Apellidos:_____ Nombre:_____ DNI: _____

Sistemas Operativos – Grado Ingeniería Informática UDC. Junio 2021
Sólo puede usar lápiz, bolígrafo y calculadora.

Parte **Sistema Ficheros (**2 puntos)

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

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

P1) Un sistema de archivos tipo UNIX System V tiene un tamaño de bloque de 8Kbytes, i-nodos con 12 direcciones directas, una indirecta simple, una indirecta doble y una indirecta triple. Utiliza direcciones de bloque de 8 bytes. Calcular cuántos bloques son necesarios en el área de datos para representar un fichero con un tamaño de 8 Gbytes + 32 Mbytes + 1024 bytes, diferenciando entre bloques de datos y bloques de índices.

Un sistema de ficheiros tipo UNIX System V ten un tamaño de bloque de 8Kbytes, i-nodos con 12 enderezos directos, un indirecto simple, un indirecto dobre e un indirecto triple. Usa enderezos de bloque de 8 bytes. Calcula cantos bloques son necesarios na área de datos para representar un ficheiro cun tamaño de 8 Gbytes + 32 Mbytes + 1024 bytes, diferenciando entre bloques de datos e bloques de índices.

A UNIX file system, System V type, has a block size of 8Kbytes, i-nodes with 12 direct addresses, a single indirect, a double indirect and a triple indirect address. It uses 8-byte block addresses. Calculate how many blocks are needed in the data area to represent a file with a size of 8 Gbytes + 32 Mbytes + 1024 bytes, differentiating between data blocks and index blocks.

Nº bloques de datos / Nº bloques de datos / Number of data blocks: 1 Mbloques + 4Kbloques +1 (0.30p)

Nº de bloques de índices / Nº de bloques de índices / Number of index blocks: 1031 (0.25p)

Fragmentación interna del fichero/Fragmentación interna do ficheiro/ Internal file fragmentation:

7x1024 (bytes) (0.05p)

P2) Un proceso abre 2 veces el archivo anterior (en P1) *"/home/usr1/datos" (tamaño* 8 Gbytes + 32 Mbytes + 1024 bytes*)* en modo lectura, cuyo número de *hard links* es 2. El usuario efectivo del proceso coincide con el propietario del fichero, y tiene además los permisos de acceso a los directorios *home* y *usr1*. Las cachés de datos e inodos están inicialmente vacías y la entrada *usr1* está en el séptimo 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 anterior (en P1) *"/home/usr1/datos" (tamaño* 8 Gbytes + 32 Mbytes + 1024 bytes*)* en modo lectura, cuxo número de *hard links* é 2. O usuario efectivo do proceso coincide co propietario do ficheiro e tamén ten os permisos de acceso os directorios *home* e *usr1*. Os cachés de datos e inodos están inicialmente baleiros e a entrada *usr1* está no séptimo 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 previous file (in P1) *"/home/usr1/datos" (size* 8 Gbytes + 32 Mbytes + 1024 bytes*)* in read mode, whose number of *hard links* is 2. The effective user of the process coincides with the owner of the file, and also has the access permissions to the directories *home* and *usr1*. The data and inode caches are initially empty and the entry *usr1* is in the seventh 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:

```
struct stat buf; char c1, c2;
chmod("/home/usr1 /datos", 0752);
printf("%s", convertir_permisos(0752));  /* se convierten los permisos en octal a formato rwxrwxrwx y se imprimen*/
                                         /* se converten os permisos en octal a formato rwxrwxrwx e se imprimen*/
                                         /* permissions in octal are converted to format rwxrwxrwx and printed*/
link ("/home/usr1/dir1/datos", "/home/usr1 /datos2")
int fd1=open("/home/usr1 /datos", 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/usr1 /datos2", O_RDONLY);
lseek(fd2, 4.000.000, SEEK_SET);/*SEEK_SET indica que el desplazamiento se considera a partir del origen del fichero */
                                /* SEEK_SET indica que o desprazamiento considérase desde a orixe do ficheiro */
                                /* SEEK_SET specifies that the offset is considered from the origin of the file */
c1=fgetc(fd1);  c2=fgetc(fd2);
close(fd1); close(fd2);
unlink("/home/usr1 /datos");
```

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, on the first file opening?: 9

C. Indica el número de bloques que el SO necesita leer en disco para obtener el valor de *c2* en *fgetc(fd2)*:
   Indica o número de bloques que o SO cómpre ler desde o disco para obter o valor de *c2* en *fgetc (fd2)*:
   Indicate the number of blocks that the OS requires to read from disk to get the value of *c2* in *fgetc (fd2)*: 2

D. Indica los permisos del fichero que se imprimen en formato *rwxrwxrwx*:
   Indica os permisos de ficheiro que se imprimen en formato *rwxrwxrwx*:
   Indicate the file permissions that are printed in *rwxrwxrwx* format: rwx r-x –w-

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

P3) Supongamos la siguiente secuencia de comandos:

1. Se crea un *soft link* al fichero "*practica.c*" (cuyo número de inodo es 22342, tamaño 455 bytes y num. de *hard links*=3) en su mismo directorio:

ln -s practica.c slink /* el comando *ls -l* mostraría *slink -> practica.c* */

2. Posteriormente se crea un hard link al fichero *slink*:

ln slink hard_slink /* crea entrada *hard_slink* */

Contestar a lo siguiente:

A. Indicar el tamaño (bytes) del fichero *hard_slink*: 10 (bytes) (0.1p)

B. Indicar cuál es el número de (hard) links del fichero *hard_slink*: 2 (0.1p)

C. Una vez creados los ficheros *slink* y *hard_slink*, al borrar el fichero *practica.c* (*rm practica.c*) se decrementa su número de *hard links*. ¿Se puede acceder al contenido del fichero con número de inodo 22342 a través del link simbólico *slink*? (Sí/No): No (0.05p)

D. Misma pregunta a través del fichero *hard_slink*: (Sí/No) No (0.05p)

Supoñamos a seguinte secuencia de comandos:

1. Se crea un soft link ao ficheiro "*practica.c*" (cuxo número de inodo é 22342, tamaño 455 bytes e num. de *hard links*=3) no seu mismo directorio:

ln -s practica.c slink /* o comando *ls -l* amosaría *slink -> practica.c* */

2. Posteriormente se crea un hard link ao ficheiro *slink*:

ln slink hard_slink /* crea entrada *hard_slink* */

Contestar o seguinte:

A. Indicar o tamaño (bytes) do ficheiro *hard_slink*: _____ (bytes) (0.1p)

B. Indicar cal é o número de (hard) links do ficheiro *hard_slink*: _____ (0.1p)

C. Unha vez creados os ficheiros *slink* e *hard_slink*, ao borrar o ficheiro *practica.c* (*rm practica.c*) se decrementa o seu número de *hard links*. ¿Se pode acceder o contido do ficheiro con número de inodo 22342 a través do link simbólico *slink*? (Sí/Non): _____ (0.05p)

D. Mesma pregunta a través do ficheiro *hard_slink*: (Sí/Non) _____ (0.05p)

Suppose the following sequence of commands:

1. A soft link is created to the file "practica.c" (whose inode number is 22342, size 455 bytes and number of hard links = 3) in the same directory:

ln -s practica.c slink /* the command *ls -l* would show *slink -> practica.c* */

2. Subsequently, a hard link is created to the file *slink*:

ln slink hard_slink /* entry *hard_slink* is created*/

Answer the following:

A. Indicate the size (bytes) of the file *hard_slink*: _____ (bytes) (0.1p)

B. Indicate the number of (hard) links of the file *hard_slink*: _____ (0.1p)

C. Once the files *slink* and *hard_slink* have been created, deleting the file *practica.c* (*rm practica.c*) its number of hard links is decreased. Can the content of the file with inode number 22342 be accessed through the symbolic link *slink*? (Yes/No): _____ (0.05p)

D. Same question through the file *hard_slink*: (Yes/No) _____ (0.05p)

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. El S.O. mantiene una copia en memoria principal del inodo de un fichero abierto, al menos hasta el último cierre del fichero. C

B. Al lincar un código con una librería dinámica, las funciones necesarias de la librería se integran en el fichero ejecutable. F

C. Al ejecutar la función *printf* de C se incrementa tanto el tiempo de ejecución en modo usuario como en modo sistema. C

D. Es posible crear *soft links* entre diferentes sistemas de ficheros montados. C

E. La idea fundamental de un sistema de ficheros Unix basado en registro (*journaling file system*) es llevar control/registro de las asignaciones de los inodos a lo largo del tiempo. F

F. El *Buffer Cache* reduce el número de lecturas o escrituras físicas sobre los discos montados. C

Indique se é verdadeiro/falso en cada pregunta.

Cada apartado puntúa 0.1p. Cada resposta incorrecta puntúa -0.1p. 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. O S.O. mantén una copia en memoria principal do inodo dun ficheiro aberto, polo menos ata o último peche do ficheiro. ____

B. Ao lincar un código cunha librería dinámica, as funcións necesarias da librería se integran no ficheiro executable. ____

C. Ao executar a función *printf* de C se incrementa tanto o tempo de execución en modo usuario como en modo sistema. ____

D. E posible crear *soft links* entre diferentes sistemas de ficheiros montados. ____

E. A idea fundamental dun sistema de ficheiros Unix basado en rexistro (*journaling file system*) é levar un control/rexistro das asignacións dos inodos ao longo do tempo. ____

F. A *Buffer Cache* reduce o número de lecturas ou escrituras físicas sobre os discos montados. ____

Indicate whether it is true/false for each question.

Each question scores 0.1p. Each wrong answer scores -0.1p. 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. The O.S. keeps a copy in main memory of the inode of an open file, at least until the final closing of the file. ____

B. When linking a code with a dynamic library, the necessary functions of the library are integrated into the executable file. ____

C. Executing the C *printf* function increases both the runtime in user mode and in system mode. ____

D. It is possible to create *soft links* between different mounted filesystems. ____

E. The fundamental idea of a Unix *journaling file system* is to keep track/record of inode assignments over time. ____

F. The *Buffer Cache* reduces the number of physical reads or writes on mounted disks. ____

Apellidos:_____Nombre:_____ DNI: _____

Sistemas Operativos - Grado Ingeniera Informática UDC. Julio 2021
Parte Memoria (2 puntos)

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.

- **En la pila del proceso está el código de las funciones que se llaman**: __F___
Na pila do proceso está o código das funcións que se chaman
The process stack contains the code of called functions

-**Las variables locales del main() están en el segmento de datos:** _F____
As variables locais de main() están no segmento de datos
The local variables of main() are in the data segment

-**Para un proceso con varios hilos (threads) de ejecución, los hilos comparten código y datos:** __V__
Para un proceso varios fíos de execución (threads), os fíos comparten código e datos.
For a process consisting of several threads, the threads share code and data.

-**El código de la función de librería malloc usada en un programa C está en el espacio de direcciones del kernel:__F__**
O código da función de librería malloc usado nun programa C está no espacio de direccións do kernel
The code for the malloc library function used in a C program is in the kernel address space

-**Para un sistema con una tabla de páginas de tres niveles (sin caché, ni TLB), traer y ejecutar una instrucción que añade un valor constante a un registro, implica exactamente cuatro accesos a memoria:** _V__
Para un sistema con táboa de páxinas en tres niveis (sin caché, nin TLB), traer e executar unha instrucción que engade un valor constante a un rexistro, implica exactamente catro accesos a memoria.
For a system with 3 levels page table (without caché, without TLB), fetch and execute an instruction with adds a constant value to a register, implies exactly four memory accesses.

-**Para un sistema con tablas de páginas en tres niveles, donde la tabla de páginas raíz no está fija en memoria, y procesador con caché y TLB, traer y ejecutar una instrucción que añade un valor constante a un registro, puede implicar ningún acceso a memoria:** _V__
Para un sistema con táboa de páxinas en tres niveis, donde a táboa de páxinas non está fixa en memoria, e procesador con caché e TLB, traer e executar unha instrucción que engade un valor constante a un rexistro, pode implicar ningún acceso a memoria.
For a system with three-level page table, the root page table is not locked in memory, and processor with caché and TLB, fetching and executing an instruction that adds a constant value to a register, may involve no memory access.

-En un sistema con segmentación que NO TIENE paginación (segmentación pura), es imposible implementar memoria virtual:_F_
En un sistema con segmentación que NON TEN paxinación (segmentación pura), é imposible implementar memoria virtual
For a system with segmentation and no paging (pure segmentation), it is impossible to implement virtual memory

-Si las direcciones físicas son de 32 bits y las páginas de 4Kbytes, el número de página virtual podría venir dado por 33 bits:_V__
Si as direccións físicas son de 32 bits e as páxinas de 4Kbytes, o número de páxina virtual podría vir dado por 33 bits.
With 32 bits physical addresses and 4Kbytes pages, it is possible 33 bits virtual page numbers.

