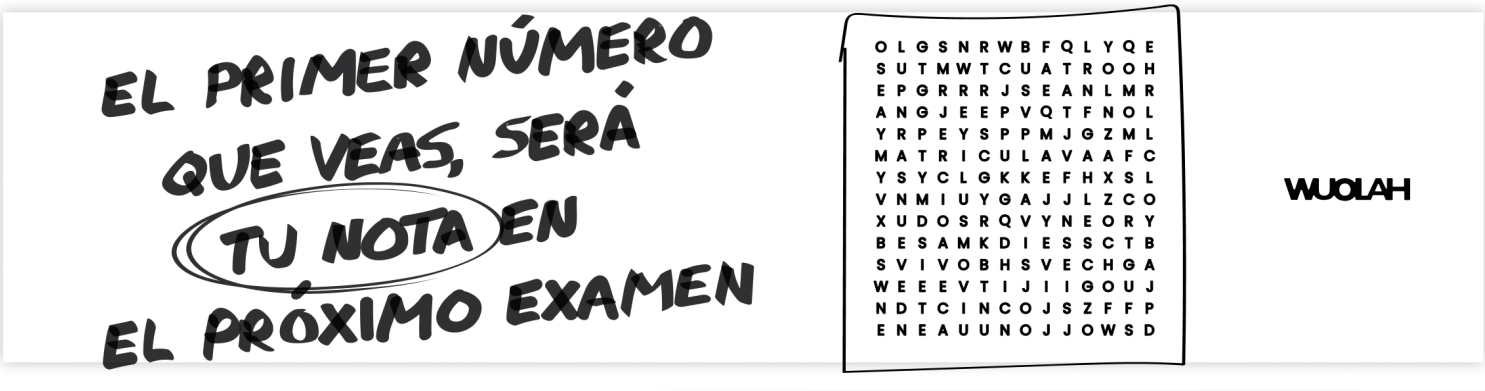
SO2022-01.pdf

_lopezzz

Sistemas Operativos

2º Grado en Ingeniería Informática

Facultad de Informática
Universidad de A Coruña



Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

SIBUYA URBANSUSHIBAR
Hay rollos y rollos... Me vas a comparar tu rollo de ayer con un Hot Roll, claramente el sushi está más bueno
HAZ CLIC PARA HACER TU RESERVA

Apellidos:_____ **Nombre:** Antón ____ **DNI:** _____

Sistemas Operativos – Grado Ingeniería Informática UDC. Enero 2022
Sólo puede usar lápiz, bolígrafo y calculadora.

Parte **Sistema Ficheros (**2 puntos)

Examen sin identificación completa NO se califica.
Esta parte NO admite entrega de hojas adicionales. Hay que contestar en los espacios proporcionados.

Puntuación preguntas: P1: 0.6, P2: 0.3, P3: 0.5, P4:0.6

P1) Un sistema de archivos tipo UNIX System V tiene un tamaño de bloque de 2Kbytes, i-nodos con 12 direcciones directas, una indirecta simple, una indirecta doble y una indirecta triple. Utiliza direcciones de bloque de 4 bytes. Calcular el tamaño máximo de un fichero al utilizar los niveles de indirección simple, doble y triple. Indicar también el número de bloques de índices (máximo) que se usa exclusivamente en cada nivel de indirección.

Un sistema de ficheiros tipo UNIX System V ten un tamaño de bloque de 2 Kbytes, i-nodos con 12 enderezos directos, un indirecto simple, un indirecto dobre e un indirecto triple. Usa enderezos de bloque de 4 bytes. Calcula o tamaño máximo dun ficheiro ao utilizar os niveis de indirección simple, dobre e triple. Indicar tamén o número de bloques de índices (máximo) que se usa exclusivamente en cada nivel de indirección

A UNIX file system (System V type) has a block size of 2Kbytes, i-nodes with 12 direct addresses, a single indirect, a double indirect and a triple indirect address. It uses 4-byte block addresses. Calculate the maximum size of a file when using the levels of single, double and triple indirection. Also indicate the (maximum) number of index blocks that is used exclusively at each level of indirection.

Tamaño máximo usando puntero de indirección simple:
Tamaño máximo usando punteiro de indirección simple:
Maximum size using single indirection pointer: **1Mbyte + 24Kbytes** (0.05p)

Número máximo de bloques de índices usando solo indirección simple:
Número máximo de bloques de índices usando só indirección simple:
Maximum number of index blocks using only single indirection: **1** (0.05p)

Tamaño máximo usando puntero de indirección doble:
Tamaño máximo usando punteiro de indirección doble:
Maximum size using double indirection pointer: **0.5Gbytes + 1Mbyte + 24Kbytes** (0.1p)

Número máximo de bloques de índices usando solo indirección doble:
Número máximo de bloques de índices usando só indirección doble:
Maximum number of index blocks using only double indirection: **513** (0.1p)

Tamaño máximo usando puntero de indirección triple:
Tamaño máximo usando punteiro de indirección triple:
Maximum size using triple indirection pointer: **256 Gbytes + 0.5Gbytes + 1Mbyte + 24Kbytes** (0.15p)

Número máximo de bloques de índices usando solo indirección triple:
Número máximo de bloques de índices usando só indirección triple:
Maximum number of index blocks using only triple indirection: **256K + 512 +1** (262657) (0.15p)

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

P2) En el sistema de archivos UNIX previo (tamaño de bloque de 2Kbytes), el *boot* ocupa los 3 primeros bloques de su partición de disco. Por tanto, el superbloque comienza en el bloque lógico 3 de la partición de disco del SF (se numera a partir del bloque 0). Un fichero “datos” tiene asociado el *inodo* 325 y ocupa el bloque (lógico) 30 desde el principio de la partición. El tamaño del *inodo* es de 128 bytes. Calcula el tamaño (en Kbytes) del *superbloque*.

Tamaño del superbloque : **14** (Kbytes)

No sistema de ficheiros UNIX previo (tamaño de bloque de 2 Kbytes), o *boot* ocupa os 3 primeiros bloques da súa partición de disco. Polo tanto, o superbloque comeza no bloque lóxico 3 da partición de disco do SF (númerase a partir do bloque 0). Un ficheiro “datos” ten asociado o inodo 325 e ocupa o bloque (lóxico) 30 desde o inicio da partición. O tamaño do inodo é de 128 bytes. Calcula o tamaño (en Kbytes) do superbloque.

Tamaño do superbloque : _____ (Kbytes)

In the previous UNIX file system (2Kbytes block size), the *boot* occupies the first 3 blocks of its disk partition. Therefore, the superblock begins at logical block 3 of the file system disk partition (it is numbered starting from block 0). A file “datos” has the inode 325 associated and occupies the (logical) block 30 from the beginning of the partition. The inode size is 128 bytes. Calculate the size (in Kbytes) of the superblock.

