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

fSO

etsinf

Ejercicio de Evaluación EEE1+EEE2
21 de Enero de 2014

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 11 cuestiones, en cada una de ellas se indica su puntuación.**

1. Considera los dos programas E y F, obtenidos al compilar los códigos fuente siguientes:

```
/**Código Programa E.c **/
#include <…
int pid;
main(){
  pid=fork();
  if (pid==0)
     {sleep(3);}
  execl("./F","F",NULL);
}
```

```
/**Código Programa F.c **/
#include <…
int pid;
main(){
  pid=fork();
  if (pid==0)
     { sleep(1);}
  execl("/bin/date","date","+%S",NULL);
}
```

Nota: La orden "date +%S" imprime en pantalla los segundos de la hora actual. Por ejemplo a las 20:30:12, esta orden imprime "12".

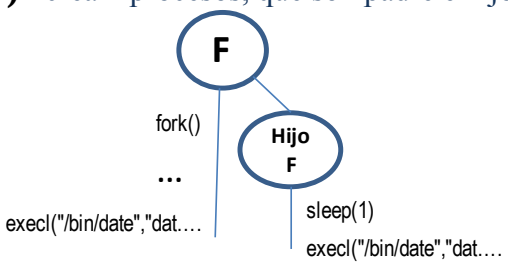
Suponga que los ejecutables de E y F se encuentran en el directorio de trabajo y que se ejecutan sin incidencias. Indique razonadamente:

a. Cuántos procesos se crean al ejecutar la orden "`./F`" y qué parentesco existe entre ellos.
b. Cuál será la salida por pantalla, si se ejecuta la orden "`./F`" a las 09:10:25.
c. Cuántos procesos se crean al ejecutar la orden "`./E`" " y qué parentesco existe entre ellos.
d. Cuál será la salida por pantalla, si se ejecuta la orden "`./E`" a las 09:10:25.

1,0 puntos

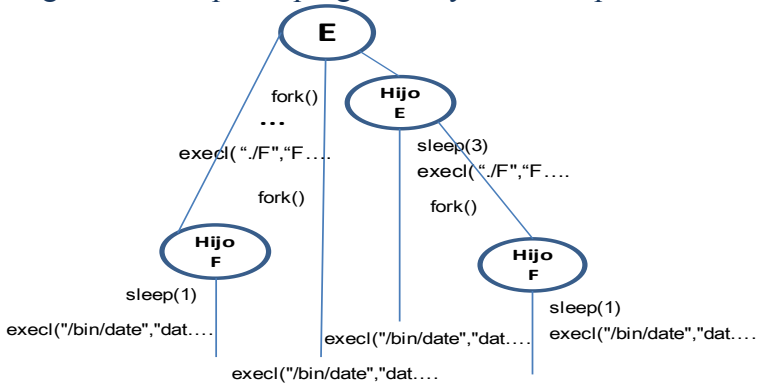
1

a) F crea 2 procesos, que son padre e hijo



b) El padre escribe "25", el hijo se espera 1 segundo y escribe "26".

c) 4 procesos en total: E crea 2 procesos, padre e hijo; la función exec() sustituye el código de ambos por el programa F y cada uno producen un hijo



d) E ejecuta F y se muestra "25" y "26"; E-hijo se espera 3 segundos muestra 28 y 29.

2. Un sistema informático está dotado de un planificador a corto plazo con tres colas que denominaremos QSRTF, QRRB y QRRA. La política de planificación entre colas es por prioridades expulsivas, siendo la cola QRRA la más prioritaria y la QSRTF la de menor prioridad. La política de planificación de cada cola es:

- la cola QSRTF utiliza el algoritmo **SRTF**. Esta cola es la menos prioritaria
- la cola QRRB utiliza el algoritmo **RR** (turno rotatorio) con un quantum **q = 1 ut**
- la cola QRRA utiliza **RR** con un quantum **q = 2 ut**. Esta cola es la más prioritaria

Los procesos nuevos y los procedentes de E/S siempre se ubican en la cola QRRA, es decir, la más prioritaria. Existe una política de degradación entre colas de manera que cuando un proceso consume un quantum q es degrado a la cola inmediatamente inferior en prioridad. Un proceso que llega a la cola QSRTF permanece en ella hasta que invoca una operación de E/S o finaliza. La E/S es sobre el mismo dispositivo con política FCFS.
Suponga que llegan al sistema tres procesos A, B y C, con el siguiente perfil de ejecución:

