, 0x9A000) (B, 0x3A120) (A, 0x99050) (B, 0x3A650) (B, 0x28495) (A, 0x6E350). Teniendo en cuenta que en el instante actual el orden de búsqueda de víctima, para cada uno de los procesos, coincide con el orden creciente del número de marco asignado a los procesos. Muestre la evolución de los marcos de memoria implicados, para ello rellene la tabla adjunta.

1,0 punto

5

Marco	Páginas Inicial Bit ref	Números de página referenciadas						
		A,0x24	A,0x9A	B,0x3A	A,0x99	B,0x3A	B,0x28	A,0x6E
0x003	0xA5 **→1**	0xA5 **→1**	**0x9A 1**	0x9A 1	0x9A 1	0x9A 1	0x9A 1	0x9A 1
0x004	0x24 0	0x24 1	0x24 **→0**	0x24 **→0**	**0x99** 1	0x99 1	0x99 1	0x99 1
0x005	0x6E 1	0x6E 1	0x6E 0	0x6E 0	0x6E **→0**	0x6E **→0**	0x6E **→0**	0x6E **→1**
0x009	0x9A **→** 0	0x9A **→0**	0x9A **→0**	**0x3A 1**	0x3A 1	0x3A 1	0x3A **→1**	0x3A **→1**
0x00A	0x27 1	0x27 1	0x27 1	0x27 **→1**	0x27 **→1**	0x27 **→1**	0x27 0	0x27 0
0x00B	0x3F 0	0x3F 0	0x3F 0	0x3F 0	0x3F 0	0x3F 0	**0x28 1**	0x28 1
	INI	ACIER	FALLO A	FALLO B	FALLO A	ACIER	FALLO B	ACIER

Total fallos de página = 4

6. En un sistema de tiempo compartido donde se han ejecutado los procesos A y B, se ha detectado que el orden de la serie de referencias a páginas que solicitó la CPU fue el siguiente:

Instante	0	1	2	3	4	5	6	7	8	9	10	11
Proc, pag	A1	B2	A3	B4	A2	B1	A5	B6	A2	B1	A2	B3

Instante	12	13	14	15	16	17	18	19
Proc, pag	A7	B6	A3	B2	A1	B2	A3	B6

Teniendo en cuenta que el tamaño de ventana de área activa es 4, indique el área activa para cada uno de los procesos en los instantes t=6, t=10, t=14 y t=19.

0,5 Puntos

6

Instante	Área Activa
t=6	A={1, 2, 3, 5} B={1, 2, 4}
t=10	A={2, 5} B={1, 4, 6}
t=14	A={2, 3, 7} B={1, 3, 6}
t=19	A={1, 3, 7} B={2, 6}

7. Dado el siguiente listado del contenido de un directorio en un sistema UNIX:

```
drwxr-xr-x     2 peter     users    4096 sep  8     2012  .
drwxr-xr-x     8 peter     users    4096 dec 10     14:39 ..
-rwsrw-r-x     1 peter     users    9706 sep  9     2012  append
-rw-rw-r--     1 peter     users    4310 sep  9     2012  f1
-r--rw-r--     1 peter     users    4157 sep  9     2012  f2
lrwxrwxrwx     1 peter     users       6 sep  9     2012  new->append
```

Donde el programa "append" añade el contenido del archivo especificado como primer argumento a otro archivo especificado en el segundo argumento. Considere la ejecución de la siguiente orden:

```
$ append f1 f2
```

Indique si son verdaderas (V) o falsas (F) cada una de las siguientes afirmaciones:
(Nota: Un error penaliza una respuesta correcta)

(0,75 puntos)

7	**V/F**	
	F	El usuario *john* que pertenece al grupo *students* no puede iniciar la ejecución del archivo "append"
	F	El usuario *mary* que pertenece al grupo *users* puede iniciar la ejecución del archivo "append" pero recibirá un error al intentar escribir en el archivo *f2* el proceso "append"
	F	Los dos usuarios mencionados anteriormente, *john* and *mary*, ejecutarán la orden correctamente y no recibirán mensajes de error
	F	El usuario *peter* que pertenece al grupo *users* ejecutará la orden correctamente
	V	Si se establecen los siguiente permisos "`-rwxrwsr-x`" en el archivo "append" con el bit SETGUID" entonces el usuario *john* podrá ejecutar correctamente la orden