Size of the superblock: _____ (Kbytes)

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

si lees esto me debes un besito

Las miraditas en la biblioteca me gustan muy calientes... y el sushi flambeado

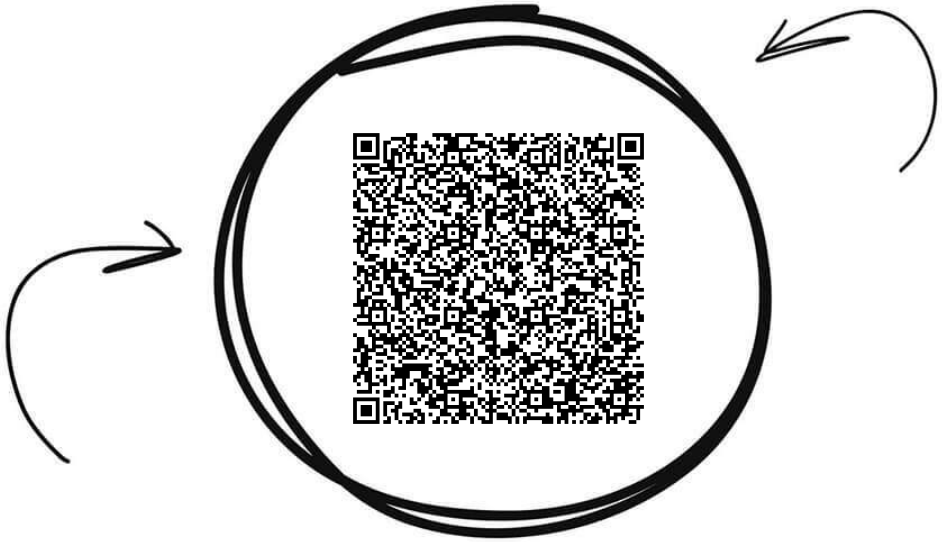
SIBUYA ES OTRO ROLLO

PRAZA DA UNIVERSIDADE 3 • SANTIAGO

HAZ CLIC PARA HACER TU

RESERVA

Sistemas Operativos



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

P3) Un proceso abre 2 veces el archivo *"/home/user1/so/practicas/p1.c"*, cuyo número de *hard links* inicial es 1. El tamaño del fichero es de 32 Gbytes y se suponen los datos referentes al sistema de ficheros del ejercicio P1 (tamaño bloque, punteros acceso directos e indirectos). El usuario efectivo del proceso coincide con el propietario del fichero, y tiene además los permisos de acceso a los directorios *raíz*, *home*, *user1, so y practicas*. Las cachés de datos e inodos están inicialmente vacías y la entrada *user1* está en el tercer bloque del directorio *home*, mientras las demás entradas están en el primer bloque de sus directorios padre. Indicar lo siguiente referente al trozo de código:

Un proceso abre duas veces o ficheiro *"/home/user1/so/practicas/p1.c"*, cuxo número de *hard links* inicial é 1. O tamaño do ficheiro é de 32 Gbytes e se supoñen os datos referentes ao sistema de ficheiros do exercicio P1 (tamaño bloque, punteiros acceso directos e indirectos). O usuario efectivo do proceso coincide co propietario do ficheiro e tamén ten os permisos de acceso os directorios *raíz, home*, *user1, so e practicas*. Os cachés de datos e inodos están inicialmente baleiros e a entrada *user1* está no terceiro bloque do directorio *home*, mentres que as outras entradas están no primeiro bloque dos seus directorios pai. Indique o seguinte sobre o anaco de código:

A process opens twice the file "*/home/user1/so/practicas/p1.c*", whose initial number of *hard links* is 1. The file size is 32 Gbytes and the data referring to the file system from exercise P1 is assumed (block size, direct and indirect access addresses). The effective user of the process coincides with the owner of the file, and also has the access permissions to the directories *root*, *home*, *user1*, *so* and *practicas*. The data and inode caches are initially empty and the entry *user1* is in the third block of the directory *home*, while the other entries are in the first block of their parent directories. Indicate the following regarding the piece of code:

```
chmod("/home/user1/so/practicas /p1.c", 0752);
link("/home/user1/so/practicas /p1.c", "/home/user1/so/practicas /p1_hlink.c");
symlink("/home/user1/so/practicas /p1.c", "/home/user1/so/practicas/p1_slink"); /*crea link simbólico p1_slink */
                                                                                /* crea link simbólico p1_slink */
                                                                                /* symbolic link p1_slink is created*/
int fd1=open("/home/user1/so/practicas /p1.c", O_RDONLY); /* es la primera apertura de fichero del proceso */
                                                          /* é a primeira apertura de ficheiro do proceso */
                                                          /* it is the first file open of the process*/
int fd2=open("/home/user1/so/practicas /p1_hlink.c", O_RDWR);
lseek(fd2, 1.000.000, SEEK_SET);/*SEEK_SET indica que el desplazamiento se considera a partir del origen del fichero */
                                /* SEEK_SET indica que o desprazamento considérase desde a orixe do ficheiro */
                                /* SEEK_SET specifies that the offset is considered from the origin of the file */
char c=fgetc(fd2);
close(fd1); close(fd2);
unlink("/home/user1/so/practicas /p1_slink");
```

Cada apartado puntúa 0.1p/ Each question scores 0.1p.

A. ¿Cuál es el valor asignado al descriptor de fichero *fd2*?:
Cal é o valor asignado ao descriptor de ficheiro *fd2*?:
What is the value assigned to the file descriptor *fd2*?: **4**

B. ¿Cuál es el número de accesos necesarios a disco, únicamente en el área de datos, en la primera apertura del fichero?:
Cal é o número de accesos ao disco necesarios, só na área de datos, ao abrir o ficheiro a primeira vez?:
What is the number of disk accesses required, only in the data area, in the first file opening?: **7**

C. Indica el número de bloques que el SO necesita leer en disco para obtener el valor de *c* en *fgetc(fd2)*:
Indica o número de bloques que o SO cómpre ler desde o disco para obter o valor de *c* en *fgetc (fd2)*:
Indicate the number of blocks that the OS requires to read from disk to get the value of *c* in *fgetc (fd2)*: **2**

D. Indica los permisos del fichero en formato *rwxrwxrwx*:
Indica os permisos de ficheiro en formato *rwxrwxrwx*:
Indicate the file permissions in *rwxrwxrwx* format: **rwx r-x -w-**

E. ¿Cuál es el número de *hard links* de "p1.c" después de ejecutar *unlink*?:
Cal é o número de ligazóns duros (*hard links*) de "p1.c" despois de executar *unlink*?:
What is the number of *hard links* of "p1.c" after running *unlink*?: **2**

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

si lees esto me debes un besito

P4) Indicar si es cierto/falso en cada pregunta.

Cada apartado puntúa 0.1p. Cada respuesta errónea puntúa -0.1. Cuestiones no respondidas no puntúan. La puntuación mínima de P4 es 0, es decir, en ningún caso P4 lleva a puntuación negativa para el total del examen.

