Apellidos:_____Nombre:_____ DNI: _____

Sistemas Operativos – Grado Ingeniera Informática UDC. Enero 2019
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: 0.7, P2: 0.5, P3: 0.5, P4: 0.3

P1) Un sistema de archivos tipo *system V* tiene un tamaño de bloque de 4 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 4 bytes. Consideremos el fichero "p1.c" en el directorio /home/usr1/so/p1, con un tamaño de 4Gbytes+6 Mytes+45 KBytes.

Calcular cuántos bloques de disco son necesarios para representar ese archivo. Indicar también cuánta fragmentación interna (en Kbytes) se produce.

Nº bloques de datos: 1Mbloques + 1.5Kbloques + 12 = 1050124

Nº de bloques de índices: 1029

Fragmentación interna del fichero: 3 Kbytes

P2) Supongamos la siguiente secuencia de comandos desde el directorio /home/usr1/so/p1:

i) Se crea un *hard link* al fichero *p1.c* (cuyo número de inodo es 11545, tamaño 4Gbytes+6 Mytes+45 KBytes y num. de *hard links*=3):
ln p1.c practica1.c / crea hard link practica1.c */*

ii) Posteriormente se crea un *soft link* al archivo *practica1.c*:
ln -s practica1.c slink / el comando ls -l mostraría slink -> practica1.c */*

iii) Finalmente se crea un *hard link* al fichero *slink*:
ln slink hlinkslink / crea hard link hlinkslink */*

Contestar a lo siguiente:

A. Indicar cuál es el número de (*hard*) *links* del fichero *hlinkslink* después de las tres operaciones: 2

B. Indicar el tamaño del fichero *hlinkslink*: 11

C. Una vez creados los ficheros *slink* y *hlinkslink*, al borrar el fichero *p1.c* (*rm p1.c*) se puede seguir accediendo al contenido del fichero con número de inodo 11545 a través del *link simbólico slink* (Cierto/Falso): C

D. Con un sistema de ficheros Unix basado en registro (*journaling file system*), al borrar el fichero *p1.c*, primero se realiza una copia de su inodo en el registro antes de liberar sus bloques del área de datos. (Cierto/Falso): F