-Con Tabla de Páginas Invertida, hay una tabla de páginas para cada proceso:__F__
Con Táboa de Páxinas Invertida, hay unha táboa de páxinas para cada proceso
With the technique of Inverted Page Table, theres is a page table for each process

-Para un proceso cuyo espacio virtual ocupa N páginas, el algoritmo de reemplazo LRU produce siempre los mismos fallos de página con N marcos asignados al proceso que con N+1:__V__
Para un proceso con un espacio virtual de N páxinas, o algoritmo de reemplazo LRU produce sempre os mesmos fallos de páxina con N marcos asignados ó proceso que con N+1
For a process with a N pages virtual space, the LRU replacement algorithms shows always the same number of page faults with N frames assigned to the process than with N+1 frames assigned

2. (0.4 puntos)
2.v1.**Un proceso tiene la cadena de referencias a páginas que se muestra y tiene asignados tres marcos de memoria. Las cuatro primeras referencias, estos es, la referencias a las páginas 4, 3, 4, 5, producen necesariamente 3 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 FIFO Segunda Oportunidad? Obviamente hay que contar esos 3 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.**
Un proceso ten a cadea de referencias a páxinas que se mostra e ten asignadas tres marcos de memoria. As catro primeiras referencias, isto é, as referencias as páxinas 4, 3, 4, 5 producen necesariamente 3 fallos de páxina porque ningunha páxina do proceso estaba en memoria. ¿Cal é o número total de fallos de páxina que se producen con algoritmo de reemplazo FIFO segunda oportunidade? Obviamente hay que contar esos 3 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 3 page frames is the one shown below. The first four page references, i.e. references to pages 4, 3, 4, 5, produce 3 page faults because none of the pages of this process were in memory. What is the total number of page faults applying the FIFO second chance replacement algorithm? Obviously, you have to include the initial

three 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.
Cadena de referencias. String of page references:
4 3 4 5 1 4 2 1 3 4 1 4 **Respuesta: 8 fallos**

4	3	4	5	1	4	2	1	3	4	1	4
4*	4*	4*	>4*	1*	1*	>1*	>1*	3*	3*	>3*	>3*
>	3*	3*	3*	>3	4*	4*	4*	>4	>4*	4	4*
	>	>	5*	5	>5	2*	2*	2	2	1*	1*
F	F		F	F	F	F		F		F	

2.v2..**Un proceso tiene la cadena de referencias a páginas que se muestra y tiene asignados tres marcos de memoria. Las cuatro primeras referencias, estos es, la referencias a las páginas 4, 3, 4, 5, producen necesariamente 3 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 FIFO Segunda Oportunidad? Obviamente hay que contar esos 3 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.**
Un proceso ten a cadea de referencias a páxinas que se mostra e ten asignadas tres marcos de memoria. As catro primeiras referencias, isto é, as referencias as páxinas 4, 3, 4, 5 producen necesariamente 3 fallos de páxina porque ningunha páxina do proceso estaba en memoria. ¿Cal é o número total de fallos de páxina que se producen con algoritmo de reemplazo FIFO Segunda Oportunidade? Obviamente hay que contar esos 3 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 3 page frames is the one shown below. The first four page references, i.e. references to pages 4, 3, 4, 5, produce 3 page faults because none of the pages of this process were in memory. What is the total number of page faults applying the FIFO Second Chance replacement algorithm? Obviously, you have to include the initial three 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.
Cadena de referencias. String of page references:
4 3 4 5 1 4 2 1 3 4 1 5 **Respuesta: 9 fallos**

4	3	4	5	1	4	2	1	3	4	1	5
4*	4*	4*	>4*	1*	1*	>1*	>1*	3*	3*	>3*	3
>	3*	3*	3*	>3	4*	4*	4*	>4	>4*	4	5
	>	>	5*	5	>5	2*	2*	2	2	1*	>1*
F	F		F	F	F	F		F		F	F

3. (0.6 puntos: 0.2 cada apartado A) B) C))

3.v1. **Un sistema tiene direcciones lógicas de 32 bits y páginas de 4Kbytes. En las direcciones lógicas los bits menos significativos son para el offset en la página. La tabla de páginas es de un nivel con entradas de 2 bytes. Para cada entrada de la TP, los 8 bits menos significativos son para el número de marco, bit 15 (presencia, 1 presente), bit 14 (1 R/W, 0 sólo R), bit 13 (1**

locked), bit 12 (1 acceso modo usuario, 0 acceso sólo modo kernel), bits 8-11 (otros bits de control y no usados). Un proceso tiene una TP en la que los contenidos de las tres primeras entradas son, en hexadecimal:
entrada 0: 0x7F24
entrada 1: 0xFF24
entrada 2: 0xDF14
Indicar en hexadecimal la dirección física que se corresponde para ese proceso con una operación de escritura en memoria para las direcciones lógicas de los casos A, B y C. Si no se puede obtener una dirección física, dar la razón. Tanto los cálculos como el resultado final tienen que ser correctos para conseguir la puntuación

Un sistema ten direccións lóxicas de 32 bits e páxinas de 4Kbytes. Nas direccións lóxicas os bits menos significativos son para o offset na páxina. A táboa de páxinas é de un nivel con entradas de 2 bytes. Para cada entrada da TP, os 8 bits menos significativos son para o número de marco, bit 15 (presencia, 1 presente), bit 14 (1 R/W, 0 só R), bit 13 (1 locked), bit 12 (1 acceso modo usuario, 0 acceso só modo kernel), bits 8-11 (outros bits de control e non usados). Un proceso ten unha TP na que os contidos das tres primeiras entradas son, en hexadecimal:
entrada 0: 0x7F24
entrada 1: 0xFF24
entrada 2: 0xDF14
Indicar en hexadecimal a dirección física que se corresponde para ese proceso con unha operación de escritura en memoria para aas direcciones lóxicas dos casos A, B e C. Si non se pode obter unha dirección física, dar a razón. Tanto os cálculos como o resultado final teñen que ser correctos para conseguir a puntuación

A system has 32 bits logical addresses and 4 Kbytes pages. Least significant bits of the logical addresses are used for the page offset. The system has a one level page table with 2 bytes entries. For each PT entry, the 8 least significant bits are used for the frame number, bit 15 (presence bit, 1 presence), bit 14 (1 R/W, 0 sólo R), bit 13 (1 locked), bit 12 (1 user access, 0 only kernel access), bits 8-11 (other control and not used bits). For a process, the contents of the first PT entries are, in hexadecimal:
entry 0: 0x7F24
entry 1: 0xFF24
entry 2: 0xDF14
Show in hexadecimal, the physical address obtained with a memory write operation for this process for the logical addresses of the cases A, B and C. If a physical address cannot be obtained, state the reason. Both the calculations and final result must be right to get the score for this question

A) Dirección Lógica/Dirección Lóxica/Logical Address: 0x0224
Respuesta/Resposta/Answer: _____página no presente en memoria_____

numero pagína 0. offset 0x224. Entrada TP 0, cuatro primero bits entrada 0111, por tanto bit 15=0, página no presente en memoria.