A. Una llamada al sistema se puede implementar con una interrupción software, que cambia el modo de ejecución del procesador. **C**

B. Al ejecutar la función de entrada/salida de C *printf*, el proceso está ejecutando exclusivamente en modo sistema. **F**

C. El código binario de *lseek()* se encuentra en la librería estándar de C (*libC*). **F**

D. Las librerías estáticas permiten compartir código entre procesos en tiempo de ejecución. **F**

E. La fragmentación externa no existe con un sistema de ficheros con asignación indexada de los bloques de datos. **C**

F. El bit *UID* y el bit pegajoso (*sticky bit*) forman parte del campo de "modo" del inodo. **C**

Indique se é verdadeiro/falso en cada pregunta.

Cada apartado puntúa 0.1p. Cada resposta incorrecta puntúa -0.1. Cuestións non respondidas non puntúan. A puntuación mínima para P4 é 0, é dicir, en ningún caso P4 ten puntuación negativa para o exame total.

A. Unha chamada ao sistema pódese implementar cunha interrupción software, que cambia o modo de execución do procesador. ____

B. Ao executar a función de entrada/salida de C *printf*, o proceso está executando exclusivamente en modo sistema. ____

C. O código binario de *lseek()* se atopa na librería estándar de C (*libC*). ____

D. As librerías estáticas permiten compartir código entre procesos en tempo de execución. ____

E. A fragmentación externa non existe cun sistema de ficheiros con asignación indexada dos bloques de datos. ____

F. O bit *UID* e o bit pegajoso (*sticky bit*) forman parte do campo de "modo" do inodo. ____

Indicate whether it is true/false for each question.

Each question scores 0.1p. Each wrong answer scores -0.1. Unanswered questions do not score.The minimum score for P4 is 0, that is, in no case does P4 have a negative score for the total exam.

A. A system call can be implemented with a software interrupt, which changes the execution mode of the processor. ____

B. When executing the input/output C function *printf*, the process is executing exclusively in system mode. ____

C. The binary code for *lseek()* is in the standard C library (*libC*). ____

D. Static libraries allow code sharing between processes at runtime. ____

E. External fragmentation does not exist with a filesystem with indexed allocation of data blocks. ____

F. The bit *UID* and the *sticky bit* are part of the inode's "mode" field. ____

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

si lees esto me debes un besito

SIBUYA URBANSUSHIBAR
Hay rollos y rollos... Me vas a comparar tu rollo de ayer con un Hot Roll, claramente el sushi está más bueno
HAZ CLIC PARA HACER TU RESERVA

Apellidos:_____Nombre:_____ DNI: _____

Examen sin identificación completa NON se califica
Sistemas Operativos - GEI UDC. Xaneiro 2022. Memoria (2 puntos)
Esta parte NON admite entrega follas adicionais. Hay que contestar nas páxinas e espacios proporcionados.

1. **Cada una de las preguntas siguientes son de responder V/F (Verdadero/Falso) o no responder. Respuesta correcta: +0.1; Respuesta incorrecta: -0.1. (0.1 x 10 = 1 punto) Puntuación mínima para esta pregunta: 0.**
Cada unha das preguntas seguintes son de responder V/F (Verdadeiro/Falso) ou non responder. Resposta correcta: +0.1; Resposta incorrecta: -0.1. (0.1 x 10 = 1 punto) Puntuación mínima para esta pregunta: 0.
Answer T/F (True/False) or leave the answer blank. Correct answer: 0.1; Wrong answer: -0.1 (0.1 x 10 = 1 point) Minimum score for this question: 0.

1	2	3	4	5	6	7	8	9	10
F	F	F	F	F	V	V	V	V	F

1. **En los sistemas sin paginación que intercambiaban procesos enteros a disco, no podía haber relocalización de direcciones, ni estática ni dinámica, y si ejecución de código absoluto.**
Nos sistemas sin paxinación que intercambiaban procesos enteiros a disco, non podía haber relocalización de direccións, nin estática nin dinámica, e si execución de código absoluto.
In non-paging systems that swapped entire processes to disk, there could be no address relocation, neither static nor dynamic, and there could be absolute code execution.

2. El código está en las direcciones bajas del espacio de direcciones virtuales de los procesos para que el acceso a las instrucciones sea más rápido.
O código está nas direccións inferiores do espazo de direccións virtuais dos procesos para que o acceso ás instrucións sexa máis rápido.
The code is in the lower addresses of the virtual address space of the processes so that the access to the instructions is faster.

3. Un servicio de fallo de página de un proceso necesita poner al proceso en el estado de bloqueado intercambiado a disco.
Un servizo de fallo de páxina dun proceso debe poñer o proceso no estado de bloqueado intercambiado a disco.
A process's page fault service needs to put the process in the blocked swapped out state.

4. Los algoritmos de reemplazo de página FIFO y FIFO segunda oportunidad, tienen el mismo coste de implementación.
Os algoritmos de reemplazo de páxinas FIFO e FIFO segunda oportunidade teñen o mesmo custo de implementación
The FIFO and FIFO second chance page replacement algorithms have the same implementation cost.

5. Con la llamada al sistema execve() se copian las tablas de páginas del proceso que invoca execve(), proceso padre, en el proceso hijo creado con execve().
Coa chamada ao sistema execve(), as táboas de páxinas do proceso que invoca execve(), proceso pai, cópianse no proceso fillo creado con execve()
With the execve() system call, the page tables of the process that invokes execve(), parent process,

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

are copied to the child process created with execve ()

6. Con una llamada al sistema fork() en un sistema Linux, se copian las áreas de memoria vm_areas del proceso padre en el proceso hijo.
Cunha chamada de sistema fork () nun sistema Linux, as áreas de memoria vm_areas do proceso pai cópianse no proceso fillo..
With a fork () system call on a Linux system, the memory areas (vm_areas) from the parent process are copied to the child process.

7. De entre las funciones de librería para la gestión de memoria dinámica, la función free() puede llevar como argumento una variable estática.
Entre as funcións da biblioteca para a xestión dinámica da memoria, a función free () pode tomar unha variable estática como argumento.
Among the functions of the library for dynamic memory management, the free () function can take a static variable as an argument.

8. Considere un sistema con memoria virtual, paginación por demanda, asignación fija de N marcos de memoria a cada proceso y algoritmo de reemplazo aleatorio. Para un proceso con N-1 páginas virtuales se producen N-1 o menos fallos de página.
Considere un sistema con memoria virtual, paxinación baixo demanda, asignación fixa de N marcos de memoria a cada proceso e un algoritmo de reemplazo aleatorio. Para un proceso con N-1 páxinas virtuais, ocorren N-1 ou menos fallos de páxina..
Consider a system with virtual memory, on-demand paging, fixed allocation of N memory frames to each process, and a random replacement algorithm. For a process with N-1 virtual pages, there are N-1 or fewer page faults.

