Apellidos:_____Nombre:_____ DNI: _____

Sistemas Operativos – Grado Ingeniera Informática UDC. Enero 2018
Sólo puede usar lápiz, bolígrafo y calculadora. Smartphones apagados y en la mochila.
Tiempo máximo para todo el examen: 2h 45m

Parte Sistema Ficheros
(Se deben contestar correctamente todas las cuestiones de cada pregunta para puntuar la misma).
Puntuación preguntas: P1: 0.75, P2: 0.5, P3: 0.25, P4: 0.5

P1) Un sistema de archivos tipo system V tiene un tamaño de bloque de 2 Kbytes e inodos con 10 direcciones directas de bloques, una indirecta simple, una indirecta doble y una indirecta triple. Además, utiliza direcciones de bloques de 8 bytes. Consideremos el fichero "datos" en el directorio /home/usr1/dir1, con un tamaño de 257 MBytes.

Calcular cuántos bloques de disco son necesarios para representar ese archivo de 257 MBytes.

Nº bloques de datos: 128.5 Kbloques (131584)

Nº de bloques de índices: 518

P2) Supongamos la siguiente secuencia de comandos desde el directorio *home/usr1/dir1*:

i) Se crea un soft link al archivo *datos* (cuyo número de inodo es 45689, tamaño 257 MBytes y num. de hard links=1, con el comando *ln -s datos slink* /*el comando *ls -l mostraría slink -> datos* */
ii) Posteriormente se crea un hard link al fichero slink:
ln slink hslink /* crea hard link *hslink* */
iii) Una vez creados los ficheros *slink* y *hslink*, se borra el fichero *datos* (*rm datos*).

Contestar a lo siguiente:

A. El inodo de *slink* es el 45689 (Cierto/Falso): Falso

B. El tamaño de *hslink* es 14 bytes (Cierto/Falso): Falso

C. Tras el borrado del fichero datos (*rm datos*) se libera su inodo (queda como libre en la lista de inodos en disco) (Cierto/Falso): Cierto

D. Con un sistema de ficheros Unix basado en registro (*journaling file system*), al borrar un fichero, parte de la información del inodo se guarda en el registro. (Cierto/Falso): Falso

P3) Un proceso abre el archivo anterior "/home/usr1/dir1/datos" en modo lectura, cuyo número de hard links es 1. El usuario efectivo del proceso coincide con el propietario del fichero, y tiene además los permisos de acceso al directorio *home*, *usr1 y dir1*. Las cachés de datos e inodos están inicialmente vacías y la entrada *home* está en el tercer bloque del directorio raíz, mientras los demás entradas están en el primer bloque de su directorio padre. Indicar lo siguiente referente al trozo de código:

```
struct stat buf;
char c;
chmod("/home/usr1/dir1/datos", 0640);
if (lstat ("/home/usr1/dir1/datos", &buf)!=-1){
    printf("%s", convertir_permisos(buf.st_mode));     /* se convierten los permisos a formato
                                                          rwxrwxrwx y se imprimen*/
    int fd=open("/home/usr1/dir1/datos", O_RDONLY); /* es la primera apertura de fichero del
                                                          proceso */
    lseek(fd, 67108864, SEEK_SET);     /* SEEK_SET indica que el desplazamiento
                                          se considera a partir del origen del fichero
                                          (67108864 = 2^26)  */
   c=fgetc(fd);
   close(fd);
    }
```

A. ¿Cuál es el valor asignado al descriptor de fichero *fd*?: 3

B. ¿Cuál es el número de accesos necesarios a disco, únicamente en el área de datos, en la apertura del fichero?: 6

C. Indica el número de bloques que el S.O. necesita leer en disco para obtener el valor de *c* en *fgetc(fd*): 3

D. *fgetc* es una función de librería de C. Cierto/Falso: Cierto

E. Indica los permisos del fichero impresos en formato *rwxrwxrwx*: rw- r-- ---

P4) Calcular qué número de bloque lógico corresponde al inodo del raíz (se comienza a contar en el bloque lógico 0), y cuál bloque lógico al inodo del fichero *datos*, suponiendo lo siguiente:

i) El nº de inodo del "/" es el 2, y al fichero datos le corresponde el inodo nº 202 (los inodos se comienzan a numerar a partir de 1).
ii) El tamaño de un inodo es de 64 bytes (tamaño bloque = 2Kbytes).
iii) El boot ocupa 2 bloques y el superbloque 9 bloques.