B) Dirección Lógica/Dirección Lóxica/Logical Address: 0x1224
Respuesta/Resposta/Answer: _____0x24224_____

número de página 1, offset 0x224, Entrada TP 1, cuatro primeros bits 1111, bit 15=1 presente en memoria, bit 14=1 R/W, bit 12=1 acceso modo usuario. Por tanto número frame válido 0x24. Concatenando número de frame y offset: 0x24224

C) Dirección Lógica/Dirección Lóxica/Logical Address: 0x2224

Respuesta/Resposta/Answer: _____0x14224_____

número de página 2, offset 0x224, Entrada TP 2, cuatro primeros bits entrada 1101, bit 15=1 presente en memoria, bit 14=1 R/W, bit 12=1 acceso modo usuario. Por tanto número frame válido 0x14. Concatenando número de frame y offset: 0x14224

3.v2. **Un sistema tiene direcciones lógicas de 32 bits y páginas de 4Kbytes. En las direcciones lógicas los bits menos significativos son para el offset en la página. La tabla de páginas es de un nivel con entradas de 2 bytes. Para cada entrada de la TP, los 8 bits menos significativos son para el número de marco, bit 15 (presencia, 1 presente), bit 14 (1 R/W, 0 sólo R), bit 13 (1 locked), bit 12 (1 acceso modo usuario, 0 acceso sólo modo kernel), bits 8-11 (otros bits de control y no usados). Un proceso tiene una TP en la que los contenidos de las tres primeras entradas son, en hexadecimal:**
entrada 0: 0xFF24
entrada 1: 0xDF14
entrada 2: 0x7F24
Indicar en hexadecimal la dirección física que se corresponde para ese proceso con una operación de escritura en memoria para las direcciones lógicas de los casos A, B y C. Si no se puede obtener una dirección física, dar la razón. Tanto los cálculos como el resultado final tienen que ser correctos para conseguir la puntuación

Un sistema ten direccións lóxicas de 32 bits e páxinas de 4Kbytes. Nas direccións lóxicas os bits menos significativos son para o offset na páxina. A táboa de páxinas é de un nivel con entradas de 2 bytes. Para cada entrada da TP, os 8 bits menos significativos son para o número de marco, bit 15 (presencia, 1 presente), bit 14 (1 R/W, 0 só R), bit 13 (1 locked), bit 12 (1 acceso modo usuario, 0 acceso só modo kernel), bits 8-11 (outros bits de control e non usados). Un proceso ten unha TP na que os contidos das tres primeiras entradas son, en hexadecimal:
entrada 0: 0xFF24
entrada 1: 0xDF14
entrada 2: 0x7F24
Indicar en hexadecimal a dirección física que se corresponde para ese proceso con unha operación de escritura en memoria para aas direcciones lóxicas dos casos A, B e C. Si non se pode obter unha dirección física, dar a razón. Tanto os cálculos como o resultado final teñen que ser correctos para conseguir a puntuación

A system has 32 bits logical addresses and 4 Kbytes pages. Least significant bits of the logical addresses are used for the page offset. The system has a one leve page table with 2 bytes entries. For each PT entry, the 8 least significant bits are used for the frame number, bit 15 (presence bit, 1 presence), bit 14 (1 R/W, 0 sólo R), bit 13 (1 locked), bit 12 (1 user access, 0 only kernel access), bits 8-11 (other control and not used bits). For a process, the contents of the first PT entries are, in hexadecimal:
entry 0: 0xFF24
entry 1: 0xDF14
entry 2: 0x7F24
Show in hexadecimal, the physical address obtained with a memory write operation for this process for the logical addresses of the cases A, B and C. If a physical address cannot be obtained, state the reason. Both the calculations and final result must be right to get the score for this question
A) Dirección Lógica/Dirección Lóxica/Logical Address: 0x0224
Respuesta/Resposta/Answer: _____0x24224_____

numero pagína 0. offset 0x224. Entrada TP 0, cuatro primero bits entrada 1111, por tanto bit 15=1, página presente en memoria, bit 14=1 R/W, bit 12=1 acceso modo usuario. Por tanto número frame válido 0x24, Concatenando número de frame y offset: 0x24224

B) Dirección Lógica/Dirección Lóxica/Logical Address: 0x1224
Respuesta/Resposta/Answer: _____0x14224_____

número de página 1, offset 0x224, Entrada TP 1, cuatro primeros bits 1101, bit 15=1 presente en memoria, bit 14=1 R/W, bit 12=1 acceso modo usuario. Por tanto número frame válido 0x14. Concatenando número de frame y offset: 0x14224

C) Dirección Lógica/Dirección Lóxica/Logical Address: 0x2224

Respuesta/Resposta/Answer: _____página no presente en memoria_____

número de página 2, offset 0x224, Entrada TP 2, cuatro primeros bits entrada 0111, bit 15=0 no presente en memoria,

SISTEMAS OPERATIVOS

Segundo curso. Grado en Informática. JULIO 2021

APELLIDOS (mayúsculas), Nombre:_____

Esta parte **NO** admite entrega hojas adicionales. Hay que contestar en las páginas y espacios proporcionados

1. **(0.8 puntos)** Un sistema monoprocesador con múltiples colas tiene tres colas: SY para procesos del sistema, IT para procesos de usuario interactivos y NI para procesos de usuario no interactivos. La planificación entre las colas es por prioridades apropiativas (un proceso de una cola solo podrá ejecutarse si no hay procesos listos en las colas de mayor prioridad) siendo la prioridad más alta para la cola de procesos del sistema y la más baja para los procesos de usuario no interactivos. Suponemos que el sistema usa Round Robin de *quantum* 3 para los procesos del sistema (cola SY), Round Robin de *quantum* 1 para los interactivos (cola IT) y que los no interactivos se planifican con SJF(cola NI). En el cuadro siguiente se muestran como ha planificado el sistema 6 procesos: A y B (procesos del sistema), C y D interactivos y F y G (no interactivos).