9. Considere un sistema con memoria virtual, paginación por demanda, asignación fija de N marcos de memoria a cada proceso y algoritmo de reemplazo FIFO. Para un proceso con N-1 páginas virtuales se producen N-1 o menos fallos de página.
Considere un sistema con memoria virtual, paxinación baixo demanda, asignación fixa de N marcos de memoria a cada proceso e o algoritmo de reemplazo FIFO. Para un proceso con N-1 páxinas virtuais, ocorren N-1 ou menos fallos de páxina..
Consider a system with virtual memory, on-demand paging, fixed allocation of N memory to each process, and the FIFO replacement algorithm. For a process with N-1 virtual pages, there are N-1 or fewer page faults.

10. Considere direcciones virtuales de 48 bits y tabla de páginas en 1 nivel con entradas de 4 bytes y páginas de 8 Kbytes. La tabla de páginas ocupa 2^{24} bytes.
Considere direccións virtuais de 48 bits e táboa de páxinas dun nivel con entradas de 4 bytes e páxinas de 8 kbytes. A taboa de páxinas ocupa 2^{24} bytes.
Consider 48 bits virtual addresses and a 1 level Page Table with 4 byte entries and 8 Kbyte pages. The page table occupies 2^{24} bytes.

2. (0.5 puntos)
a) Un proceso tiene la cadena de referencias a páginas que se muestra y tiene asignados cuatro marcos de memoria. Las cuatro primeras referencias, estos es, la referencias a las páginas 2, 4, 6, 1, producen necesariamente 4 fallos de página porque ninguna página del proceso estaba en memoria. ¿Cual es el número total de fallos de páginas que se producen con el algoritmo de reemplazo LRU en las 10 primeras referencias? Obviamente hay que contar esos 4 fallos iniciales en el total. Tanto la asignación de páginas a frames como el total de número de fallos deben ser correctos para puntuar la pregunta.

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

si lees esto me debes un besito

Un proceso ten a cadea de referencias a páxinas que se mostra e ten asignadas catro marcos de memoria. As catro primeiras referencias, isto é, as referencias as páxinas 2, 4, 6, 1 producen necesariamente 4 fallos de páxina porque ningunha páxina do proceso estaba en memoria. ¿Cal é o número total de fallos de páxina que se producen co algoritmo de reemplazo LRU nas 10 primeiras referencias? Obviamente hay que contar esos 4 fallos iniciais no total. Tanto a asignación de páxinas a frames como o total de número de fallos deben ser correctos para puntuar a pregunta.

The string of page references for a process that has been allocated 4 page frames is the one shown below. The first four page references, i.e. references to pages 2, 4, 6, 1, produce 4 page faults because none of the pages of this process were in memory. What is the total number of page faults applying the LRU replacement algorithm after 10 references? Obviously, you have to include the initial 4 faults in the total amount. Both the assignment of pages to frames and the total number of page faults must be right to get the score for this question.

Respuesta: __10__ fallos

Cadena de referencias. String of page references:

2	4	6	1	3	2	4	6	1	3

Asignación de pags a marcos. Asignación de paxs a marcos. Assignment of pages to frames

2	4	6	1	3	2	4	6	1	3
2	2	2	2	3	3	3	3	1	1
	4	4	4	4	2	2	2	2	3
		6	6	6	6	4	4	4	4
			1	1	1	1	6	6	6
F	F	F	F	F	F	F	F	F	F

b) Idem para Algoritmo de reemplazo Fifo segunda Oportunidad

Respuesta: __10__ fallos

Cadena de referencias. String of page references:

2	4	6	1	3	2	4	6	1	3

Asignación de pags a marcos. Asignación de paxs a marcos. Assignment of pages to frames

2	2	2	2	3	3	3	3	1	1
	4	4	4	4	2	2	2	2	3
		6	6	6	6	4	4	4	4
			1	1	1	1	6	6	6
F	F	F	F	F	F	F	F	F	F

3. (0.5 puntos) Para cada una de las preguntas siguientes responda o deje en blanco: Respuesta correcta: +0.05; Respuesta incorrecta: -0.0125. (0.05 x 10 = 0.5 puntos) Puntuación mínima para esta pregunta: 0.
El código que se muestra se compila y ejecuta correctamente y muestra 10 direcciones. Debe indicar para cada dirección una de las respuestas posibles dada por la localización de esa

WUOLAH

si lees esto me debes un besito

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

dirección en una de las 5 posibles que se muestran (a,b,c,d,e) :
a) Espacio Direcciones Virtuales del proceso, area de stack
b) Espacio Direcciones Virtuales del proceso, area de datos dinámicos
c) Espacio Direcciones Virtuales del proceso, areas de datos globales y estáticos
d) Espacio Direcciones Virtuales del proceso, areas de código propio o librerías
e) Espacio Direcciones Virtuales del kernel
Para cada unha das preguntas seguintes responda ou deixe en blanco: Resposta correcta: +0.05; Resposta incorrecta: -0.0125. (0.05 x 10 =0.5 puntos) Puntuación mínima para esta pregunta: 0. O código que se mostra compila e execútase correctamente e mostra 10 direccións. Debes indicar para cada enderezo unha das posibles respostas dadas pola localización desa dirección nunha das 5 posibles que se indican (a, b, c, d, e):
a) Espazo de direccións virtual do proceso, área de pila
b) Espazo de direccións virtual do proceso, área de datos dinámicos
c) Espazo de direccións virtual do proceso, áreas de datos globais e estáticos
d) Espazo de direccións virtuais do proceso, áreas de código propio ou bibliotecas
e) Espazo de direccións virtuais do kernel
For each of the following questions, answer or leave the answer blank. Correct answer: +0.05; Wrong answer: -0.0125 (0.05 x 10 = 0.5 point) Minimum score for this question:
The code shown compiles and runs successfully and shows 10 addresses. You must indicate for each address one of the possible answers given by the location of that address in one of the 5 possible ones shown (a, b, c, d, e):
a) Virtual address space of the process, stack area
b) Virtual address space of the process, dynamic data area
c) Virtual process addresses space, global and static data areas
d) Virtual address space of the process, own code areas or libraries
e) Kernel Virtual Address space

dir-1	dir-2	dir-3	dir-4	dir-5	dir-6	dir-7	dir-8	dir-9	dir-10
a	a	b	b	c	c	b	d	d	d

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

double t;

void f1()
{
  static double pi =3.1415926 ;
  char *p = malloc(4096);
  t=pi;
  {
   auto int a=33;
  printf ("dir-1:     %p\n",&a);
     }
  printf ("dir-2:     %p\n",&p);
  printf ("dir-3:     %p\n",&p[0]);
  printf ("dir-4:     %p\n",p);
  printf ("dir-5:     %p\n",&pi);
  printf ("dir-6:     %p\n",&t);
  printf ("dir-7:     %p\n",sbrk(0)-0xFF);
  printf ("dir-8:     %p\n",f1);
  printf ("dir-9:     %p\n",getpid);
  printf ("dir-10:     %p\n",fork);
}
```

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

el podcast para entender que la vida da mas vueltas que la silla de un peluquero


