Apellidos:_____ **Nombre:**_____ **DNI:** _____

Sistemas Operativos – Grado Ingeniera Informática UDC. Julio 2016
Sólo puede usar lápiz, bolígrafo y calculadora.
Tiempo máximo para todo el examen: 3h

Parte Sistema Ficheros
(Se deben contestar correctamente todas las cuestiones de cada pregunta para puntuar la misma).

P1) Un sistema de archivos tipo UNIX System V tiene un tamaño de bloque de 2Kbytes, i-nodos con 10 direcciones directas, una indirecta simple, una indirecta doble y una indirecta triple. Utiliza direcciones de bloque de 8 bytes. Calcular el tamaño máximo de un fichero al utilizar los niveles de indirección simple, doble y triple.

Tamaño máximo usando puntero de indirección simple: **20Kbytes + 512 Kbytes**

Tamaño máximo usando puntero de indirección doble:
20Kbytes + 512 Kbytes + 128 Mbytes

Tamaño máximo usando puntero de indirección triple:
20Kbytes + 512 Kbytes + 128 Mbytes + 32 Gbytes

P2) En ese sistema de archivos UNIX (tamaño de bloque de 2Kbytes), el tamaño de un inodo es de 64 bytes. El superbloque mantiene un mapa de bits de inodos libres para determinar los inodos libres/ocupados de la lista de inodos. Ese mapa de bits, que es parte del superbloque, ocupa un total de 2 bloques. Calcular cuántos bloques son necesarios para la zona de inodos en el disco (lista de inodos).

Número de bloques que ocupa la lista de inodos en disco: **1Kbloques**

Apellidos:_____ **Nombre:**_____ **DNI:** _____

P3) Un proceso abre (sin retorno de ningún error) el fichero *"datos"* (descriptor *fd1*) y, posteriormente, duplica el descriptor de fichero abierto con el servicio *dup*, al ejecutar el siguiente código:

```
main()
{
int fd1, fd2;
char buffer[2048];

fd1=open("datos", O_RDONLY);
lseek(fd1, 700, SEEK_SET); /* SEEK_SET referencia desde el principio del fichero */
fd2=dup(fd1);
read (fd2, buffer, 512);
read (fd1, buffer, 512);
lseek(fd2, 900, SEEK_SET);
close(fd1);
close(fd2);
}
```

Contestar a lo siguiente.

A. ¿Qué valor se guarda en la variable *fd2* después de ejecutar el servicio *dup()*?.

Valor fd2: **4**

B. Indicar si es cierto o falso que, después de ejecutar el servicio *dup()*, el desplazamiento asociado al descriptor *fd2* queda en la posición 700.

Cierto

C. Indicar si es cierto o falso que, después de ejecutar la segunda invocación a *lseek()*, el desplazamiento asociado al descriptor *fd2* queda en la posición 1412.

Falso

D. Las dos lecturas con las llamadas *read* leen la misma información del fichero (mismo rango de bytes).

Falso

E. Las dos lecturas con las llamadas *read* leen la información necesariamente del disco al existir *Buffer Cache*.

Falso

Apellidos:_____ **Nombre:**_____ **DNI:** _____

P4) Supongamos la siguiente secuencia de comandos:

1. Se crean 2 *hard link* al fichero *"datos"* (cuyo número de inodo es 53418, tamaño 8 Gbytes y num. de *hard links*=1) en su mismo directorio:

ln datos hlink1

ln datos hlink2

2. Posteriormente se crea un *soft link* al fichero *hlink1*:

ln -s hlink1 slink1 / crea soft link slink1 */*

/ el comando ls -l mostraría slink -> hlink1 */*

y otro un *soft link* al fichero *hlink2*:

ln -s hlink2 slink2 / crea soft link slink2 */*

/ el comando ls -l mostraría slink2 -> hlink2 */*

Contestar a lo siguiente:

A. Indicar el tamaño (en Gbytes, Mbytes o bytes) del fichero *slink1*: **6 bytes**

B. Indicar cuál es el número de *(hard) links* del fichero *hlink2*: **3**

C. Una vez creados los ficheros *slink1* y *slink2*, al borrar el fichero datos (*rm datos*) se decrementa su número de *hard links*. ¿Se puede acceder al contenido del fichero con número de inodo 53418 a través del link simbólico *slink2?*

Sí

Misma pregunta a través del fichero *slink1*:

Sí

SISTEMAS OPERATIVOS I

Segundo curso Grado en Informática. Julio 2016

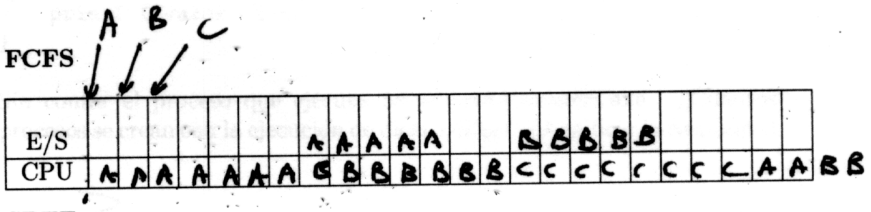
APELLIDOS, Nombre:_____

* En un sistema tenemos 3 procesos: A (ráfagas 7-**(5)**-2), B (ráfagas 7-**(5)**-2) y C (una única ráfaga de 8), que llegan en los instantes 0 1 y 2 respectivamente (x-**(y)**-z representa una ráfaga de CPU de duración x, seguida de una ráfaga e E/S de duración y, continuada con una ráfaga de CPU de duación z). Planifíquense con los algoritmos FCFS SRTF, y RR de quanto 2. Supondremos que **cuando un proceso llega a la CPU** (nuevo o proveniente de una E/S) **en el mismo instante que otro abandona la CPU,** el que abandona la CPU se coloca al final la cola de listos, detrás del que llega en ese instante (nuevo o procedente de E/S). Supondremos además que dos procesos pueden estar en E/S a la vez.

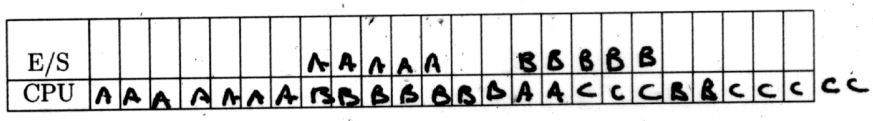
1) El enunciado indica que la única ráfaga de C es de CPU. ¿podría ser de E/S? es decir, ¿podría C estar constituido por una sola ráfaga de E/S? (justifíquese la respuesta)

   NO. Todo proceso comienza(por lo menos la parte del hijo de la llamada al sistema para crear proceso y ejecutar programa) y termina (por lo menos la llamada al sistema para terminar proceso) con una rágafa de CPU. Si tiene una sola ráfaga es de CPU

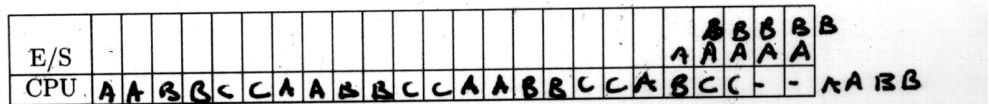
2) **FCFS**



3) **SRTF**



4) **RR de quanto 2**



* Supongamos que el siguiente código no contiene errores tipográficos y compila correctamente en un programa que llamaremos `examen.exe`

```
#include <unistd.h>
#include <stdio.h>

#define MAX 10

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

```
   int i,j;

   fork();
   fork();

   for (i=0; i<MAX; i++){
      execl("./a.out","./a.out",NULL);
      for (j=0; j<MAX/2; j++)
          printf ("%d",i);
      }
   printf ("Terminando %d\n",i);
}
```

Supongamos además que el ejecutable `./a.out` existe, es un ejecutable válido, que hay permisos de ejecución para todos los usuarios del sistema (`rwxr-xr-x`) y que su código fuente es el siguiente