Proceso	**Perfil de Ejecución**	**Instante de Llegada**
A	3 CPU + 2 I/O + 6 CPU	0 (1º)
B	6 CPU + 1 I/O + 2 CPU	0 (2º)
C	4 CPU + 3 I/O + 1 CPU	0 (3º)

a) Rellene la tabla indicando donde se encuentran los procesos en cada instante de la ejecución.
b) Calcule el tiempo medio de espera, tiempo medio de retorno y la utilización de CPU

Nota: En caso de llegadas simultaneas a colas, considere los que vienen de E/S como más prioritarios.

1,5 (1,0+0,5) Puntos

2a

T	**Q_{SRTF}**	**Q_{RRB} q=1**	**Q_{RRA}q=2**	**CPU**	**Cola E/S**	**E/S**	**Comentarios**
0			C, B, A	A			A(1st), B(2nd), C(3rd) arrive
1			C, B	A			
2		A	C, B	B			
3		A	C	B			
4		B, A	C	C			
5		B, A		C			
6		C, B, A		A			
7		C, B		B		A	
8	B	C		C		A	
9	C, B		A	A			
10	C, B			A			
11	C, B	A		A			
12	A, C, B			C			
13	A, B			B		C	A y B igual ráfaga (3) B antes en cola
14	A			B		C	
15	A			B		C	
16	A		C	C		B	
17	A		B	B			Fin C
18	A			B			
19	A			A			Fin B
20				A			
21				A			
22							**Fin A**

2b

Tiempo medio de espera = (11+ 10 + 9)/3=10
Tiempo medio de retorno = (22-0) + (19-0) + (17-0)/3= 19,3
CPU utilization = 22/22= 100%

3. Teniendo en cuenta las siguientes posibles transiciones de estado del proceso Shell:

a) Shell pasa de preparado a ejecución
b) Shell pasa de ejecución a preparado
c) Shell pasa de ejecución a suspendido
d) Shell pasa de suspendido a preparado

Indique qué transición de estado experimentará el Shell, durante una sesión de trabajo, para cada una de las siguientes incidencias propuestas:

0,75 puntos

3	**Incidencias en una sesión de trabajo**	**Transición del estado del Shell**
	Se ha tecleado una línea de órdenes y se ha pulsado la tecla de retorno de carro	d
	Mientras el Shell está procesando la línea de órdenes, interviene el planificador del sistema y le asigna la CPU a un proceso del sistema	b
	El Shell invoca una operación de apertura de un archivo	c
	Mientras los procesos hijos del Shell, creados con la llamada fork(), están en la cola de preparados del planificador, el Shell ejecuta la llamada wait()	c
	El Shell ha mostrado el promt en el terminal y está esperando que el usuario teclee una orden	c

4. Dado el siguiente código en C donde se crean dos hilos:

```
#include <……
#define VECES 200000
long int W = 100;
int llave=0;
int main (void) {
  pthread_t th1,th2;
  pthread_attr_t atrib;

  pthread_attr_init( &atrib );
  printf("Erase dos hilos agrega\n");
  pthread_create(&th1, &atrib, agrega, null);
  pthread_create(&th2, &atrib, agrega, null);
  pthread_join(th1);
  pthread_join(th2);
  printf ("Fin del hilo principal\n");
  exit(0);
}
```

```
void *agrega (void *argum)
{long int i;
 long int aux;

 /**Posicion A**/
 for (i = 0; i<VECES; i++)
 {
   /**Posicion B**/
    W= W +2;
   /**Posicion C **/
 }
/**Posicion D***/
 printf("W = %ld\n", W);
 printf("Soy hilo agrega \n");
 pthread_exit(0);
}
```

a) Identifique en que instrucciones puede haber condiciones de carrera y justifique su respuesta exponiendo un escenario donde ocurra condición de carrera.

b) En este caso se quiere solucionar el problema de la sección critica utilizando la instrucción `test_and_set(&llave)`. Indique de entre /**Posición A** / y /**Posición B**/ cuál es el lugar más adecuado para invocar al protocolo de entrada y entre /**Posición C** /y /**Posición D**/cual es el adecuado para invocar al protocolo de salida. Justifique su selección.