   Un sistema uniprocesador de varias colas ten tres colas: SY para procesos do sistema, TI para procesos de usuario interactivos e NI para procesos de usuario non interactivos. A programación entre colas é por prioridades preventivas (un proceso dunha cola só se pode executar se non hai procesos listos nas colas de maior prioridade) sendo a prioridade máis alta para a cola de procesos do sistema e a máis baixa para procesos de usuario non interactivos. Supoñemos que o sistema usa Round Robin de em quantum 3 para procesos do sistema (cola SY), Round Robin de em quantum 1 para interactivos (cola IT) e que os non interactivos están programados con SJF (fila NI). A seguinte táboa mostra como o sistema planificou 6 procesos: A e B (procesos do sistema), C e D (interactivos) e F e G (non interactivos)

CPU	a	a	a	b	b	c	d	c	d	c	-	b	b	a	a	a	b	b	f	f	g	g	d	c	g	g	g
SY	b	b										a	a		b	b											
IT	c d	c d	c d	c d	c d	d	c	d	c														c				
NI												f g	f g	f g	f g	f g	f g	f g	g	g			g	g			
E/S				a	a	b a	b a	b a	b a	b a d	b a c d	c d	c d	b c d	c d	c d	c d	c d	c d	c d	c d	c d	c d					

   Rellenar el siguiente cuadro con la duración de los procesos (x-(y)-z representa una ráfaga de CPU de duración x, seguida una de e/s de duración y seguida de otra de CPU de duración z) y los intervalos de los instantes de llegada de cada proceso (p.e., si la planificacion es compatible con que un proceso haya llegado en el instante 3, 4, 5, 6 ó 7 consignar el intervalo 3-7). **El cuadro anterior se da como ayuda; puede rellenarse si se quiere aunque no es necesario hacero**. LA E/S DE CADA PROCESO DEBE SER LA MÍNIMA (NÚMERO DE RÁFAGAS Y DURACIÓN) COMPATIBLE CON LA PLANIFICACIÓN.

Enche a seguinte táboa coa duración dos procesos (x-(y)-z representa unha ráfaga de CPU de duración x, seguida dunha E/S de duración y seguida doutra ráfaga de CPU de duración z) e os intervalos do horarios de chegada de cada proceso (por exemplo, se o planificado é compatible coa cheagada dun proceso ao tempo 3, 4, 5, 6 e 7, rexistrese o intervalo 3-7). A táboa anterior é unha axuda, pede encherse ainda **que non é necesario facelo.** *A E/S DE CADA PROCESO DEBE SER O MÍNIMO (NÚMERO DE RÁFAGAS E DURACIÓN) COMPATIBLE COA PLANIFICACIÓN*

	Ráfagas	Tiempo de llegada	Tipo
Proceso A	3-(8)-3	0	sistema (SY)
Proceso B	2-(6)-2-(1)-2	0-3	sistema (SY)
Proceso C	3-(12)-1	0-5	interactivo (IT)
Proceso D	2-(13)-1	0-6	interactive (IT)
Proceso F	2	11-18	no interactivo (NI)
Proceso G	5	11-20	no interactivo (NI)

El primer cuadro muestra la planificación con la mínima e/s posible y poniendo como instante de llegada el primero que es compatible con la planificación. El último instante de llegada válido es el momento en el que obtienen la CPU. Denominamos 0 al instante de comienzo de la planificación y asumimos tambien que cuando dos procesos llegan a la cola de listos en el mismo instante puede entrar primero en la CPU cualquiera de ellos

2. **(0.6 puntos)** Sea el siguiente código en C (con todos los *includes* necesarios y que compila correctamente), que produce un ejecutable llamado `a.out`. (*sleep(n)* deja al proceso en espera durante n segundos)

   Considere o seguinte código en C (con todos os ficheiros include necesarios e compilando correctamente), que produce un executable chamado a.out. (sleep (n) deixa o proceso en espera durante n segundos)

```
#define VECES 10
main(int argc, char * argv[])
{
  int i;
  pid_t pid;

  for (i=0; i<VECES; i++){
      pid=execv("./a.out",NULL,NULL)
      if (pid==-1)
         break;
      printf ("/%ld/",(long) pid);
  }
  sleep (60);
}
```

   ¿Qué salida produce dicho código? (utilizar *xxxx* para indicar un entero distinto de 0)

   `>Que saída produce ese código? (use xxxx para indicar un enteiro distinto de 0)`

   No produce ninguna salida

EXPLICACION

En la primera iteración del bucle, *execv* reemplaza el código (y el resto del espacio de direcciones) del proceso por el del ejecutable *a.out*. Como este es precisamente el código de *a.out*, la ejecución vuleve a comenzar desde el principio y así sucesivamente. Se trata de un bucle infinito que nunca pasa de la primera línea de la primera iteración del bucle. Nunca llega al *printf* y por tanto no produce salida

3. **(0,6 puntos)** Cada una de las preguntas siguientes son de responder V/F (Verdadero/Falso) o no responder. Respuesta correcta: +0.1; Respuesta incorrecta: -0.1. 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. Puntuación mínima para esta pregunta: 0

   USAR LA SIGUIENTE TABLA PARA RESPONDER `UTILIZA A SEGUINTE TÁBOA PARA CONTESTAR`

afirmación	(a)	(b)	(c)	(d)	(e)	(f)
V/F	V	F	F	V	F	F

   (a) Los algoritmos apropiativos desperdician más tiempo en cambios de contexto. *Os algoritmos apropiativos perden máis tempo nos cambios de contexto.* **V**

   (b) Un programa ejecutado por el root, pasa más tiempo en modo kernel que si lo ejecuta un usuario normal. *Un programa executado por o root pasa máis tempo no modo de núcleo que se o executa un usuario normal.* **F**

   (c) En un sistema donde solo hay un proceso listo, bajarle la prioridad hace que tarde más en ejecutarse. *Nun sistema onde só está listo un proceso, reducindo a súa prioridade conséguese que tarde mis en executarse.* **F**

   (d) Un usuario normal, que además no es del grupo root, puede ejecutar un fichero setuid y setgid del root con permisos rwsr-sr-x. *Un usuario normal, que tampouco pertence ao grupo root, pode executar un ficheiro setuid e setgid de root con permisos rwsr-sr-x* **V**

   (e) Los algoritmos con prioridades no apropiativas no tienen inanición (*startvation*). *Os algoritmos con priordades non apropiativas non teñen* starvation. **F**

   (f) Un bucle de llamadas exec puede llenar la tabla de procesos del sistema. *Un lazo de chamadas ao sistema exec pode encher a táboa de procesos do sistema.* **F**

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

Sistemas Operativos - Grado Ingeniera Informática UDC. Xuño de 2021
Parte E/S (1.5 puntos).

1: (0.5 puntos/points)
Respóndase V/F (Verdadero/Falso) o deje en blanco. Respuesta correcta: +0.1; Respuesta incorrecta: -0.1. Puntuación mínima en esta pregunta: 0.
Respóndase V/F (Verdadeiro/Falso) ou deixe en branco. Resposta correcta: +0.1; Resposta incorrecta: -0.1. Puntuación mínima para esta pregunta: 0.
Answer T/F (True/False) or leave as blank. Correct answer: +0.1; Wrong answer: -0.1. Minimum score for this question: 0.