```
#include <stdio.h>

int i;

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

5) Sin contar el proceso que ejecuta inicialmente `examen.exe`. ¿Cuántos procesos se crean con la ejecución de dicho código? (Justificar la respuesta)

SE CREAN 3 PROCESOS. La primera llamada fork() crea un proceso. A la segunda llamada fork() llegan dos procesos (el original y el creado en la primera llamada fork()) y cada uno de ellos crea otro proceso. Por tanto hay TRES procesos creados: el creado por el proceso original en la primera llamada fork ($hijo_1$), el creado por el proceso original en la segunda llamada fork() ($hijo_2$) y el creado por $hijo_1$ en la segunda llamada fork() ($nieto_1$)

6) ¿Cuál es la salida COMPLETA del proceso original (del que ejecuta inicialmente `examen.exe`)?

LA SALIDA ES: `valor:1`

Los cuatro procesos (el original y los tres creados) hacen exactamente lo mismo: ejecutan sólo la primera iteración del bucle *for*, y ahí reemplazan su código por el de `./a.out`.

`./a.out` **imprime el valor de la variable i, que como es variable externa SI está inicializada: el valor inicial es 0**

Apellidos:_____ Nombre:_____ DNI: _____

Sistemas Operativos – Grado Ingeniera Informática UDC. SegundaOportunidad 2016
Sólo puede usar lápiz, bolígrafo y calculadora.
Tiempo máximo para todo el examen: 3h

Parte Memoria – Total 2.3 puntos

En las preguntas 1 y 3 tienen que estar correctos los resultados y los cálculos y/o razonamientos

1. Considere un sistema con un tiempo de acceso a memoria de 100ns y tablas de páginas en tres niveles.
   a) (0.25) Si el sistema tiene una razón de fallos de páginas del 0.01% y el tiempo de servicio de un fallo de página es de 1ms, ¿cuál es el tiempo efectivo de acceso a memoria? (suponga también que no hay TLB ni cachés): _____800 ns____
   * Observe que los fallos de página pueden producirse en cualquier acceso a memoria, sean páginas del proceso o tablas de páginas, incluso en la TP de primer nivel que no se supone fija en memoria

Tal y como se dijo el peor caso y más fácil de calcular sería suponer que en el 0.01% de las referencias hay 4 fallos de página, uno por cada TP más uno por la página que contiene el dato o instrucción que se referencia:

(1-0.0001) x 400 ns + 0.0001 x 4 x 1000000 ns = 0.9999 x 400 ns + 400 ns ≈ 800 ns.

Por supuesto los que consideraron que las tres TP estuvieran siempre en memoria y que su acceso se hiciese siempre en 300 ns más los 1ms del fallo de página en el 0.01% de referencias que producen fallo de página, se les dió la máxima puntuación si lo hicieron bien. Lo mismo con los que lo hicieron con cualquier escenario intermedio si justificaron su escenario e hicieron bien los cálculos.

b) (0.25) Suponga ahora que no hay fallos de página. Se añade un TLB con un tiempo de acceso de 1 ns. ¿Que porcentaje de acierto se necesita en el TLB para reducir el tiempo efectivo de acceso a memoria a sus 2/5 partes?: _____80%_____

Si no hay fallos de página el tiempo de acceso efectivo en memoria serían 400 ns y eso es lo que se reduce en 2/5, pero para que no hubiese ninguna duda sobre ese dato, eso se dijo en el examen. Sea PA el porcentaje de acierto en el TLB

(2/5) x 400 = 160 ns
101 x PA + (1- PA) x 401 = 160
PA ≈ 0,80

2. (0.25) Para una tabla de páginas invertida (TPI) marque con un X las que sean ciertas (puede haber cero, una o varias correctas). Si marca como correcta una incorrecta, se valorará con cero. Para alcanzar la puntuación máxima debe marcar sólo y todas las correctas (o ninguna si fuese el caso).