0,75 Puntos (0,4 + 0,35)

4	**a)**
	W=W+2; Supongamos W=2. El hilo th1 lee el valor de W que es 2, a continuación, la CPU se le concede a th2 que lee W=2 y le suma 2 actualizando la variable W=4. A continuación se ejecuta th1 y actualiza W=4. Se han llevado a cabo dos sumas y sólo se ha reflejado una. Esto supone una incosistencia en el valor de W.--> Condición de carrera

b)

La solución correcta es /**Posicion B**/ y /**Posicion C**/ para protocolo de entrada y salida respectivamente.

También sería una posible solución al problema de la sección crítica si se ubican los protocolos fuera del bucle, en /**Posicion A**/y /**Posicion D**/. Pero con esta ubicación de los protocolos realmente el acceso a la variable W se realizaría de forma secuencial por parte de los diferentes hilos. Por lo que cuando un hilo accede a la sección crítica, ejecuta todas las instrucciones del bucle antes de abandonar la sección crítica y por tanto el efecto es como si los hilos accedieran a la misma de forma secuencial. No es adecuada cuando lo que se requiere es concurrencia

---

5. En el siguiente pseudocódigo, H1, H2 y H3, representa hilos de un mismo proceso que se ejecutan concurrentemente y que comparten memoria. Utilice operaciones sobre semáforos (P(S) y V(S)), para garantizar que las funciones cuyo nombre empieza por "secuenciaX()" se ejecutan, según el orden que sugieren los números de sus nombres. Si hay varias funciones con el mismo número, ninguna de ellas debe empezar a ejecutarse antes de que termine la función con el número anterior y todas ellas deben haber terminado antes de que empiece la función con el número siguiente. Además debe garantizar, que las funciones "sc()" se ejecutan en exclusión mutua entre ellas. Indique qué valor inicial de los semáforos que utilice.

Nota: Puede utilizar la notación propuesta por Dijkstra, P(S) y V(S)

0,75 Puntos

5

```
/** Inicialice aquí los semáforos que utiliza**
mutex=1; s1=0;   s3=0;   s5=0;   s2=0;   s4=0;
```

H1	H2	H3

P(mutex);
sc();
V(mutex);
 	
P(s1);
secuencia2();
V(s2);
 	
P(s4)
secuencia4();
V(s5)
	secuencia1();
	V(s1);
	V(s1);
	
	P(mutex);
	sc();
	V(mutex);
	
	P(s2);
	P(s3);
	secuencia 3()
	V(4)
		P(s1);
		secuencia2();
		V(s3);
		
		P(s5)
		secuencia5();
		
		P(mutex);
		sc();
		V(mutex);

6. Un sistema con gestión de memoria mediante paginación, dispone de 8 GB de espacio de direccionamiento lógico y 1 GB de memoria física. En un momento dado, la tabla de páginas del sistema presenta el siguiente contenido:

Página	Nº de marco	Bit de validez
0	67	valido
3	10	valido
…	…	…
23	4	valido
…	…	…
42	22	valido
…	…	…
600	1	valido
…	…	…

Complete las celdas en blanco de la siguiente tabla. Cada fila representa un supuesto distinto de distribución de campos de las direcciones físicas y lógicas. Ignore las celdas en gris. Justifique su resultado para cada caso indicando las operaciones oportunas que ha realizado.

1.0 Puntos

6

Rellene las celdas en blanco con los valores correspondientes

	Direc. física	Direc. lógica	Tamaño de página	Nº de marcos	Nº de páginas
Caso 1			262144=2^{18}	4096=2^{12}	
Caso 2				131072= 2^{17}	1048576=2^{20}
Caso 3	20830	6494	2048=2^{11}		
Caso 4	314816	1560000		16384 =2^{14}	