```
int main(int argc, char *argv[])
{
  f1();
  exit(0);
}
```

Con esta promo,
te llevas **5€** por
tu cara bonita al
subir **3 apuntes**
a Wuolah
Wuolitah

WUOLAH

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

APELLIDOS:_____ **NOMBRE:**_____ **DNI:** _____

Sistemas Operativos (PROCESOS) – Grado Ingeniería Informática UDC. Enero 2022
-Apellidos en MAYUSCULA.
-Examen sin nombre equivale a NO PRESENTADO)

Q1.-(0.8 puntos) Responda Verdadero/Falso (o no conteste) a cada pregunta (0.1 cada una). Respuesta incorrecta -0.1. La puntuación mínima es cero para esta pregunta. (rellenar en el espacio correspondiente de la siguiente tabla)/*Answer V/F (True/False) or leave the answer blank. Correct answer: 0.1;Wrong answer: -0.1 (0.1 x 8 = 0.8 points) Minimum score for this question: 0/* Cada unha das preguntas seguintes son de responder V/F (Verdadeiro/Falso) ou non responder. Resposta correcta: +0.1; Resposta incorrecta: -0.1. Puntuación mínima para esta pregunta: 0

1	2	3	4	5	6	7	8
V	***F***	***F***	***F***	***V***	***F***	***F***	***V***

1-Un proceso que NO ES DEL ROOT, SIEMPRE comienza su ejecución en modo kernel
A process of a user other than ROOT, ALWAYS starts its execution in kernel mode
Un proceso que NON É DO ROOT, SEMPRE comeza a súa execución en modo kernel

2-En un sistema tipo UNIX donde hay 15 procesos de 6 usuarios distintos ejecutando el mismo programa, hay 6 copias en memoria del código de dicho programa
In a UNIX type system where there are 15 processes of 6 different users executing the same program, there are 6 copies in memory of the code of said program
Nun sistema tipo UNIX onde hai 15 procesos de 6 usuarios diferentes que executan o mesmo programa, hai 6 copias na memoria do código do devandito programa.

3.-Cuando se crea un proceso mediante fork, su pid (Process IDentifier) es 0
When a process is created by fork, its pid (Process IDentifier) is 0
Cando se crea un proceso con fork, o seu pid (identificador de proceso) é 0

4.-La estructura de procesos en un sistema UNIX es un grafo
The process structure on a UNIX system is a graph
A estrutura dos procesos nun sistema UNIX é un gráfo

5-La tabla de procesos del sistema está almacenada en la zona de datos del kernel
The system process table is stored in the kernel data area
A táboa de procesos do sistema almacénase na zona de datos do núcleo

6-El código de la rutina de servicio de la interrupción de teclado es parte de los datos del kernel
The keyboard interrupt service routine code is part of the kernel data
O código de rutina do servizo de interrupción do teclado forma parte dos datos do núcleo

7-La llamada *execvp* devuelve 0 si ha podido ejecutar el programa que se le pasa como argumento y -1 si ha habido algún tipo de error
The execvp call returns 0 if it has been able to execute the program that is passed to it as an argument and -1 if there has been some kind of error
A chamada execvp devolve 0 se foi capaz de executar o programa que se lle pasa como argumento e -1 se houbo algún tipo de erro

8-Despues de hacer una de las llamadas exec (execl, execv....) sobre un fichero setuid, el PID del

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

si lees esto me debes un besito

APELLIDOS:_____ **NOMBRE:**_____ **DNI:** _____

proceso es el mismo
After making one of the exec calls (execl, execv ....) on a setuid file, the PID of the process is the same
Despois de facer unha das chamadas exec (execl, execv ....) nun ficheiro setuid, o PID do proceso é o mesmo

Q2.-(0.6 puntos) Sea el siguiente código en C, con todos los includes necesarios y que compila correctamente
Consider the following C code, with all the necessary includes and that compiles correctly
Sexa o seguinte código en C, con todas os includes necesarios e que compilar correctamente