__ El número de entradas en la TPI está dado por el tamaño del espacio de direcciones lógicas del proceso.
X El número de entradas en la TPI está dado por el tamaño de la memoría física
__ El número de entradas en la TPI está dado por el tamaño de la memoría virtual

__ Con esta solución a la traslación de direcciones, no se pueden producir fallos de página
X Dos páginas lógicas de dos procesos distintos no pueden apuntar a la misma página física, lo que implica no compartición de páginas.

3. Considere un sistema de memoria virtual con tablas de páginas en dos niveles, direcciones virtuales de 32 bits donde los 12 bits menos significativos son para el offset, los 10 más significativos para la entrada en la TP de primer nivel y los 10 siguientes para la entrada en la TP de segundo nivel. Cada TP es de tamaño 1 página, que es lo habitual. Las direcciones físicas tienen los 20 bits más significativos para el número de página física y los 12 menos significativos para el offset. Las entradas en las TP tienen 32 bits cuyo significado se muestra en el orden de bits más significativos a menos significativos, y este mismo orden es de almacenamiento en memoria.
   20 bits: número de página física
   3 bits: reservados
   1 bit: 0
   1 bit: reservado
   1 bit: dirty bit
   1 bit: bit de acceso
   1 bit: no cache
   1 bit: write-through
   1 bit: 1 user, 0 kernel
   1 bit: 1 escritura, 0 read-only
   1 bit: página válida

   a) (0.15) Támaño en bytes de la página lógica: ___2^{12} = 4096 bytes ____
   Tamaño en bytes de la página física: ___2^{12} = 4096 bytes ____

   b) (0.25) Suponga un proceso que sólo necesita un página al principio de sus espacio de direcciones y otra al final. ¿Qué tamaño en bytes ocuparía en tablas de paǵinas?:
   ___3 x 4096 = 12228 bytes___

   La TP de primer nivel tendrá todos sus punteros nulos excepto el primero y el último que apunta a tablas de páginas de segundo nivel. Por tanto se necesitan 3 tablas de páginas = ___3 páginas___

   c) (0.25) ¿Cuál sería en bytes el tamaño máximo empleado por un proceso en tablas de páginas en este sistema? : ____1025 x 4096 = 4198400 bytes____

   Si todos los punteros de la TP de primer nivel son no nulos, serían 4096/4= 1024, es decir, apuntarían a 1024 tablas de página de segundo nivel y todas ellas se necesitarían para trasladar páginas virtuales a páginas físicas en el caso de que un proceso utilice todo el espacio de direcionamiento posible.

   d) (0.9) Suponga que para un proceso el registro base a la TP de primer nivel contiene el valor (en hexadecimal) 0x00200000. Indique el resultado de las siguientes operaciones de memoria a partir de los contenidos de memoria física que se adjuntan. Una operación Load devuelve unvalor de 8 bits o un error. Una operación Store devuelve Ok o error. Errores posibles son página inválida, read-only o kernel-only.

Load 0x00001047 Resultado:_____0x50____

número de página de primer nivel= 0
número de página de primer nivel= 1
offset en la página = 0x047

entrada 0 (primera entrada) TP de primer nivel =0x00100007
número página física TP segundo nivel= 0x00100
dir física base TP segundo nivel = 0x00100000
entrada 1 (segunda entrada) TP de segundo nivel = 0x00002067, acaba en binario en 111: modo user, permiso escritura, página válida

número de página 0x00002
dir física base de la página 0x00002000
dir física base de la página + offset = 0x00002047
contenido de la dir 0x00002047 = **0x50**