Justifique para cada caso los valores de la tabla anterior

Caso 1	Tamaño de página en Bytes = $2^{\text{bits de desplazamiento de página}}$ Dirección física =30 bits→ 1 GBytes = 2^{30}; Número total de marcos = 4096= 2^{12} → 12 bits para el número de marco Bits desplazamiento página = Bits de desplazamiento de marco= 30 bits – bits para nº de marco = 30 – 12 = 18 bits; Tamaño de página = 2^{18} = 262144 (256 KB)
Caso 2	Número total de marcos = $2^{\text{bits nº de marco}}$; Dirección Lógica=33 bits→ 8 GBytes= 2^{33}; Total de paginas =1048576 =2^{20} → 20 bits para el número de página Bits desplazamiento página = Bits de desplazamiento de marco = 33 bits – 20 bits para nº de página = 33 – 20 = 13 bits Dirección física =30 bits→ 1 GBytes = 2^{30} bits para el nº de marco = 30 – bits desplazamiento = 30 – 1 = 17bits Tamaño de marco = 2^{17} = 131072 =128 KB
Caso 3	Dirección lógica = (Nº de Página lógica*Tamaño de página)+desplazamiento Número de Marco = Direc. Física/ tamaño de marco = 20830 div 2048 = 10 Desplazamiento = 30830 mod 2048 = 350; Según la tabla de páginas, en el marco 10 está ubicada la Página lógica 3 Dirección lógica = (3*2048)+350=6494
Caso 4	Dirección física = (Nº de Marco*Tamaño de marco)+desplazamiento Dirección física =30 bits→ 1 GBytes = 2^{30}; Número total de marcos = 16384= 2^{14} → 14 bits para el número de marco Bits desplazamiento página = Bits de desplazamiento de marco= 30 bits – bits para nº de marco = 30 – 14 = 16 bits; Tamaño de marco = 2^{16} = 65536 (64 KB) Número de Página lógica = 1560000 div 65536 = 23; Desplazamiento dentro de la página lógica = 1560000 mod 65536 = 52672 Según la tabla de páginas la página 23 está ubicada en el Marco 4. Dirección física = (4*65536)+52672=314816

7. Sea un sistema con **paginación por demanda** con páginas 4Kbytes, y cuyo tamaño lógico máximo por proceso es de 256 páginas. En un momento dado dicho sistema dispone únicamente de 6 marcos (0x12, 0x13, 0x14, 0x2A, 0x2B y 0x2C) para ejecutar los procesos de usuario Y y Z. Suponga que la información relativa a los procesos Y y Z en el instante t=50 es la mostrada en la tabla siguiente.

PROCESO:PÁGINA	Marco (hexadecimal)	Instante de carga	Instante de última referencia	Bit Validez
Y:0x0	0x12	10	50	1
Y:0x1	0x13	15	35	1
Y:0x2				0
Y:0x3	0x14	20	20	1
Z:0x40	0x2A	12	39	1
Z:0x41	0x2B	17	17	1
Z:0x42				0
Z:0x43				0

A partir del instante t=50 se referencia la siguiente secuencia de páginas Y:0x2, Y:0x1, Z:0x42, Z:0x43, Y:0x3, Z:0x40, Y:0x0, Y:0x3 como se indica en las tablas que debe completar.

a) A partir del instante t=50, indique para cada instante t, la evolución del contenido de los marcos asignados a Y y Z si se aplica un algoritmo de **reemplazo FIFO de ámbito GLOBAL.**

b) A partir del instante t=50, suponga que el sistema aplica una política de reparto de marcos equitativa para los proceso Y y Z e indique para cada instante t, la evolución del contenido de los marcos si se aplica un **algoritmo de reemplazo LRU de ámbito LOCAL.**

c) Indique de forma justificada si en un sistema con paginación por demanda puede aparecer fragmentación externa o interna y que cantidad de memoria podría inutilizarse por esa causa.

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

7

a) Algoritmo de reemplazo FIFO de ámbito GLOBAL.

	t=50	t=51	t=52	t=53	t=54	t=55	t=57	t=58	t=59
Marco	**----**	**Y:0x2**	**Y:0x1**	**Z:0x42**	**Z:0x43**	**Y:0x3**	**Z:0x40**	**Y:0x0**	**Y:0x3**
0x12	Y:0	Y:0	Y:0	Z:42(F)	Z:42	Z:42	Z:42	Z:42	Z:42
0x13	Y:1	Y:1	Y:1 (A)	Y:1	Y:1	Y:1	Z:40(F)	Z:40	Z:40
0x14	Y:0x3	Y:3	Y:3	Y:3	Y:3	Y:3(A)	Y:3	Y:3	Y:3(A)
0x2A	Z:40	Z:40	Z:40	Z:40	Z:43(F)	Z:43	Z:43	Z:43	Z:43
0x2B	Z:41	Z:41	Z:41	Z:41	Z:41	Z:41	Z:41	Y:0(F)	Y:0
0x2C		Y:2 (F)	Y:2	Y:2	Y:2	Y:2	Y:2	Y:2	Y:2