P3) Después de crear el *hard link practica1.c* en el apartado *i)* del ejercicio previo, un proceso abre ese archivo "/home/usr1/so/p1/practica1.c" en modo lectura 2 veces seguidas. El usuario efectivo del proceso coincide con el propietario del fichero (practica1.c), y tiene además los permisos de acceso y lectura a los directorios *home*, *usr1, so y p1*. Las cachés de datos e inodos están inicialmente vacías y la entrada *p1* está en el sexto bloque de su directorio padre (*so)*, mientras las 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/so/p1/practica1.c", 0664);
if (lstat ("/home/usr1/so/p1/practica1.c", &buf)!=-1){
    printf("%s", convertir_permisos(buf.st_mode));     /* se convierten los permisos a
                                                          formato rwxrwxrwx y se imprimen*/
    int fd1=open("/home/usr1/so/p1/p1.c", O_RDONLY);  /* es la primera apertura
                                                          de fichero del proceso */
    int fd2=open("/home/usr1/so/p1/practica1.c", O_RDONLY);  /* segunda apertura */
    lseek(fd2, 524288, SEEK_SET);          /* SEEK_SET indica que el desplazamiento
                                              se considera a partir del origen del fichero
                                              (524288 = 2^19)  */
   c=fgetc(fd2);
   close(fd2); close(fd1);
   }
```

A. ¿Cuál es el número de accesos necesarios a disco, únicamente en el área de datos, en la apertura del fichero practica1.c?: 10
B. ¿Cuál es el valor asignado al descriptor de fichero *fd2*?: 4
C. Indica el número de bloques que el S.O. necesita leer en disco para obtener el valor de *c* en *fgetc(fd2*): 2
D. *fgetc* es una llamada al sistema operativo. (Cierto/Falso): F
E. Indica los permisos del fichero *p1.c* impresos en formato *rwxrwxrwx*: rw-rw-r--

P4) En ese sistema de archivos UNIX (tamaño de bloque de 4Kbytes), el tamaño de la lista de inodos en disco es de 16 Mbytes. 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 8 bloques. Calcular el tamaño (en bytes) de un inodo.

Tamaño inodo: 64 bytes

Apellidos:_____Nombre:_____ DNI: _____

Sistemas Operativos - Grado Ingeniera Informática UDC. Enero 2019
Sólo puede usar lápiz, bolígrafo y calculadora.
Tiempo máximo para todo el examen: **2h45m**

Parte Memoria.

1. (0.7 puntos) Responda Verdadero/Falso (o no conteste) a cada pregunta sin justificaciones, pero cada respuesta incorrecta anula una correcta.

-Un sistema puede tener direcciones virtuales de 32 bits y direcciones físicas de 40 bits.
Respuesta: ____V____

-Un sistema puede tener direcciones virtuales de 40 bits y direcciones físicas de 32 bits.
Respuesta: ____V____

-Un sistema puede tener 12 bits para el offset en una página virtual y 14 bits para el offset en una página física
Respuesta: _____F___

-Un sistema puede tener 14 bits para el offset en una página virtual y 12 bits para el offset en una página física
Respuesta: _____F___

-En clase se estudió una aproximación al modelo de gestión de memoria del Working Set (WS) que se implementa con un *timer*, bit de referencia de las páginas y copia en memoria de los bits de referencia. El modelo teórico puro de WS no se implementa porque se necesitaría computar el WS en cada referencia a memoria.
Respuesta: _____V___

-Un problema con los sistemas que trabajaban con código absoluto es que la resolución de la referencias a memoria en el código era muy lenta
Respuesta: _____F___

-Los sistemas de memoria basados en un registro base y un registro límite pueden tener swapping (intercambio) de procesos
Respuesta: ____V____

-En un sistema de gestión de memoria con tabla de páginas invertidas tiene que haber tantas tablas de páginas invertidas como procesos en la memoria física
Respuesta: ____F____

-En un sistema de gestión de memoria con segmentación paginada y con TLB, la TLP mapea páginas lógicas en páginas físicas
Respuesta: ___V_____

-Un sistema de segmentación puede tener fragmentación externa por tanto uno de segmentación paginada también
Respuesta: ____F____

2. Considere el siguiente código y de la respuesta correcta (0.2 puntos)

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
char *f(char *a) {

        static char *b;
        b=a;
        return b;
}
int main () {

        char a[30] = "Hello f\n";
        printf(f(a));
        strcpy(a, "Bye f\n");
        printf(f(a));
}
```

a) Da error en tiempo de ejecución por asignar una variable global (a) a una estática (b)
b) Da error en tiempo de ejecución por asignar una variable local (a) a una estática (b)
c) Producen la salida pretendida, esto es:

```
Hello f
Bye f
```

porque nada lo impide
d) Produce la salida

```
Hello f
Hello f
```

porque b es estática y no puede cambiar de valor
e) Da error en tiempo de compilación porque printf no puede llevar de argumento una función
f)Niguna de las anteriores es cierta

Respuesta: _____C_____

3. Considere el siguiente código y de la respuesta correcta (0.2 puntos)

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
char *f(char *a) {

        static char *b;
        b=a;
        return b;
}
int main () {

        char *a = malloc(sizeof(char)*30);
        strcpy(a,"Hello f\n");
        printf(f(a));
        strcpy(a, "Bye f\n");
        printf(f(a));
}
```

a) Da error en tiempo de ejecución por asignar una variable dinámica (a) a una estática (b)
b) Da error en tiempo de ejecución por asignar una variable local (a) a una estática (b)
c) Da error en tiempo de compilación porque printf no puede llevar de argumento una función
d) Producen la salida pretendida, esto es:

```
Hello f
Bye f
```

porque nada lo impide
e) Produce la salida

```
Hello f
Hello f
```
porque b es estática y no puede cambiar de valor
f)Niguna de las anteriores es cierta

Respuesta: _____D_____

4. (0.6 puntos) Supón un esquema de paginación con dos niveles de tablas de pagínas, con páginas de 4Kbytes, entradas en las tablas de páginas de 4 bytes y direcciones virtuales de 32 bits. Si no hay información suficiente para responder alguna pregunta, indíquelo.

-¿Cuál es el formato de las direcciones virtuales?
Páginas de 4Kbytes (2^12 bytes) y 4 bytes por entrada, por tanto 2^10 entradas en cada TP, por tanto 10 bits para indicar la entrada en la TP de primer nivel y la de seugndo nivel, por tanto las direcciones virtuales son
10 bits para entrada TP primer nivel, 10 bits para entrada TP segundo nivel, 12 bits para offset en la página

-¿Cuantos bits de control y reservados hay en cada entrada de una tabla de páginas?
En cada entrada de 4bytes se usa un número de bits para indicar el número de frame y el resto para bits de control y reservados pero se desconoce esta información

-¿Cual sería el tamaño máximo en tablas de páginas que puede ocupar un proceso?
1025 = 1 + 2^10, es decir, 1025 páginas de 4Kb, o (2^12 bytes + 2^22 bytes), o 4Kbytes + 4Mbytes

-Para un proceso con 3 páginas (una de código, una de datos, una de pila) y con la disposición habitual del espacio de direcciones virtuales, ¿cuál es el tamaño ocupado en tablas de páginas para este proceso?
La TP de primer nivel. La primera entrada de la TP de primer nivel apunta a una TP de segundo nivel que apunta a las páginas de código y datos. La última entrada de la TP de primer nivel apunta a una TP de segundo nivel que apunta a la página de pila. Por tanto 3 páginas en tablas de páginas, es decir 12Kbytes.

-Para el proceso del apartado anterior, si la página de datos no está en la memoria física, ¿cuál es el tamaño ocupado en tablas de páginas para este proceso?
Idem que la cuestión anterior

-¿Cuál es el tamaño del espacio de direcciones físicas expresado en número de frames?
Se desconoce el número de bits reservado para indicar el número de frame, por tanto no se puede responder a esta cuestión

5. (0.3 puntos) En un sistema con memoria virtual y segmentación paginada, suponga que la programación es correcta, es decir, que las invocaciones de free llevan como argumento un puntero obtenido con malloc y que nunca se ha accedido a bytes de memoria que no fuesen obtenidos con malloc. Una buena práctica de programación es liberar con free todos los bloques de memoria asignados con malloc antes de que un proceso termine. Para estas invocaciones de free de la respuesta correcta, no es necesario justificarla
a) algún free puede provocar fallo de segmento por acceso página inválida

b) algún free puede provocar fallo de segmento por acceso a segmento inválido
c) algún free puede provocar fallo de página por acceso a una página no usada en mucho tiempo
d) algún free puede provocar fallo de segmento si la página fue modificada y salvada al dispositivo de swapping
e) algún free puede provocar fallo de página si la página fue modificada y no salvada al dispositivo de swapping
f) ninguna de las anteriores es cierta

Respuesta correcta: _____C_____

SISTEMAS OPERATIVOS

Segundo curso Grado en Informática. Enero 2019

APELLIDOS, Nombre:_____

1	2	3	T
0.5	1	0.5	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=fork()+ fork();  /********/

  if (pid==0)
     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? (suponemos que tanto fork() como exec() se completan correctamente)

i vale 100

i vale 1

i vale 100

i vale 100 (no necesariamente en ese orden)

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

la misma, ya que ninguno de los procesos hijos modifica variables desu proceso padre. (la i del programa original y la i de a.out son variables distintas)

2. Un sistema tiene 4 procesos, A, B, C, D cuyas duraciones son 1-(6)-3, 2-(6)-1, 3-(5)-6 y 1-(1)-4 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 SJF. 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	B	B	D	C	C	C	A	A	B	A	D	D	D	D	C	C	C	C	C	C
HPQ	B									A											
LPQ	C D	C D	C D	C	D	D	D	D	D	D	D		C	C	C						
E/S		A	A	A B	A B D	A B	A B	C B	C B	C	C	C									

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

	Ráfagas	Tiempo Retorno	Tiempo de Espera
Proceso A	1-(6)-3	11	1
Proceso B	2-(6)-1	10	1
Proceso C	3-(5)-6	21	7
Proceso D	1-(1)-4	15	9

3. ¿Cual o cuales, si es que alguna, de las siguientes afirmaciones son ciertas en un sistema unix, referidas a la pila del kernel?

   (a) La bateria que mantiene la configuración del kernel cuando el sistema se apaga

   (b) Zona de memoria usada para para pasar parámetros a funciones y variables locales cuando un proceso se ejecuta en modo kernel

   (c) Lo que proporciona un pulso de corriente al procesador para cambiar a modo kernel

   (d) Zona de memoria donde se almacenan las variables de entorno y las credenciales de los usuarios del sistema

   (e) Solo hay una en el sistema

   (f) Hay una para cada procesador/núcleo

   (g) Hay una para cada proceso

   (h) Debe ser de solo lectura

   (i) Hay una para cada thread

   Las correctas son: b), g) i) Justificación:

   b) es prácticamente la definición de pila del kernel. i) es cierta si el proceso tiene varios threads y el S.O. permite además que distintos threads del mismo proceso realicen llamadas al sistema concurrentemente. Para procesos de un solo thread g) es correcta

Apellidos: ..... **Nombre:** ..... **DNI:** .....

Sistemas Operativos – Grado Ingeniería Informática UDC. Enero 2019
Sólo puede usar lápiz, bolígrafo y calculadora. Smartphones apagados y en la mochila.
Tiempo máximo para todo el examen: **2h45m**

Parte E-S [1.50 puntos]

1. (**1.00 puntos**) Disponemos de un disco duro con 1000 cilindros numerados del 0 al 999. Sabemos que en las dos últimas peticiones atendidas se accedió a los cilindros 402 y al 400 respectivamente, y que la cabeza está ahora situada sobre el cilindro 400.

   Si en el instante actual la cola de peticiones de acceso a cilindros contiene las peticiones: $\langle 501, 625, 250, 552, 99, 800\rangle$, indíquese en qué orden se atenderán dichas peticiones si se utiliza un algoritmo de planificación:

FCFS o FIFO:	501	625	250	552	99	800
SSTF:	501	552	625	800	250	99
SCAN:	250	99	501	552	625	800
C-SCAN:	250	99	800	625	552	501
C-LOOK:	250	99	800	625	552	501

   Nota, asúmase que no llegarán más peticiones mientras se atienden las peticiones actuales.

Nota: Recuérdese que los prototipos de las funciones read, readp y readv son los siguientes:

```
ssize_t read (int fd, void *buf, size_t count);
ssize_t pread(int fd, void *buf, size_t count, off_t offset);
ssize_t readv(int fd, const struct iovec *iov, int iovcnt);
off_t lseek(int fd, off_t offset, int whence);
```

2. (**0.5 puntos**) El fichero "fichero.txt" contiene: `"012345678901234546789\n"`, y sabemos que al ejecutar el siguiente código no se produjo ningún error de ejecución:

```
#include "includes.h"
int main(){
int fd = open("fichero.txt",O_RDONLY);
lseek(fd, 5, SEEK_SET);
char BUF[5]= {'-','-','-','-','\0'};
```

```
#ifdef CASO_A
 read(fd,BUF,1);
 read(fd,BUF+2,2);
#endif
```

```
#ifdef CASO_B
 pread(fd,BUF,1,0);
 lseek(fd, 2, SEEK_CUR);
 pread(fd,BUF,2,3);
 read(fd,BUF,1);
#endif
```

```
#ifdef CASO_C
 struct iovec iov[2];
 iov[0].iov_base= BUF;
 iov[0].iov_len= 3;
 iov[1].iov_base= BUF+3;
 iov[1].iov_len= 2;
 readv(fd,iov,2);
#endif
```

```
printf("%s",BUF); close(fd);
}
```

Asumiendo que el contenido del buffer **BUF** ($BUF[i], i = 0 \ldots 4$) en función de si está definido CASO_A, CASO_B, o CASO_C es el siguiente:

BUF[i], i=	CASO_A:				
	5	–	6	7	\0
	0	1	2	3	4

i=	CASO_B:				
	7	4	–	–	\0
	0	1	2	3	4

i=	CASO_C:				
	0	1	2	3	4
	0	1	2	3	4

Indíquese en qué casos, el estado de **BUF** representa la realidad: Indique **V**/**F** (Verdadero/Falso) y razone los casos en los que considere que es **F**also.

	V/F	**Respuesta razonada**
CASO_A:	V	
CASO_B:	V	
CASO_C:	F	Se lee respectivamente "567" y "89"