Nº bloque lógico inodo "/": 11

Nº bloque lógico inodo "datos": 17

Apellidos:_____Nombre:_____ DNI: _____

Sistemas Operativos – Grado Ingeniera Informática UDC. Enero 2018
Sólo puede usar lápiz, bolígrafo y calculadora.
Tiempo máximo para todo el examen: **2h45m**

Parte Memoria.

1. (0.5 puntos) Considere el siguiente código C.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char** f();
char** g;

int main(void)
{
   char a[]="Hello function f";
   char **b=NULL;
   b = f(a);
   g = f(a);
   printf("Cadena[%s] Dirección[%p] \n", a, &a);
   printf("Cadena[%s] Dirección[%p] \n", *b, b);
   printf("Cadena[%s] Dirección[%p] \n", *g, g);


   return 0;
}

char** f(char *a)
{
   static char *c;
   c = (char *) malloc(strlen(a)+1);
   strcpy(c, a);
   return &c;
}
```

Una vez compilado y enlazado, la ejecución de este código imprime 3 veces la cadena "Hello function f" por pantalla y tres direcciones en hexadecimal (directiva %p). Considere las zonas de direcciones de la memoria de un proceso: P Pila, H Heap, y la zona de variables globales y estáticas en la que se distinguen dos zonas según están inicializadas o no, es decir, GSI Variables globales y estáticas inicializadas, GSN Variables globales y estáticas no inicializadas (llamada históricamente BSS). Las tres direcciones que se imprimen por pantalla se corresponden con las siguientes zonas:

a)La primera de P, la segunda y la tercera de H.
b)La primera de GSI, la segunda y la tercera de H.
c)La primera de P, la segunda y la tercera de GSN.
d) La primera de GSI, la segunda y la tercera de GSN.
e)Las tres de P.
f)Ninguna de las anteriores es cierta.

Indique la respuesta correcta y **razónelo. Tanto la respuesta como el razonamiento deben ser correctos para puntuar la pregunta.**
Respuesta correcta: **_____c____**

Cadena[Hello function f] Dirección[0x7ffd7d350170]
Cadena[Hello function f] Dirección[0x600b38]
Cadena[Hello function f] Dirección[0x600b38]

2. (0.5 puntos) Considere la siguiente cadena de referencia a páginas de un proceso:
1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 4
¿Cuántos fallos de página se producen si el proceso tiene asignadas 6 frames (páginas físicas), contando todos los fallos incluidos los que se producen al principio de la cadena que no implican reemplazo, y considerando que las frames inicialmente están vacías?. Rellena también la tabla con la asignación de páginas a frames. T**anto el número de fallos como la asignación de páginas a frames deben ser correctas para puntuar el apartado.**

(0.2 puntos) LRU, número de fallos:___**8**

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	4
1	1	1	1	*	*	1	1	*	*	*	*	1	*	*	*	*	*	*	1
	2	2	2	*	*	2	2	*	*	*	*	2	*	*	*	*	*	*	2
		3	3	*	*	3	3	*	*	*	*	3	*	*	*	*	*	*	3
			4	*	*	4	4	*	*	*	*	7	*	*	*	*	*	*	7
						5	5	*	*	*	*	5	*	*	*	*	*	*	4
							6	*	*	*	*	6	*	*	*	*	*	*	6
F	F	F	F			F	F					F							F

(0.2 puntos) FIFO, número de fallos:___**11**

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	4
1	1	1	1	*	*	1	1	*	*	*	*	7	*	*	*	7	7	7	7
	2	2	2	*	*	2	2	*	*	*	*	2	*	*	*	1	1	1	1
		3	3	*	*	3	3	*	*	*	*	3	*	*	*	3	2	2	2
			4	*	*	4	4	*	*	*	*	4	*	*	*	4	4	3	3
						5	5	*	*	*	*	5	*	*	*	5	5	5	4
							6	*	*	*	*	6	*	*	*	6	6	6	6
F	F	F	F			F	F					F				F	F	F	F

(0.1 puntos) Optimo, número de fallos:___**7**

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	4
1	1	1	1	*	*	1	1	*	*	*	*	1	*	*	*	*	*	*	*
	2	2	2	*	*	2	2	*	*	*	*	2	*	*	*	*	*	*	*
		3	3	*	*	3	3	*	*	*	*	3	*	*	*	*	*	*	*
			4	*	*	4	4	*	*	*	*	4	*	*	*	*	*	*	*
						5	5	*	*	*	*	7	*	*	*	*	*	*	*
							6	*	*	*	*	6	*	*	*	*	*	*	*
F	F	F	F			F	F					F							

3. a) (0.3 puntos) Considere un sistema con 32Gbytes de memoría física, páginas de 8Kbytes y entradas en las tablas de páginas de 8 bytes. Con un nivel la TP ocupa una pagina. Con varios niveles la TP raíz o la de un nivel que no es el último, apunta a páginas que contienen TPs. ¿Cual es el tamaño en bytes del espacio virtual direccionable si se usa tablas de paginas de 1, 2 o 3 niveles?