Numero de fallos de Página =5

b) Algoritmo de reemplazo LRU de ámbito LOCAL

	t=50	t=51	t=52	t=53	t=54	t=55	t=57	t=58	t=59
Marco	**------**	**Y:0x2**	**Y:0x1**	**Z:0x42**	**Z:0x43**	**Y:0x3**	**Z:0x40**	**Y:0x0**	**Y:0x3**
0x12	Y:0	Y:0	Y:0	Y:0	Y:0	Y:3 (F)	Y:3	Y:3	Y:3 (A)
0x13	Y:1	Y:1	Y:1 (A)	Y:1	Y:1	Y:1	Y:1	Y:1	Y:1
0x14	Y:0x3	Y:2 (F)	Y:2	Y:2	Y:2	Y:2	Y:2	Y:0 (F)	Y:0
0x2A	Z:40	Z:40	Z:40	Z:40	Z:40	Z:40	Z:40 (A)	Z:40	Z:40
0x2B	Z:41	Z:41	Z:41	Z:41	Z:43 (F)	Z:43	Z:43	Z:43	Z:43
0x2C		-	-	Z:42 (F)	Z:42	Z:42	Z:42	Z:42	Z:42

Numero de fallos de Página = 5

c)Fragmentación externa o interna y cantidad de memoria podría inutilizarse

En paginación puede ocurrir fragmentación interna, debido a que el tamaño de un proceso no tiene porque ajustarse a un número de páginas exactas. El desperdicio de memoria suele ocurrir en la última página y se estima una media de media página por proceso. En este caso 2 KBytes.

8. Utilizando código en lenguaje C y llamadas al sistema UNIX, complete el siguiente código propuesto para que realice la ejecución de la siguiente línea de comandos:

```
$ ls -la | wc -l > resultado.txt
```

0,75 Puntos

8

```
int main (int argc, char *argv[]) {

char* argumentos1[] = { "ls" , "-la" , 0};
char* argumentos2[] = { "wc" , "-l" , 0};
mode_t fd_mode=S_IRWXU;
int fildes[2], fd, ;

pipe(fildes);
if (fork()==0) {
  dup2 (fildes[1], STDOUT_FILENO),
   close (fildes[0]);
   close (fildes[1]);
   execvp("ls",argumentos1)
   fprintf (stderr, "ejecución exec fallida\n"); exit(-1);
 }
else {
   fd=open ("resultado.txt" , O_RDWR|O_CREAT, fd_mode);
   dup2 (fd, STDOUT_FILENO);
   dup2 (fildes[0],STDIN_FILENO),
   close (fildes[0]);
   close (fildes[1]);
   close (fd);
   execvp("wc",argumentos2)
   fprintf (stderr, "ejecución exec fallida\n");exit(-1);
   }
}
```

9. 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. Asuma que en /*Punto Inicio*/ la tabla de descriptores de archivos para el proceso P es la mostrada.

```
/** Sección de Código de P**/
1    …… /*Punto Inicio*/
2    fd1=open("f1",…);
3    close(STDOUT_FILENO);
4    fd2=open("f2",...);
5   dup2(fd1,STDERR_FILENO);
6    /*Punto 1*/
7    dup(STDERR_FILENO);
8    dup(STDIN_FILENO);
9    fd3=open("f3",...);
10   /*Punto 2*/
11
12    if (fork()==0) {
13     dup2(fd3, STDIN_FILENO);
14      }
15    close(fd1);
16    close(fd2);
17    pipe(tubo);
18
19     /*Punto 3*/
20 ……
```

0,75 puntos

9

/*Punto Inicio*/
Valores variables=
No procede valores

Tabla descriptores	
0	STDIN_FILENO
1	STDOUT_FILENO
2	STDERR_FILENO
3	
4	
5	
6	
7	

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 3*/
Valores variables=
fd1=3, fd2=1,fd3=6
tubo[0]=1,
tubo[1]=7
/*padre*/