A	B	C	D	E
F	V	F	V	F

← Responde V/F aquí : o/ou responde en blanco either answer T/F here, or leave blank

A- **Considerando la estructura en capas del software de Entrada/Salida, cuando se llama a read para leer 1 único byte de un fichero previamente abierto, el *device driver* debe verificar si el bloque que contiene dicho byte está ya en memoria**: __F___
Considerando a estrutura en capas do software de Entrada/saída, cando se chama a read para ler 1 único byte dun ficheiro previamente aberto, o device driver *debe verificar se o bloque que contén dito byte está xa en memoria.*
Considering the layered structure of the Input/Output Software, when we call read to read 1 unique byte from a file that was previously opened, the *device driver* must verify if the block that contains such byte is already loaded into memory.

B- **Considérese la estructura en capas del software de E/S. La capa denominada software independiente de dispositivo es la encargada de asignar un nuevo bloque a un fichero cuando al escribir 100 nuevos bytes, éstos no caben en el último bloque que tenía asignado:** _V___
Considérese a estructura en capas do software de E/S A capa denominada software independente do dispositivo é a encargada de asignar un novo bloque a un ficheiro cando ao escribir 100 novos bytes nel, estes non caben no último bloque que tiña asignado.
Let us consider the layered structure of the I/O software. The layer named device independent software is in charge of assigning a new block to a given file when we aim at appending 100 additional bytes to it and there is not enough room within the last block that is currently assigned to it.

C- **En un determinado instante, la cola de peticiones de E/S para acceder a cilindros de un disco contenía las peticiones: [25, 100, 360, 77, 124, 122]. Se han atendido las peticiones [100] (en primer lugar) y [77]. El algoritmo utilizado puede ser el SSTF.** __F___
Nun determinado instante, a cola de peticións de E/S para acceder a cilindros dun disco contiña as peticións: [25,100, 360, 77, 124, 122]. Xa se atenderon as peticións [100] (en primeiro lugar) e [77]. O algoritmo utilizado pode ser SSTF.
At a given time, the I/O-requests queue for accessing cylinders within a disk contained the following requests [25,100, 360, 77, 124, 122]. Requests [100] (in first place) and [77] have already been attended. The I/O scheduling algorithm used could have been SSTF.

D- Tenemos un disco que contiene 32768 (=32*1024) sectores. El disco tiene 4 platos y 2 caras en cada plato. Cada sector contiene 512 bytes, y cada pista contiene 16 sectores. Se ha formateado usando bloques de 2048 bytes. Por lo tanto, el número total de cilindros que contiene el disco es 256. ___V____
*Temos un disco que contén 32768 (=32*1024) sectores. O disco ten 4 platos e 2 caras en cada plato. Cada sector contén 512 bytes, e cada pista contén 16 sectores. Formateouse usando bloques de 2048 bytes. Polo tanto, o número de cilindros que contén o disco é 256.*
We have a disk composed of 32768 (=32*1024) sectors. The disk has 4 platters and there are 2 sides in each platter. Each sector contains 512 bytes, and each track contains 16 sectors. The disk was formatted using blocks of 2048 bytes each. Therefore, the total number of cylinders in the disk is 256.
There are 32768 sect * (1track/16sect) = 2048 tracks.
Each cylinder contains 2*4=8 tracks (because there are 4 platters and 2 sides in each of them)
Therefore, there are 2048tracks * (1cil/8tracks) = 256 cil.

E- Tras ejecutar el siguiente código, y sabiendo que no se produjo ningún error, podemos asegurar que las variables *a* y *b* contienen el mismo valor tras la línea 98. ___F____
Nota: Asúmase que file.txt contiene 'zyxwvut', y no fue modificado durante la ejecución de dicho código.

Tras executar o seguinte código, e sabendo que non se produciu ningún erro, podemos asegurar que as variables a *e* b *conteñen o mesmo valor tras a liña 98.*

Nota: Asúmase que file.txt contén 'zyxwvut' e non foi modificado durante a execución de dito código.

We have executed the following code and no errors occurred. Therefore, we can ensure that the variables *a* and *b* must contain the same value after line 98.

Note: Assume that file.txt contains 'zyxwvut' and that it was not modified during the execution of such code.

```
line 09     int main() {
line 10         char a,b;
line 11         int fd = open("file.txt",O_RDONLY);
...             ...
line 98         read(fd,&a,1);   lseek(fd,0,SEEK_SET);   read(fd,&b,1);
```

We can ensure b='z' (because, lseek set the offset at the beginning of the file). However, we do not know what happened during lines 11-98, and we cannot know which is the value assigned to variable a.

2: (0.3 puntos/points)

El fichero "file.txt" contiene "ABCDEFGHIJK\n". ¿Qué imprime printf por pantalla?

O ficheiro "file.txt" contén ABCDEFGHIJK\n". Que imprime printf por pantalla?

The file "file.txt" contains "ABCDEFGHIJK\n". What is printed by printf into the screen?

printf outputs → printf imprime →

A	B	C	D	-	-	4		

<-- Fill your answer here (1 char per cell)
<-- Contesta aquí (pon 1 char en cada celda)

```
1.01:   int  main(int argc, char *argv[])
1.02:   {    errno=0;
1.03:        char BUF[5]={'\0','\0','\0','\0','\0'};
1.04:        int fd = open("file.txt", O_RDONLY);
1.05:        struct aiocb aio;     long leidos;
1.06:        aio.aio_fildes  = fd;
1.07:        aio.aio_buf     = BUF; aio.aio_nbytes  = 4;
1.08:        aio.aio_reqprio = 0;   aio.aio_offset  = 0;
1.09:        aio.aio_sigevent.sigev_notify = SIGEV_NONE;
1.10:        aio_read(&aio);
1.11:
1.12:        while(aio_error(&aio)==EINPROGRESS);
1.13:        leidos = aio_return(&aio);
1.14:        printf("%s--%lu",BUF,leidos);
1.15:   }
```

***Asúmase que no se producen errores durante la ejecución** / *Asúmase que non se producen erros durante a execución* / Assume there were no errors during the execution of the code above.

3: (0.7 puntos/points)

Dado el siguiente programa, y asumiendo que el fichero file1.dat contiene "0123456789\n", y el fichero file2.dat no existe.

Dado o seguinte programa, e asumindo que o ficheiro file1.dat contén "0123456789\n", e o file2.dat non existe.