De la respuesta en bytes y potencias de 2, y a continuación los cálculos. **Tanto la respuesta como el razonamiento deben ser correctos para puntuar la pregunta.**

Con la TP de un nivel: _____ $2^{\wedge 23}$ bytes
Con TPs de dos niveles: _____$2^{\wedge 33}$ bytes
Con TPs de tres niveles: _____$2^{\wedge 43}$ bytes

8KB = $2^{\wedge 13}$ bytes, por tanto cada TP tiene $2^{\wedge 13}$ / $2^{\wedge 3}$ = $2^{\wedge 10}$ entradas, y cada entrada apunta a una página de $2^{\wedge 13}$ bytes por tanto con una TP de un nivel se direccionan $2^{\wedge 10}$ x $2^{\wedge 13}$ = $2^{\wedge 23}$ bytes

Con TPs de dos niveles, la primera apunta a $2^{\wedge 10}$ tablas de páginas de segundo nivel y cada una de segundo nivel puede direccionar $2^{\wedge 23}$ bytes, por tanto se pueden direccionar $2^{\wedge 10}$ x $2^{\wedge 23}$ = $2^{\wedge 33}$ bytes

Con TPs de tres niveles, la primera apunta a $2^{\wedge 10}$ tablas de páginas de segundo nivel y cada una de segundo nivel

puede direccionar $2^{\wedge 33}$ bytes, por tanto se pueden direccionar $2^{\wedge 10} \times 2^{\wedge 33} = 2^{\wedge 43}$ bytes

b) (0.2 puntos) ¿Cuál es el número mínimo de bits necesarios en cada entrada de una tabla de páginas para poder direccionar toda la memoría física?
De la respuesta y a continuación los cálculos
Número de bits necesarios: _____**22 bits**

32GB = $2^{\wedge 35}$ bytes, ya que las páginas físicas de 8KB, es un total de $2^{\wedge 35} / 2^{\wedge 13} = 2^{\wedge 22}$ páginas. Por tanto se necesitan 22 bits en la entrada de tabla de páginas para indicar la página física.

c) (0.2 puntos) ¿Como es tamaño y formato de las direcciones lógicas en la arquitectura de 3 niveles?

Son de 43 bits. 10 bits para la entrada en la TP de primer nivel (raíz), 10 bits para la entrada en la TP de segundo nivel, 10 bits para la entrada en la TP de tercer nivel, 13 bits para el offset.

4. (0.3) Considere una arquitectura de tres niveles de TP como la del ejercicio anterior, un SO con memoria virtual y la situación en la que un proceso está en la cola de procesos listos para ejecución.

a) Las páginas de código de ese proceso tienen que estar en memoria.
b) Las páginas de código y pila de ese proceso tienen que estar en memoria.
c) Las páginas de código, datos y pila de ese proceso tienen que estar en memoria.
d) No es necesario que ninguna página de código, datos o pila de ese proceso esté en memoria.
e) Un registro del procesador dedicado a dar acceso a la TP, debe apuntar a la dirección virtual de la tabla de páginas de primer nivel.
f)Un registro del procesador dedicado a dar acceso a la TP, debe apuntar a la dirección física de la tabla de páginas de primer nivel.
g)Ninguna de las anteriores es cierta.

Indique la respuesta correcta y razónelo. **Tanto la respuesta como el razonamiento deben ser correctos para puntuar la pregunta.**
Respuesta correcta: _____d____
Imagine que el proceso se acaba de crear y que memoria virtual es con paginación por demanda. Cuando el proceso tome la CPU y se ejecute la primera instrucción se producirá un fallo de página que traerá la primera página del proceso a memoria y sucesivos fallos de páginas irán cargando en memoria las páginas necesarias. En este momento f) sería cierto pero no en la situación del enunciado, en esa situación habrá otro proceso en CPU y ese registro contiene dirección física de la TP de primer nivel de ese proceso.

SISTEMAS OPERATIVOS

Segundo curso Grado en Informática. Enero 2018

APELLIDOS, Nombre:_____

1	2	3	T
0.5	0.5	1	2

1. Considérese el siguiente código en C (se supone que contiene todos los ficheros include necesarios y que compila correctamente)