8. Dada la siguiente secuencia de código en C. Indique el contenido de las tablas de descriptores de archivo en los puntos del código marcados como /* Punto 1*/, /* Punto 2*/ y /*Punto 3*/ para cada uno de los procesos activos en dicho punto y los valores de las variables fd1, fd2, fd3, tubo[0] y tubo[1] si procede.

```
/**Código fuente **/
……..
fd1=open("f1",…);
close(STDOUT_FILENO);
fd2=open("f2",...);
dup2(fd1,STDERR_FILENO); /*Punto 1*/
dup(STDERR_FILENO);
dup(STDIN_FILENO);
fd3=open("f3",...); /*Punto 2*/
if (fork()==0) {
   dup2(fd3, STDIN_FILENO);
   close(fd1);
}
close(fd2);
pipe(tubo); /*Punto 3*/
……
```

0,75 puntos

8

Punto = /*Punto 1*/
Valores variables

fd1=3, fd2=1

Tabla descriptores

0	stdin
1	f2
2	f1
3	f1
4	
5	
6	
7	

Punto =/*Punto 2*/
Valores variables

fd1=3, fd2=1, fd3=6

Tabla descriptores

0	stdin
1	f2
2	f1
3	f1
4	f1
5	stdin
6	f3
7	

Punto =/*Punto 3*/
Valores variables

fd1=3, fd2=1, fd3=6
tubo[0]=1, tubo[1]=7
Tabla descriptores

0	stdin
1	tubo[0] /*padre*/
2	f1
3	f1
4	f1
5	stdin
6	f3
7	tubo[1] /*padre*/

Punto =/*Punto 3*/
Valores variables

fd1=3, fd2=1, fd3=6
tubo[0]=1, tubo[1]=3
Tabla descriptores

0	f3
1	tubo[0] /*hijo*/
2	f1
3	tubo[1] /*hijo*/
4	f1
5	stdin
6	f3
7	

9. El siguiente programa ordena una lista de números mediante la invocación del programa sort, redirecciones y tubos. La columna de la derecha muestra el resultado de ejecución correcto.

/**Código fuente **/	/* Resultado de la ejecución **/

#include "los necesarios"
int main(int argc, char *argv[])
{ int backup, fd[2];
  char lista[]=" 4 \n 3 \n 1 \n 2 \n";
 	
 pipe(fd);
 printf("Antes:\n %s Despues:\n", lista);
 	
 if (fork()==0){
  dup2 (_____,_____);
  close (_____); close (_____);
  execlp("sort","sort",NULL);
 }else{
  backup=dup(STDOUT_FILENO);
  dup2 (_____, _____);
  close (_____); close (_____);
 	
  printf("%s", lista);
  dup2 (backup, STDOUT_FILENO);
  wait(NULL);
  printf("Y aquí termina todo\n");
  }
  return 0;
}
	Antes:
	 4
	 3
	 1
	 2
	Despues:
	 1
	 2
	 3
	 4
	Y aquí termina todo

a) Complete los parámetros de las llamadas *dup2* y *close* para que el programa funcione como se indica.

b) Justifique la funcionalidad de la variable *backup* que aparece en el código propuesto.

c) Explique qué sucedería si en el código propuesto se cambian las líneas de esta forma:

/*codigo propuesto **/	/** Nuevo código **/
……. dup2 (backup, STDOUT_FILENO); wait(NULL); printf("Y aquí termina todo\n"); ….	….. dup2 (backup, STDOUT_FILENO); printf("Y aquí termina todo\n"); wait(NULL); …

d) Explique qué sucedería si en el código propuesto se cambian las líneas de esta forma:

/*codigo propuesto **/	/** Nuevo código **/
… dup2 (backup, STDOUT_FILENO); wait(NULL); printf("Y aquí termina todo\n"); …	…. wait(NULL); dup2 (backup, STDOUT_FILENO); printf("Y aquí termina todo\n"); ….