```
#define VECES 2

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

  for (i=0; i<VECES; i++){
      pid=fork();
      if (pid==-1)
          break;
      if (pid==0)
          fork();
      printf ("/%ld/*\n",(long) pid);
  }
}
```

a) ¿cuántas líneas de salida produce? *How many output lines does it produce?.* ¿cantas liñas de saida produce?
12
b)¿cuántas de ellas son «/0/*»?-*How many of them are «/0/*»?* ¿cántas delas son «/0/*»?
8
Explicación: En la primera iteración del bucle (i=0), en el primer fork() se crea un proceso (la variable pid vale 0, para el hijo y distinto de 0 para el padre). Luego el proceso hijo ejecuta otro fork() (if (pid==0), creando otro proceso en le que, como es una copia del hijo, su variable pid vale 0;.Los tres procesos llegan al printf y se imprimen tres líneas, dos de ellas "/0/". La segunda iteracción del bucle es igual que la primera, con la diferencia de que la ejecutan los tres procesos que completaron la primera, de manera que se imprimirán 9 líneas, 6 de ellas "/0/". Total 12 lineas imprimidas de las que 8 son "/0/"

Q3.-(0.6 puntos) En linux los procesos en tiempo real se planifican con prioridades estáticas de 1 a 99 de manera que se ejecuta siempre el proceso de la máxima prioridad (mayor número indica mayor prioridad). Los procesos en tiempo real pueden ser SCHED_RR (se planifican en round-robin con un quanto de 100ms) y SCHED_FIFO (se planifican por FCFS). En un sistema llegan (en ese orden), en un instante dado 3 procesos en tiempo real, A, B y C

In Linux, real-time processes are scheduled with static priorities from 1 to 99 so that the process with

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

si lees esto me debes un besito

APELLIDOS:_____ **NOMBRE:**_____ **DNI:** _____

the highest priority is always executed (a higher number indicates a higher priority). The real-time processes can be SCHED_RR (they are scheduled in round-robin with a quanto of 100ms) and SCHED_FIFO (they are scheduled by FCFS). Three real time proceses arrive at a given time in the order A,B,C

En Linux, os procesos en tempo real planifícanse con prioridades estáticas de 1 a 99 de xeito que se execute sempre o proceso con maior prioridade (un número maior indica unha maior prioridade). Os procesos en tempo real poden ser SCHED_RR (en round-robin cun canto de 100ms) e SCHED_FIFO (FCFS). Nun sistema chegan (por esa orde), nun momento dado 3 procesos en tempo real, A, B e C

Proceso	**PRIORIDAD ESTATICA**	**TIPO**	**RAFAGAS CPU-(I/O)-CPU**
A	70	SCHED_RR	500-(800)-300
B	60	SCHED_RR	200-(900)-100
C	70	SCHED_FIFO	300-(400)-100

a) Calcular los tiempos de retorno y de espera de A, B y C./ ~~Calculate the turaround and waiting times for A, B & C./~~ Calcula os tempos de retorno e espera para A, B e C.

	Tiempo retorno	Tiempo de espera
Proceso A	***1900***	***300***
Proceso B	***2100***	***900***
Proceso C	***900***	***100***

b) ¿Cuánto tiempo queda disponible (si es que alguno) para los procesos normales (prioridad estática 0) desde que llegan estos procesos hasta que han terminado los tres?/*How much time (if any) is available for normal processes (static priority 0) from the moment these processes arrive until all three have finished?/* Canto tempo (se hai) queda dispoñible para os procesos normais (prioridade estática 0) desde que chegan estes procesos ata que remataron os tres?

Como se ve en la tabla quedan 600ms para los otros procesos (cada cuadro representa 100ms).

Se ejecuta siempre el de mas prioridad, y si hay dos procesos listos con la misma prioridad se ejecuta el SCHED_RR durante un quanto y el SCHED_FIFO hasta el final de su ráfaga.

(puede usarse la siguiente tabla, aunque no es lo que se va a corregir)

cpu	A	C	C	C	A	A	A	A	C	B	B						A	A	A		B	
e/s					C	C	C	C	A	A	A	A B	A B	A B	A B	A B	B	B	B	B		

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

si lees esto me debes un besito

5€ DE BIENVENIDA

APELLIDOS:_____ **NOMBRE:**_____ **DNI:** _____

Sistemas Operativos (PROCESOS) Grado Ingeniería Informática UDC. Enero 2022
-Apellidos en MAYUSCULA. ***MODELO B***
-Examen sin nombre equivale a NO PRESENTADO)

Con esta promo, te llevas **5€** por tu cara bonita al subir **3 apuntes** a Wuolah
Wuolitah

Q1.-(0.8 puntos) Responda Verdadero/Falso (o no conteste) a cada pregunta (0.1 cada una). Respuesta incorrecta -0.1. La puntuación mínima es cero para esta pregunta. (rellenar en el espacio correspondiente de la siguViente tabla)*/Answer V/F (True/False) or leave the answer blank. Correct answer: 0.1;Wrong answer: -0.1 (0.1 x 8 = 0.8 points) Minimum score for this question: 0/* Cada unha das preguntas seguintes son de responder V/F (Verdadeiro/Falso) ou non responder. Resposta correcta: +0.1; Resposta incorrecta: -0.1. Puntuación mínima para esta pregunta: 0

1	2	3	4	5	6	7	8
V	***V***	***V***	***V***	***F***	***V***	***F***	***F***

1-Un proceso que NO ES DEL ROOT, SIEMPRE termina su ejecución en modo kernel
A process of a user other than ROOT, ALWAYS ends its execution in kernel mode
Un proceso que NON É DO ROOT, SEMPRE remata a súa execución en modo kernel

2-En un sistema tipo UNIX donde hay 15 procesos de 6 usuarios distintos ejecutando el mismo programa, hay solo 1 copia en memoria del código de dicho programa
In a UNIX type system where there are 15 processes of 6 different users executing the same program, there is only 1 copy in memory of the code of said program
Nun sistema tipo UNIX onde hai 15 procesos de 6 usuarios diferentes que executan o mesmo programa, hai so 1 copia na memoria do código do devandito programa.

3.-Cuando se crea un proceso mediante fork, su pid (Process IDentifier) es mayor 0
When a process is created by fork, its pid (Process IDentifier) is greater than 0
Cando se crea un proceso con fork, o seu pid (identificador de proceso) é maior 0

4.-La estructura de procesos en un sistema UNIX es un árbol
The process structure on a UNIX system is a tree
A estrutura dos procesos nun sistema UNIX é unha árbore

5-La tabla de procesos del sistema es parte del código del kernel
The system process table is part of the kernel code
A táboa de procesos do sistema e parte do código do núcleo

6-El código de la rutina de servicio de la interrupción de teclado es parte del código del kernel
The keyboard interrupt service routine code is part of the kernel code
O código de rutina do servizo de interrupción do teclado forma parte do código du núcleo

7-La llamada *execvp* devuelve >=0 si ha podido ejecutar el programa que se le pasa como argumento y -1 si ha habido algún tipo de error
The execvp call returns >=0 if it has been able to execute the program that is passed to it as an argument and -1 if there has been some kind of error
A chamada execvp devolve >=0 se foi capaz de executar o programa que se lle pasa como argumento e -1 se houbo algún tipo de erro

8-Despues de hacer una de las llamadas exec (execl, execv....) sobre un fichero setuid, el UID

WUOLAH
WUOLAH
Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

APELLIDOS:_____ **NOMBRE:**_____ **DNI:** _____

efectivo del proceso es el mismo
After making one of the exec calls (execl, execv ....) on a setuid file, the effective UID of the process is the same
Despois de facer unha das chamadas exec (execl, execv ....) nun ficheiro setuid, o UID efectivo do proceso é o mesmo

Q2.-(0.6 puntos) Sea el siguiente código en C, con todos los includes necesarios y que compila correctamente
Consider the following C code, with all the necessary includes and that compiles correctly
Sexa o seguinte código en C, con todas os includes necesarios e que compilar correctamente