```
main(int argc, char * argv[])
{
  int i=100;
  pid_t pid;

  pid=fork();  /********/

  execl("./a.out", "./a.out",NULL);

  printf ("i vale: %d\n",i);
}
```

donde el código de `./a.out` es el siguiente: (tambien se supone contiene todos los ficheros include necesarios y que compila correctamente)

```
int i;
main (int argc, char *argv[])
{
    i++;
    printf ("i vale: %d\n",i);
}
```

a) **¿Qué salida produce dicho código?**

```
i vale: 1
i vale: 1
```

b) **¿Qué salida produciría si sustituimos la llamada *fork()* por una llamada *vfork()* en la línea marcada com /********/?**

```
i vale: 1
i vale: 1
```

Explicación: La llamada *fork()* crea un proceso, por tanto hay DOS procesos que llegan a la llamada *execl*. *execl* REEMPLAZA el espacio de direcciones por el de `./a.out`. En `./a.out` i es una variable externa sin inicialización explicita, y por tanto inicializada a 0, se incrementa y se imprime (el *printf* posterior a *exec* NO SE EJECUTA). Si sustituimos el *fork()* por *vfork()* la salida es la misma pues *vfork()* es una optimización de *fork()* en la que el proceso padre presta su espacio de direcciones al proceso hijo HASTA que se hace *exec*

2. Un sistema tiene 4 procesos, A, B, C, D cuyas duraciones son 3-(4)-1, 1-(6)-2, 4-(5)-3 y 6-(1)-1 respectivamente (x-(y)-z indica una ráfaga de CPU de duración x, seguida de una ráfaga de e/s de duración y, seguida de una ráfaga de CPU de duración z). El sistema tienen dos colas de planificación: la cola de alta prioridad que se planifica con Round Robbing de quanto 2 y la cola de baja prioridad que se planifica por FCFS. Un proceso de la cola de baja prioridad solo se ejecuta cuando no hay listo ningun proceso de la cola de alta prioridad, suponemos además que todos los procesos llegan en el instante 0 en el orden A,B,C,D y que pueden realizar e/s concurrentemente. Rellenar el siguiente cuadro con la planificación, indicando el cada instante que proceso está en CPU, listo en la cola de Alta Prioridad (HPQ), listo en la cola de Baja Prioridad(LPQ) o en E/S. Suponemos que A y B son procesos de Alta Prioridad y que C y D son procesos de Baja Prioridad

CPU	A	A	B	A	C	C	C	C	A	B	B	D	D	D	D	D	D	C	C	C	D	
HPQ	B	B	A																			
LPQ	C D	C D	C D	C D	D	D	D	D	D	D	D			C	C	C	C		D	D		
E/S				B	B A	B A	B A	B A	B C	 C	 C	 C	 C					D				

Calcular además los tiempos de retorno y espera para cada proceso.

	Ráfagas	Tiempo Retorno	Tiempo de Espera
Proceso A	3-(4)-1	9	1
Proceso B	1-(6)-2	11	2
Proceso C	4-(5)-3	20	8
Proceso D	6-(1)-1	21	13

3. Un sistema en tiempo real está atendiendo 4 eventos periódicos de periodos $T_1 = 100ms$, $T_2 = 50ms$ y $T_3 = 200ms$ $T_4 = 10ms$ y cuyos tiempos de procesamiento son respectivamente $C_1 = 5ms$, $C_2 = 2ms$ $C_3 = 100ms$ y $C_4 = 3ms$. Se quiere que atienda a un quinto evento periódico que ocurre cada 300ms. ¿Cual es la duración máxima de procesamiento de este evento para que el sistema siga siendo planificable?

Para que el sistema en tiempo real sea planificable se requeire que

$$\sum_i \frac{C_i}{T_i} \leq 1$$

en nuestro caso

$$\frac{5ms}{100ms} + \frac{2ms}{50ms} + \frac{100ms}{200ms} + \frac{3ms}{10ms} + \frac{x}{300ms} \leq 1$$

de donde la duración maxima de este quinto evento periódico es 33ms.

ENTRADA/SALIDA Puntuación:

EXAMEN DE SISTEMAS OPERATIVOS (Grado en Ing. Informática) 18/01/2018.

APELLIDOS Y NOMBRE: .....

1. **(0.75 puntos)** Complete las líneas `l.9` y `l.10` para que la salida del comando `ls` en la línea `l.14` se almacene en el fichero `out.txt`. Complete también las líneas `l.20` y `l.21` para que la salida del comando `ls` en la línea `l.25` se muestre por la salida estándar.