Store 0x00C07665 Resultado:____OK____
0000 0000 1100 0000 0111 0110 0110 0101

número de página de primer nivel= 3
número de página de primer nivel= 7
offset en la página = 0x665

entrada 3 TP de primer nivel =0x00103007
número página física TP segundo nivel= 0x0010300
dir física base TP segundo nivel = 0x0010300000
entrada 7 TP de segundo nivel = 0xEEFF0067, acaba en binario en 111: modo user, permiso escritura, página válida, **Store OK**

número de página 0xEEFF0
dir física base de la página 0xEEFF0
dir física base de la página + offset = 0xEEFF0665

Store 0x00C005FF Resultado: ___Error: read-only___
0000 0000 1100 0000 0000

número de página de primer nivel= 3
número de página de primer nivel= 0
offset en la página = 0x5FF

entrada 3 TP de primer nivel =0x00103007
número página física TP segundo nivel= 0x0010300
dir física base TP segundo nivel = 0x0010300000
entrada 0 TP de segundo nivel = 0x11220055, acaba en binario en 101: modo user, sólo lectura, página válida, **Store da Error: read-only**

Load 0x00003012 Resultado: ___0x84____

número de página de primer nivel= 0
número de página de primer nivel= 3
offset en la página = 0x012

entrada 0 (primera entrada) TP de primer nivel =0x00100007
número página física TP segundo nivel= 0x00100
dir física base TP segundo nivel = 0x00100000
entrada 3 (cuarta entrada) TP de segundo nivel = 0x00004007, acaba en binario en 111: modo user, permiso escritura, página válida

número de página 0x00004
dir física base de la página 0x00004000
dir física base de la página + offset = 0x00004012
contenido de la dir 0x00004012 = **0x84**

Physical Memory [All Values are in Hexidecimal]

Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
00000000	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
00000010	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D
....																
00001010	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
00001020	40	03	41	01	30	01	31	03	00	03	00	00	00	00	00	00
00001030	00	11	22	33	44	55	66	77	88	99	AA	BB	CC	DD	EE	FF
00001040	10	01	11	03	31	03	13	00	14	01	15	03	16	01	17	00
....																
00002030	10	01	11	00	12	03	67	03	11	03	00	00	00	00	00	00
00002040	02	20	03	30	04	40	05	50	01	60	03	70	08	80	09	90
00002050	10	00	31	01	10	03	31	01	12	03	30	00	10	00	10	01
....																
00004000	30	00	31	01	11	01	33	03	34	01	35	00	43	38	32	79
00004010	50	28	84	19	71	69	39	93	75	10	58	20	97	49	44	59
00004020	23	03	20	03	00	01	62	08	99	86	28	03	48	25	34	21
....																
00100000	00	00	10	65	00	00	20	67	00	00	30	00	00	00	40	07
00100010	00	00	50	03	00	00	00	00	00	00	00	00	00	00	00	00
...																
00103000	11	22	00	05	55	66	77	88	99	AA	BB	CC	DD	EE	FF	00
00103010	22	33	44	55	66	77	88	99	AA	BB	CC	DD	EE	FF	00	67
...																
001FE000	04	15	00	00	48	59	70	7B	8C	9D	AE	BF	D0	E1	F2	03
001FE010	10	15	00	67	10	15	10	67	10	15	20	67	10	15	30	67
...																
001FF000	00	00	00	00	00	00	00	65	00	00	10	67	00	00	00	00
001FF010	00	00	20	67	00	00	30	67	00	00	40	65	00	00	50	07
...																
001FFFF0	00	00	00	00	00	00	00	00	10	00	00	67	00	10	30	65
...																
00200000	00	10	00	07	00	10	10	07	00	10	20	07	00	10	30	07
00200010	00	10	40	07	00	10	50	07	00	10	60	07	00	10	70	07
00200020	00	10	00	07	00	00	00	00	00	00	00	00	00	00	00	00
...																
00200FF0	00	00	00	00	00	00	00	00	00	1F	E0	07	00	1F	F0	07
...																