```
#define VECES 2

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

  for (i=0; i<VECES; i++){
      pid=fork();
      if (pid==-1)
          break;
      if (pid>=0)
          fork();
      printf ("/%ld/*\n",(long) pid);
  }
}
```

a) ¿cuántas líneas de salida produce? *How many output lines does it produce?.* ¿cantas liñas de saida produce?
20
b)¿cuántas de ellas son «/0/*»? *How many of them are «/0/*»?* ¿cántas delas son «/0/*»?
10
Explicación: En la primera iteración del bucle (i=0) en el primer fork() se crea un proceso (la variable pid vale 0, para el hijo y distinto de 0 para el padre). Luego tanto el proceso padre como e hijo ejecutan otro fork() (if (pid>=0), creando otros dos procesos que son copias, con los mismos valores de las variables. Los cuatro procesos llegan al printf y se imprimen cuatro líneas, dos de ellas "/0/". La segunda iteracción del bucle es igual que la primera, con la diferencia de que la ejecutan los cuatro procesos que completaron la primera, de manera que ahora se imprimirán 16 líneas, 8 de ellas "/0/".
Total: 20 lineas imprimidas de las que 10 son "/0/"
Q3.-(0.6 puntos) En linux los procesos en tiempo real se planifican con prioridades estáticas de 1 a 99 de manera que se ejecuta siempre el proceso de la máxima prioridad (mayor número indica mayor prioridad). Los procesos en tiempo real pueden ser SCHED_RR (se planifican en round-robin con un quanto de 100ms) y SCHED_FIFO (se planifican por FCFS). En un sistema llegan (en ese orden), en un instante dado 3 procesos en tiempo real, A, B y C

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

si lees esto me debes un besito

APELLIDOS:_____ **NOMBRE:**_____ **DNI:** _____

In Linux, real-time processes are scheduled with static priorities from 1 to 99 so that the process with the highest priority is always executed (a higher number indicates a higher priority). The real-time processes can be SCHED_RR (they are scheduled in round-robin with a quanto of 100ms) and SCHED_FIFO (they are scheduled by FCFS). Three real time proceses arrive at a given time in the order A,B,C

En Linux, os procesos en tempo real planifícanse con prioridades estáticas de 1 a 99 de xeito que se execute sempre o proceso con maior prioridade (un número maior indica unha maior prioridade). Os procesos en tempo real poden ser SCHED_RR (en round-robin cun canto de 100ms) e SCHED_FIFO (FCFS). Nun sistema chegan (por esa orde), nun momento dado 3 procesos en tempo real, A, B e C

Proceso	**PRIORIDAD ESTATICA**	**TIPO**	**RAFAGAS CPU-(I/O)-CPU**
A	70	SCHED_RR	500-(800)-300
B	60	SCHED_RR	200-(900)-100
C	60	SCHED_FIFO	300-(400)-100

a) Calcular los tiempos de retorno y de espera de A, B y C./ *Calculate the turaround and waiting times for A, B & C./* Calcula os tempos de retorno e espera para A, B e C.

	Tiempo retorno	Tiempo de espera
Proceso A	***1600***	***0***
Proceso B	***2000***	***800***
Proceso C	***1700***	***900***

b) ¿Cuánto tiempo queda disponible (si es que alguno) para los procesos normales (prioridad estática 0) desde que llegan estos procesos hasta que han terminado los tres?*/How much time (if any) is available for normal processes (static priority 0) from the moment these processes arrive until all three have finished?/* Canto tempo (se hai) queda dispoñible para os procesos normais (prioridade estática 0) desde que chegan estes procesos ata que remataron os tres?

Como se ve en la tabla quedan 500ms para los otros procesos (cada cuadro representa 100ms).

Se ejecuta siempre el de mas prioridad, y si hay dos procesos listos con la misma prioridad se ejecuta el SCHED_RR durante un quanto y el SCHED_FIFO hasta el final de su ráfaga

(puede usarse la siguiente tabla, aunque noes lo que se va a corregir)

cpu	A	A	A	A	A	B	C	C	C	B				A	A	A	C			B		
e/s						A	A	A	A	A C	A C B	A C B	A C B	B	B	B	B	B	B			

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

si lees esto me debes un besito

APELLIDOS:_____ **NOMBRE:**_____ **DNI:** _____

USE LETRAS MAYÚSCULAS EN APELLIDOS Y NOMBRE

Sistemas Operativos - Grado Ingeniera Informática UDC. Enero de 2022

Parte E/S (1.5 puntos).

PREGUNTA 1: (0.5 puntos)

Tenemos un disco que contiene 32768 (=32*1024) sectores. El disco tiene 2 platos y 2 caras en cada plato. Cada pista contiene 8 sectores.

*Temos un disco que contén 32768 (=32*1024) sectores. O disco ten 2 pratos e 2 caras en cada plato. Cada pista contén 8 sectores.*

*We have a disk composed of 32768 (=32*1024) sectors. The disk has 2 platters and there are 2 sides in each platter. Each track contains 8 sectors.*

A ¿Cuántos cilindros hay? Núm cilindros = 1024

¿Cantos cilindros hay? Number of cylinders =

How many cylinders does it contain?

Justifique su respuesta: / *Xustifique a súa resposta: / Explain your answer:*

32768 sectores x 1 pista/8sectores = 4096 pistas
4096 pistas x 1 cilindro / 4 pistas = 1024 cilindros (hay 4 caras)

B ¿Cuántos bloques verá el S.O. si se formatea de modo que cada bloque contenga 16 sectores?

¿Cantos bloques verá o S.O. se se formatea de modo que cada bloque conteña 16 sectores?

Núm bloques = 2048

Number of blocks =

Justifique su respuesta: / *Xustifique a súa resposta: / Explain your answer:*

Trivialmente se ve que hay 32768 sectores y cada bloque contiene 16 sectores → el S.O. verá 32768/16 = 2048 bloques

PREGUNTA 2: (0.5 puntos)

El fichero "file.txt" contiene "0123456789012345\n". ¿Qué se escribe en pantalla en las llamadas a printf?

O ficheiro 'file.txt"contén "0123456789012345\n". Que se escribe en pantalla nas chamadas a printf?

File "file.txt" contains "0123456789012345\n". What is written into the screen in the following calls to printf?

```
/* 1.01 */   int main(){                /*program xc2.c*/
/* 1.02 */     char BUF[6]= {'x','x','x','x','x','\0'};
/* 1.03 */     int fd = open("file.txt",O_RDONLY);
/* 1.04 */     int fd2 = dup(fd);
/* 1.05 */     lseek(fd2, 3, SEEK_SET);
/* 1.06 */     printf("%s\n",BUF);
/* 1.07 */     pread(fd,BUF+1,2,0);
/* 1.08 */     lseek(fd, 1, SEEK_CUR);
/* 1.09 */     printf("%s\n",BUF);
/* 1.10 */     pread(fd,BUF,2,3);
/* 1.11 */     printf("%s\n",BUF);
/* 1.12 */     read(fd,BUF+2,1);
/* 1.13 */     read(fd,BUF+3,2);
/* 1.14 */     printf("%s",BUF); close(fd);
/* 1.15 */   }
```

printf lin.06: =	x	x	x	x	x	
printf lin.09: =	x	0	1	x	x	←
printf lin.11: =	3	4	1	x	x	← **Fill your answers here** **Contesta aquí**
printf lin.14: =	3	4	4	5	6	←

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH

si lees esto me debes un besito

PREGUNTA 3: (0.5 puntos)

El fichero "file.txt" contiene "0123456789\n". El fichero file2.dat no existe.

O ficheiro "file.txt" contén "0123456789\n". O ficheiro "file2.dat" non existe.

File "file.txt" contains "0123456789\n". File "file2.dat" does not exist.