```
l. 0 #include "includes.h"
l. 1 int main() {
l. 2   char * comand[] = {"/bin/ls", "-l", "/", NULL};
l. 3   int fd, backup, status;  pid_t pid;
l. 4
l. 5   unlink("out.txt");
l. 6   if ( (fd=open("out.txt", O_CREAT|O_EXCL|O_WRONLY,0777) )== -1) {
l. 7      perror("OPEN FAILED:"); exit(0);
l. 8   }
l. 9   backup = dup(STDOUT_FILENO);  //respuesta
l.10   close (STDOUT_FILENO);        //respuesta
l.11
l.12   dup(fd);
l.13   if (fork() ==0) {
l.14      if (-1 == execv("/bin/ls", comand)){
l.15         perror("command failed:");exit(0);
l.16      }
l.17   }
l.18   waitpid(pid,&status);
l.19
l.20   close(STDOUT_FILENO);     //respuesta
l.21   dup(backup);              //respuesta
l.22
l.23   close(fd);close(backup);
l.24   if (fork() ==0) {
l.25      if (-1 == execv("/bin/ls", comand)){
l.26          perror("command failed:");exit(0);
l.27      }
l.28   }
l.29 }
```

2. **(0.75 puntos)** En un determinado instante la cola de peticiones de E/S para acceder a cilindros de un disco duro contenía las peticiones: $\langle 33, 341, 367, 440, 278, 101\rangle$. Desconocemos el cilindro X que estaba siendo accedido justo antes de estas peticiones, pero sabemos que X no estaba entre los cilindros de la lista de peticiones. Si sabemos que el orden de atención de

dichas peticiones fue: $\langle 367, 440, 33, 101, 278, 341\rangle$. Indíquese qué algoritmo/s de planificación de accesos a disco pudo/pudieron haber ser utilizados. Razone brevemente su respuesta. Adicionalmente, indíquese también UN posible número de cilindro (X) que podría estar siendo accedido antes de atender a dichas peticiones.

algoritmo	**posible cilind X**	**breve justificación de la respuesta**
C-SCAN o C-LOOK	$X \in [342, 366]$	El ascensor inicialmente asciende y atiende $\langle 367, 440\rangle$ Después vuelve a subir y atiende $\langle 33, 101, 278, 341\rangle$ Inicialmente el ascensor debe estar en el rango $[342, 366]$