Given the following program, and that file file1.dat contains "0123456789\n", and file2.dat does not exists yet.

```
lin.01:   #include "includes.h"    // STDIN_FILENO = 0; STDOUT_FILENO = 1;
lin.02:   int main() {
lin.03:     char buf[15];  int i=0,ifd,ofd,bk;
lin.04:     ifd = open("file1.dat",O_RDONLY);
lin.05:     ofd = open("file2.dat",O_WRONLY|O_CREAT, 0666);
lin.06:     bk=dup(STDOUT_FILENO);
lin.07:     close(STDOUT_FILENO);
lin.08:     dup(ifd);
lin.09:     close(STDIN_FILENO);
lin.10:     dup(ofd);
lin.11:     while (read(STDOUT_FILENO,buf+i,1)) i++;
lin.12:     write(STDIN_FILENO,buf,i);
lin.13:     close(STDIN_FILENO);
lin.14:     dup(bk);
lin.15:     write(STDIN_FILENO,"done\n",5);
lin.16:     close(bk); close(ifd); close(ofd);
lin.17:   }
```

***Nótese que *read* devuelve 0 cuando no quedan datos por leer** */ Nótese que* read *devolve 0 cando non quedan datos por ler* / Note that read returns 0 where there are no data to read.

A: Si consideramos que el contenido de la tabla de ficheros abiertos del proceso tras ejecutar las líneas lin.02 y lin.05 es el que se indica a continuación ¿cuál será el contenido de las entradas 0 a 6, tras ejecutar la línea lin.13?

Se consideramos que o contido da táboa de ficheiros abertos do proceso tras executar as liñas lin.02 e lin.05 é o que se indica a continuación, indíquese cal será o contido das entradas 0 a 6 tras executar a liña lin.13.

If we consider that the contents of the opened-files table of the process after executing the lines lin.02 and lin.05 is shown below, which will be the contents of the entries from 0 to 6 after executing lin.13?

Tras/after lin.02

0	**stdin**
1	**stdout**
2	**stderr**
3	
4	
5	
6	

Tras/after lin.05

0	**stdin**
1	**stdout**
2	**stderr**
3	**file1.dat**
4	**file2.dat**
5	
6	

Tras/after lin.13

0	
1	**file1.dat**
2	**stderr**
3	**file1.dat**
4	**file2.dat**
5	**stdout**
6	

B: Indíquese qué sucede al ejecutar la línea lin.15. (Ej. Se escribe XYZT por pantalla/ en el fichero file2.dat, no se escribe nada, se produce un error porque..., etc.) Justifique brevemente su respuesta.

Indíquese que ocorre ao executar a liña lin.15.(Ex. Escríbese XYZT por pantalla/ no ficheiro file2.dat, non se escribe nada, prodúcese un erro porque..., etc.). Xustifíquese brevemente a súa resposta.

Indicate what happens when line lin.15 is executed. (E.g. XYZT is written into the screen/into the file file2.dat, nothing is written, an error occurs because..., others...) Include a brief explanation of your answer.

Line 15 writes done\n **into the screen, because, because after lin 14, entry 0 (STDIN_FILENO) is redirected to the standard output (STDIN_FILENO is closed in lin 13, and bk=5 –the copy of stdout- is duplicated in lin 14).**

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

Sistemas Operativos - Grado Ingeniera Informática UDC. Xuño de 2021

Parte E/S (1.5 puntos).

1: (0.5 puntos/points)

Respóndase V/F (Verdadero/Falso) o deje en blanco. Respuesta correcta: +0.1; Respuesta incorrecta: -0.1. Puntuación mínima en esta pregunta: 0.

Respóndase V/F (Verdadeiro/Falso) ou deixe en branco. Resposta correcta: +0.1; Resposta incorrecta: -0.1. Puntuación mínima para esta pregunta: 0.

Answer T/F (True/False) or leave as blank. Correct answer: +0.1; Wrong answer: -0.1. Minimum score for this question: 0.

A	B	C	D	E
F	F	V	V	V

← Responde V/F aquí : o/ou responde en blanco either answer T/F here, or leave blank

A- **Considerando la estructura en capas del software de Entrada/Salida, cuando se llama a open para abrir un fichero en modo lectura, el *device driver* se encarga de verificar los permisos del fichero**: __F___

*Considerando a estrutura en capas do software de Entrada/saída, cando se chama a **open** para abrir un ficheiro en modo lectura, o* device driver *encárgase de verificar os permisos do ficheiro.*

Considering the layered structure of the Input/Output Software, when we call **open** to open a file in read mode, the *device driver* is in charge of verifying the permissions of the file.

B- **Considérese la estructura en capas del software de E/S. La capa denominada software a nivel de usuario es la encargada de asignar un nuevo bloque a un fichero cuando al escribir 100 nuevos bytes, éstos no caben en el último bloque que tenía asignado:** _F___

Considérese a estructura en capas do software de E/S. A capa denominada software a nivel de usuario é a encargada de asignar un novo bloque a un ficheiro cando ao escribir 100 novos bytes nel, estes non caben no último bloque que tiña asignado.

Let us consider the layered structure of the I/O software. The layer named user level software is in charge of assigning a new block to a given file when we aim at appending 100 additional bytes to it and there is not enough room within the last block that is currently assigned to it.

C- **En un determinado instante, la cola de peticiones de E/S para acceder a cilindros de un disco contenía las peticiones: [26, 124, 126, 100, 160, 77]. Se han atendido las peticiones [100] (en primer lugar) y [77]. El algoritmo utilizado puede ser el SSTF. __V___**

Nun determinado instante, a cola de peticións de E/S para acceder a cilindros dun disco contiña as peticións: [26,124, 126, 100, 160, 77]. Xa se atenderon as peticións [100] (en primeiro lugar) e [77]. O algoritmo utilizado pode ser SSTF.

At a given time, the I/O requests queue for accessing cylinders within a disk contained the following requests [26, 124, 126, 100, 160, 77]. Requests [100] (in first place) and [77] have already been attended. The I/O scheduling algorithm used could have been SSTF.

D- Tenemos un disco que contiene 32768 (=32*1024) sectores. El disco tiene 2 platos y 2 caras en cada plato. Cada sector contiene 512 bytes, y cada pista contiene 32 sectores. Se ha formateado usando bloques de 2048 bytes. Por lo tanto, el número total de cilindros que contiene el disco es 256. ___V____

*Temos un disco que contén 32768 (=32*1024) sectores. O disco ten 2 platos e 2 caras en cada plato. Cada sector contén 512 bytes, e cada pista contén 32 sectores. Formateouse usando bloques de 2048 bytes. Polo tanto, o número de cilindros que contén o disco é 256.*

We have a disk composed of 32768 (=32*1024) sectors. The disk has 2 platters and there are 2 sides in each platter. Each sector contains 512 bytes, and each track contains 32 sectors. The disk was formatted using blocks of 2048 bytes each. Therefore, the total number of cylinders in the disk is 256.