```
/* line.01 */  int main(){              /*program redirxa22.c*/
/* line.02 */     char buf[20];
/* line.03 */     int i=0,ifd,ofd,bk;
/* line.04 */     ifd = open("file.txt",O_RDONLY);
/* line.05 */     ofd = open("file2.dat",O_WRONLY|O_CREAT|O_TRUNC, 0666);
/* line.06 */     bk=dup(STDOUT_FILENO);
/* line.07 */     close(STDOUT_FILENO); close(STDIN_FILENO);
/* line.08 */     dup(ifd); dup(ofd);
/* line.09 */     while (read(STDIN_FILENO,buf+i,1)) i++;
/* line.10 */     write(STDOUT_FILENO,buf,i);
/* line.11 */     close(STDOUT_FILENO);
/* line.12 */     dup(bk);
/* line.13 */     write(STDOUT_FILENO,"--done--",8);
/* line.14 */     close(bk); close(ifd); close(ofd);
/* line.15 */  } //read() devuelve 0 cuando no hay datos que leer
```

Asumiendo que el contenido de la tabla de ficheros abiertos del proceso, tras ejecutar las líneas **line.03** y **line.05**, fuese el que se indica a continuación:

*Asumindo que o contido da táboa de ficheiros abertos do proceso, despois de executar as liñas **line.03** e **line.05**, fose o que se indica a continuación:*

*Assuming the contents of the open-files table of the process, after executing lines **line.03** and **line.05**, are those shown below:*

	estado tras/after line.3
0	teclado-in
1	pantalla-out
2	pantalla-err
3	
4	
5	

	estado tras/after line.5
0	teclado-in
1	pantalla-out
2	pantalla-err
3	file1.dat
4	file2.dat
5	

	estado en/on line.13
0	file1.dat
1	pantalla-out
2	pantalla-err
3	file1.dat
4	file2.dat
5	pantalla-out

A. Complétese el contenido de las entradas 0 a 6 de dicha tabla cuando ejecutamos el write de la línea **line.13**.

*Complétese o contido das entradas 0 á 6 de dita táboa cando executamos o write da liña **line.13**.*

*Fill the contents of the entries from 0 to 6 of the open-files when we are executing the write on **line.13**.*

B. ¿Que se muestra por pantalla? Justifique brevemente su respuesta.

Que se amosa por pantalla? Xustifique brevemente a súa resposta

What is printed into the screen? Justify your answer briefly.

Respuesta */ resposta / answer* ➔ --done--

Justificación */ xustificación / explanation* ➔

Sólo hay 2 write que escriban datos:

- En la línea 10, el write (STDOUT_FILENO) no escribe en pantalla, pues la entrada 1 está redirigida para que se escriban datos en el fichero "file2.dat".
-
- En la línea 13, se ha deshecho la redirección (líneas 11 y 12), y por lo tanto en la línea 13, se escriben 8 bytes en pantalla correspondientes al string "--done--".

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

P1

tam bloque 2Kb

12 dir. dir

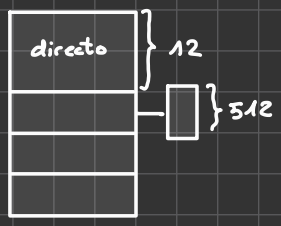
1 ind. simple

1 doble

1 triple

4 bytes dir. bloque

$N^{\circ}\text{punteros} = \frac{\text{tam bloque}}{\text{tam indice (direcc. de bloque)}} = \frac{2 \cdot 2^{10}}{4} = 512 \text{ punteros}$



Dir: $12 \cdot 2 \cdot 2^{10} = 24576 = 24KB$

S: $512 \cdot 2 \cdot 2^{10} = 1048576 = 1MB$

D: $512^2 \cdot 2 \cdot 2^{10} = 536870912 = 0{,}5\,GB = 512MB$

T: $512^3 \cdot 2 \cdot 2^{10} = 256\,GB$

Tmax S = 1MB + 24KB

N° bloques indices (max) = 1

Tmax D = 0,5 GB + 1MB + 24KB

N° bloques indices (max) = 512 + 1 = 513

Tmax T = 256 GB + 0,5 GB + 1MB + 24KB

N° bloques indices (max) = 256 + 512 + 1 = 262657

P2

boot	SB	Lista inodos	Data

0 - 2 3 - 9 (7 bloques)

$N^{\circ}\text{ inodos} = \frac{2 \cdot 2^{10}}{128} = 16 \text{ inodos}$

Tam superbloque = 7 · 2KB = 14KB

Bloque tabla de inodos = $\frac{325}{16} = 21$

30 - 21 = 9 -> 3 - 9

P3

/home /user1 /so /practicas /p1.c -> 32 GB

1° 3° 1° 1° 1°

A Iniciados automáticamente por linux:

0 - stdin

1 - stdout

2 - stderr

fd1 = 3

fd2 = 4

B 1(home) + 3(user) + 1(so) + 1(practicas) + 1(p1.c) = 7

C lseek(fd2, 1000000, SEEK_SET)

↳ bloque

$\frac{1000000}{2\,KB} = 500000$ ~> 2

D

7 5 2

111 101 010

rwx-r-x-w-

E 2, porque "unlink" deshace un slink

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

P4

A - V D - F
B - F E - V
C - F F - V

MEMORIA

1

1 - F 6 - V
2 - F 7 - V
3 - F 8 - V
4 - F 9 - V
5 - F 10 - F

2

a) LRU - Last Recently Used
4 marcos asignados

4 + 1 + 1 + 1 + 1 + 1 + 1 = 10

2	2	2	2	3	3	3	3	1	1
	4	4	4	4	2	2	2	2	3
		6	6	6	6	4	4	4	4
			1	1	1	1	6	6	6
F	F	F	F	F	F	F	F	F	F

b) FIFO 2ª oportunidad

2	2	2	2						
	4	4	4						
		6	6						
			1						

total de fallos = 10

PROCESOS

Q2

A - 12
B - 8

Q4

A

(desde que llega hasta que finaliza su ejecución)

	Retorno	Espera
Proc A	1900	300
Proc B	2100	900
Proc C	900	100

CPU	A	C	C	C	A	A	A	A	C	B	B							A	A	A		B
E/S					C	C	C	C														
									A	A	A	A	A	A	A	A						
												B	B	B	B	B	B	B	B	B		

	Retorno	Espera
Proc A	1600	0
Proc B	2000	800
Proc C	1700	900

CPU	A	A	A	A	A	B	C	C	C	B				A	A	A	C			B
E/S						A	·	·	·	·	·	·	A								
											B	·	·	·	·	·	·	·	B	
										C	·	·	C							

Quedan 500 para procesos normales

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

si lees esto me debes un besito

E/S

① 32768 sectores ; 2 platos ; 2 caras/plato ; 8 sectores/pista

A $\frac{32768}{8} = 4096$ pistas

$$N^{\circ}\,\text{cilindro} = \frac{N^{\circ}\,\text{pistas}}{N^{\circ}\,\text{pistas/cilindro}} = \frac{4096}{4} = 1024$$

N° pistas/cilindro → n° caras · n° platos

B $$N^{\circ}\,\text{bloques que verá el SO} = \frac{N^{\circ}\,\text{sectores}}{N^{\circ}\,\text{sectores/bloque}} = \frac{32768}{16} = 2048$$

②

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

si lees esto me debes un besito