Tabla descriptores	
0	stdin
1	tubo[0]
2	f1
3	tubo[1]
4	f1
5	stdin
6	f3
7	

Punto=/*Punto 3*/
Valores variables
fd1=3, fd2=1,fd3=6,
tubo[0]=1,tubo[1]=7
/*hijo*/

Tabla descriptores	
0	f3
1	tubo[0]
2	f1
3	tubo[1]
4	f1
5	stdin
6	f3
7	

10. Cada fila de la siguiente tabla, representa un caso que corresponde a un conjunto de parámetros utilizados en el formateo de una partición MINIX de 48MB con un nodo-i por cada 4 KB. Ignore las celdas en gris y complete los valores que corresponde de las celdas en blanco. Justifique su resultado indicando las operaciones que ha realizado para obtenerlos.

1,0 Puntos

10 Complete los valores que corresponde de las celdas en blanco

	Tamaño Bloque	Tamaño de zona	Tamaño de nodo-i	Bloques para mapa de nodos-i	Bloques para mapa de zonas	Bloques para nodos-i
Caso 1	1024 Bytes	1 Bloque	32 Bytes	2 Bloques		384 Bloques
Caso 2	2048 Bytes	1 Bloque	64 Bytes		2 Bloques	384Bloques
Caso 3	2048 Bytes	2Bloque	32 Bytes		1 Bloque	

Justifique su resultado indicando las operaciones que ha realizado para obtenerlos.

Caso 1	Número de nodos-i = 48MB/4KB=12K nodos-i = 12288 nodos-i Bloques para nodos-i = (total nodos-i* Tamaño en Bytes de nodo-i)/Bytes por bloque = (12KBytes*32Bytes)/1024Bytes=384Bloques Bloques para mapa nodos-i= (12 K nodos-i / 1K *8)= 1,5 bloques =2 Bloques
Caso 2	Nº de bloques del mapa de zonas = zonas/(Bytes por bloque*8) Numero de zonas = Tamaño partición/Tamaño zona=48MB/2048=24K zonas Mapa de zonas necesita un bit por cada zona de la partición Nº de bloques del mapa de zonas = 24K zonas /2048 *8 bits= 24K /2K *8=1.5 bloques = =2 Bloques Número de nodos-i = 48MB/4KB=12K nodos-i = 12288 nodos-i Bloques para nodos-i = (total nodos-i* Tamaño en Bytes de nodo-i)/Bytes por bloque = (12KBytes*64Bytes)/2048Bytes= 384Bloques
Caso 3	Numero de zonas = Tamaño partición/Tamaño zona=48MB/2048 *2=12K zonas Mapa de zonas necesita un bit por cada zona de la partición Nº de bloques del mapa de zonas = 12K zonas /2048 *8 bits= 12K /2K *8=0,75 bloques = 1 Bloque

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

```
drwxr-xr-x   2 user1     grpa            4096 ene  8     2013  .
drwxr-xr-x  11 user1     grpa            4096 ene 10     14:39 ..
-rwsr--r-x   1 user1     grpa         1139706 ene  9     2013  borrar
-rw-------   1 user1     grpa          634310 ene  9     2013  fich
lrwxrwxrwx   1 user1     grpa               3 ene  9     2013  dat→fich
```

Donde el archivo `borrar` es un programa que elimina un archivo pasado como argumento.
a)Justifique si el usuario `user2` del grupo `gprb` puede eliminar o no el archivo `fich` ejecutando la orden: `$ borrar fich`
b) Justifique si el usuario `user2` del grupo `gprb` puede crear un archivo de tipo enlace en dicho directorio ejecutando la orden: `$ ln -s borrar newborrar`

0.5 Puntos

11	a) Si puede eliminar el archivo fich. Los permiso para el usuario `user2,gprb` al ejecutar `$ borrar fich`, son los de la 3ª tripleta (`r-x`) del archivo borrar. Por tanto user2, podrá ejecutar el programa borrar y durante su tiempo de ejecución actúa como user1, gprb gracias al bit SETUID y podrá borrar el fichero fich ya que user1 tiene permiso de escritura sobre el archivo y en el directorio actual.
	b) No puede crear un archivo de tipo enlace ya que user2 no tiene permisos de escritura en el directorio actual. Para crear un archivo nuevo, el usuario debe tener permisos de escritura en el directorio actual (.). Los permiso del usuario `user2,gprb` en el directorio actual (.) son (r-x), la 3ª tripleta.