There are 32768 sect * (1track/32sect) = 1024 tracks.
Each cylinder contains 2*2=4 tracks (because there are 2 platters and 2 sides in each of them)
Therefore, there are 1024tracks * (1cil/4tracks) = 256 cil.

E- Tras ejecutar el siguiente código, y sabiendo que no se produjo ningún error, podemos asegurar que las variables *a* y *b* contienen el mismo valor tras la línea 12. ___V____

Nota: Asúmase que file.txt contiene 'zyxwvut', y no fue modificado durante la ejecución de dicho código.
Tras executar o seguinte código, e sabendo que non se produciu ningún erro, podemos asegurar que as variables a *e* b *conteñen o mesmo valor tras a liña 12.*

Nota: Asúmase que file.txt contén 'zyxwvut' e non foi modificado durante a execución de dito código.

We have executed the following code and no errors occurred. Therefore, we can ensure that the variables *a* and *b* must contain the same value after line 12.

Note: Assume that file.txt contains 'zyxwvut' and that it was not modified during the execution of such code.

```
line 09     int main() {
line 10         char a,b;
line 11         int fd = open("file.txt",O_RDONLY);
line 12         read(fd,&a,1);   lseek(fd,0,SEEK_SET);   read(fd,&b,1);
```

Both calls to read() read the 1st byte of the file ➔ a='z' and b='z'

2: (0.3 puntos/points)

El fichero "file.txt" contiene "ABCDEFGHIJK\n". ¿Qué imprime printf por pantalla?

O ficheiro "file.txt" contén ABCDEFGHIJK\n". Que imprime printf por pantalla?

The file "file.txt" contains "ABCDEFGHIJK\n". What is printed by printf into the screen?

printf outputs → / printf imprime →										<-- Fill your answer here (1 char per cell) / <-- Contesta aquí (pon 1 char en cada celda)
	4	-	-	A	B	C	D			

```
1.01:   int  main(int argc, char *argv[])
1.02:   {    errno=0;
1.03:        char BUF[5]={'\0','\0','\0','\0','\0'};
1.04:        int fd = open("file.txt", O_RDONLY);
1.05:        struct aiocb aio;     long leidos;
1.06:        aio.aio_fildes  = fd;
1.07:        aio.aio_buf     = BUF; aio.aio_nbytes  = 4;
1.08:        aio.aio_reqprio = 0;   aio.aio_offset  = 0;
1.09:        aio.aio_sigevent.sigev_notify = SIGEV_NONE;
1.10:        aio_read(&aio);
1.11:
1.12:        while(aio_error(&aio)==EINPROGRESS);
1.13:        leidos = aio_return(&aio);
1.14:        printf("%lu--%s ",leidos, BUF);
1.15:   }
```

***Asúmase que no se producen errores durante la ejecución** / *Asúmase que non se producen erros durante a execución* / Assume there were no errors during the execution of the code above.

3: (0.7 puntos/points)

Dado el siguiente programa, y asumiendo que el fichero fich1.dat contiene "ABCDEFGHIJK\n", y el fichero fich2.dat no existe.

Dado o seguinte programa, e asumindo que o ficheiro fich.1.dat contén "ABCDEFGHIJK\n", e o fich2.dat non existe.

Given the following program, and that file fich1.dat contains "ABCDEFGHIJK\n", and fich2.dat does not exists yet.

```
          #include "includes.h"
lin.01:   int main() {
lin.02:     char buf[15];  int i=0,ifd,ofd,bk;
lin.03:     ifd = open("fich1.dat",O_RDONLY);
lin.04:     ofd = open("fich2.dat",O_WRONLY|O_CREAT, 0666);
lin.05:     bk=dup(STDOUT_FILENO);
lin.06:     close(STDOUT_FILENO);
lin.07:     close(STDIN_FILENO);
lin.08:     dup(ofd);
lin.09:     dup(ifd);
lin.10:     while (read(STDOUT_FILENO,buf+i,1)) i++;
lin.11:     write(STDIN_FILENO,buf,i);
lin.12:     close(STDIN_FILENO);
lin.13:     dup(bk);
lin.14:     write(STDIN_FILENO,"done\n",5);
lin.15:     close(bk); close(ifd); close(ofd);
lin.16:   } // nota-note:    STDIN_FILENO = 0; STDOUT_FILENO = 1;
```

***Nótese que *read* devuelve 0 cuando no quedan datos por leer** */ Nótese que* read *devolve 0 cando non quedan datos por ler* / Note that read returns 0 where there are no data to read.

A: Si consideramos que el contenido de la tabla de ficheros abiertos del proceso tras ejecutar las líneas lin.01 y lin.04 es el que se indica a continuación ¿cuál será el contenido de las entradas 0 a 6, tras ejecutar la línea lin.12?

Se consideramos que o contido da táboa de ficheiros abertos do proceso tras executar as liñas lin.01 e lin.04 é o que se indica a continuación, indíquese cal será o contido das entradas 0 a 6 tras executar a liña lin.12.

If we consider that the contents of the opened-files table of the process after executing the lines lin.01 and lin.04 is shown below, which will be the contents of the entries from 0 to 6 after executing lin.12?

Tras/after lin.01

0	**stdin**
1	**stdout**
2	**stderr**
3	
4	
5	
6	

Tras/after lin.04

0	**stdin**
1	**stdout**
2	**stderr**
3	**fich1.dat**
4	**fich2.dat**
5	
6	

Tras/after lin.12

0	
1	**fich1.dat**
2	**stderr**
3	**fich1.dat**
4	**fich2.dat**
5	**stdout**
6	

B: Indíquese qué sucede al ejecutar la línea lin.14. (Ej. Se escribe XYZT por pantalla/ en el fichero fich2.dat; no se escribe nada, se produce un error porque..., etc.) Justifique brevemente su respuesta.

Indíquese que ocorre ao executar a liña lin.14.(Ex. Escríbese XYZT por pantalla / no ficheiro fich2.dat; non se escribe nada, prodúcese un erro porque..., etc.). Xustifíquese brevemente a súa resposta.

Indicate what happens when line lin.14 is executed. (E.g. XYZT is written into the screen/ into the file fich2.dat; nothing is written, an error occurs because..., others...) Include a brief explanation of your answer.

Line 14 writes done\n **into the screen, because, because after lin 13, entry 0 (STDIN_FILENO) is redirected to the standard output (STDIN_FILENO is closed in lin 12, and bk=5 –the copy of stdout- is duplicated in lin 13).**