1,25puntos (0,5+0,25+0,25+0,25)

9 **a)Complete**

```
 if (fork()==0){
    dup2 (fd[0], STDIN_FILENO);
    close (fd[0]);
    close (fd[1]);
    execlp("sort","sort",NULL);
}else{
    backup=dup(STDOUT_FILENO);
    dup2 (fd[1], STDOUT_FILENO);
    close (fd[0]);
    close (fd[1]);
```

b)
Como el programa cambia la salida estándar por el tubo, previamente se guarda el descriptor de salida estándar en backup para poder escribir luego de nuevo en él (deshacer la redirección)

c)
Al imprimir antes del wait, es posible que el orden no sea el mismo, es decir, podría aparecer en pantalla antes el mensaje de "Y aquí termina todo" que el resultado del sort. Dependerá del planificador.

d)
En este caso se bloquearía el sort esperando seguir leyendo del tubo. Ya el proceso padre q mantiene abierto el descriptor de escritura del tubo (que cierra el dup2(backup,STDOUT_FILENO)) y el proceso padre se bloqueado en el wait() esperando a que finalice el hijo.

---

10. Una partición de 6 GBytes se formatea con una versión de MINIX de las siguientes características:

- 1 bloque = 1 Kbyte, 1zona=1 bloque
- El tamaño de nodo-i es de 64 Bytes, con punteros de zona de 32 bits (7 punteros directos, 1 puntero indirecto y 1 puntero doble indirecto)
- Las entradas de directorio son de 32 Bytes
- Número de nodos-i = 8192
- La organización del sistema de archivos es la siguiente:

Bloque de arranque	Super bloque	Mapa de bits de nodos-i	Mapa de bits de zonas	Nodos-i	Zona de datos

a) Calcule de forma justificada el número de bloques que ocupan el mapa de bits de nodos-i, el mapa de bits de zonas, nodos-i y el número de zonas en el área de datos.

b) En este sistema de archivos el directorio raíz contiene:
   - 2 archivos regulares *reg1* y *reg2* de 10 Kbytes cada uno
   - 1 archivo directorio *dir* que a su vez contiene un archivo regular *reg3* de 10 Kbytes

b1) Indique de forma justificada cuántas zonas están siendo utilizadas por cada uno de los archivos y qué tipo de información contienen dichas zonas.

b2) Indique de forma justificada el número total de nodos-i ocupados en este sistema y el contenido del campo "Nº de enlaces" de cada uno de los nodos-i ocupados.

1,5 puntos (0,75+0,5+0.25)

10 a)

Nº bloques mapa nodos-i = 8192/(1024*8) = 1 bloque
Nº bloques mapa zonas = Nº de zonas /(1024*8) = (6*2^{30} /1024)/(1024*8) =
= 6*2^{20}/2^{13} = 6*2^{7} = 768 bloques
Nº bloques nodos-i = 8192 * 64 Bytes /(1024 Bytes) = 512 Bloques
Cabecera de la partición = 1 + 1 + 1 + 768 +512 = 1283 Bloques
Nº zonas area de datos = Nº bloques total – Nº bloques cabecera =
= 6*2^{30} / 1024 – 835 = 6*2^{20} – 1283 = 6290218

b1)

Archivo	Nº de zonas ocupadas	Tipo de información de zonas
/	1	5 entradas de directorio que son reg1, reg2, dir , . y .. = 5x32Bytes = 160 Bytes
/reg1	11	10 zonas con datos + 1 zona con punteros a zona que contiene 3 punteros (3 punteros * 4 Bytes=12 Bytes)
/reg2	11	10 zonas con datos + 1 zona con punteros a zona que contiene 3 punteros (3 punteros * 4 Bytes=12 Bytes)
/dir	1	3 entradas de directorio reg3, . y .. = 3x32Bytes= 96 Bytes
/dir/reg3	11	10 zonas con datos + 1 zona con punteros a zona que contiene 3 punteros (3 punteros * 4 Bytes=12 Bytes)

b2)

Hay un total de 5 nodos-i ocupados, una por cada archivo del sistema: /, reg1, reg2, reg3, dir. El número de enlaces en cada uno de los nodos-i ocupados es:
Del nodo-i del / -> 3 enlaces
Del nodo-i del /reg1 -> 1 enlace
Del nodo-i del /reg2 -> 1 enlace
Del nodo-i del /dir -> 2 enlaces
Del nodo-i del /dir/reg3 -> 1 enlace