

Nombre:

DNI:

Segundo control de teoría

Contesta todas las preguntas justificando brevemente tu respuesta. Una respuesta sin justificar se considerará como incorrecta.

1. (2 puntos) Preguntas cortas

- a. En un sistema de ficheros con bloques de 4KBs, ¿cuál es el tamaño máximo que un fichero puede tener para asegurar que una asignación encadenada realiza el mismo número de accesos a disco que una asignación indexada multinivel recorriendo secuencialmente el fichero para leerlo entero?

- b. ¿Es cierto que un sistema operativo sin gestores no tiene el estado de blocked de threads dentro de su ciclo de vida?

- c. Calcula qué tamaño debería tener un bloque de disco en un sistema de ficheros con asignación encadenada en tabla y un tamaño de índice de bloque de 4 bytes, para asegurar que la FAT siempre ocupa el 50% de la partición.

- d. El I-nodo solamente es necesario para guardar el número de referencias a un fichero en un sistema de ficheros con directorios en grafo. Si este número de referencias lo pusiesemos en el superbloque, ¿podríamos mantener la información estructurada como en un sistema de ficheros con directorios en árbol prescindiendo de los I-nodos?

SOA2 (21/12/2022)

Nombre:

DNI:

2. (4 puntos) Sistema de ficheros:

Queremos modificar la implementación de la asignación indexada de bloques a ficheros para asignar conjuntos de bloques consecutivos de disco. Para ello, en vez de tener índices a bloques de disco, tendremos, en los bloques de índices, parejas de <primer bloque de disco, número de bloques consecutivos>. Esta implementación la probaremos en un disco que tiene 2^{20} bloques de disco de 4Kbs cada uno. Los índices a bloques de disco son de 32 bits y el número de bloques consecutivos también se guardará con un entero de 32 bits. Consideraremos 2 escenarios:

-Peor escenario: no hay más de un bloque de disco consecutivo libre.

-Mejor escenario: siempre se encuentran el número de bloques de disco consecutivos necesarios.

No se tienen que tener en cuenta ni el espacio asignado a superbloque ni el asignado a I-nodes.

- a) Calcula el número de índices a bloques de disco (sin tener en cuenta la modificación planteada) que caben en un bloque de disco con una asignación indexada normal

- b) Calcula el número de parejas <bloque de datos, número de bloques consecutivos> que caben en la nueva implementación por bloque de disco.

- c) Calcula el número de bloques de disco referenciados por bloque de índices en el peor y mejor escenarios.

- d) Justifica, para el peor escenario, si esta nueva implementación mejora el acceso secuencial a ficheros con respecto a la asignación indexada normal.

SOA2 (21/12/2022)

Nombre:

DNI:

- e) Justifica, para el mejor escenario, si esta nueva implementación mejora el acceso secuencial a ficheros con respecto a la asignación indexada normal.

- f) Justifica, para el peor escenario, si esta nueva implementación mejora el acceso aleatorio a ficheros con respecto a la asignación indexada normal.

- g) Justifica, para el mejor escenario, si esta nueva implementación mejora el acceso aleatorio a ficheros con respecto a la asignación indexada normal.

- h) Si has calculado bien el apartado b, te habrás dado cuenta que en un bloque de índices sobran 4 bytes. Si decidimos utilizar estos 4 bytes para guardar el número de bloques referenciados en total, dentro de ese bloque de índices, ¿conseguimos mejorar el acceso aleatorio a ficheros?

- i) Sin tener en cuenta el acceso a ficheros, esta implementación mejora en alguna otra característica la asignación indexada normal?

- j) Que es lo que se tiene que producir para que esta nueva implementación y la implementación normal tengan el mismo rendimiento en los accesos a disco?

SOA2 (21/12/2022)

Nombre:

DNI:

- k) Después de todo este análisis, recomendarías esta nueva implementación de la asignación indexada?

Decidimos hacer la misma modificación (la que propone el enunciado, no la del apartado h) para la asignación indexada multinivel.

- l) Justifica si esta nueva asignación indexada multinivel mejora, para el peor escenario, el acceso secuencial a ficheros.

- m) Justifica si esta nueva asignación indexada multinivel mejora, para el mejor escenario, el acceso secuencial a ficheros

- n) Justifica si esta nueva asignación indexada multinivel mejora, para el peor escenario, el acceso aleatorio a ficheros

- o) Justifica si esta nueva asignación indexada multinivel mejora, para el mejor escenario, el acceso aleatorio a ficheros

- p) En el caso de la asignación indexada multinivel, ¿siguen sobrando 4 bytes por bloque de índices?

SOA2 (21/12/2022)

Nombre: _____

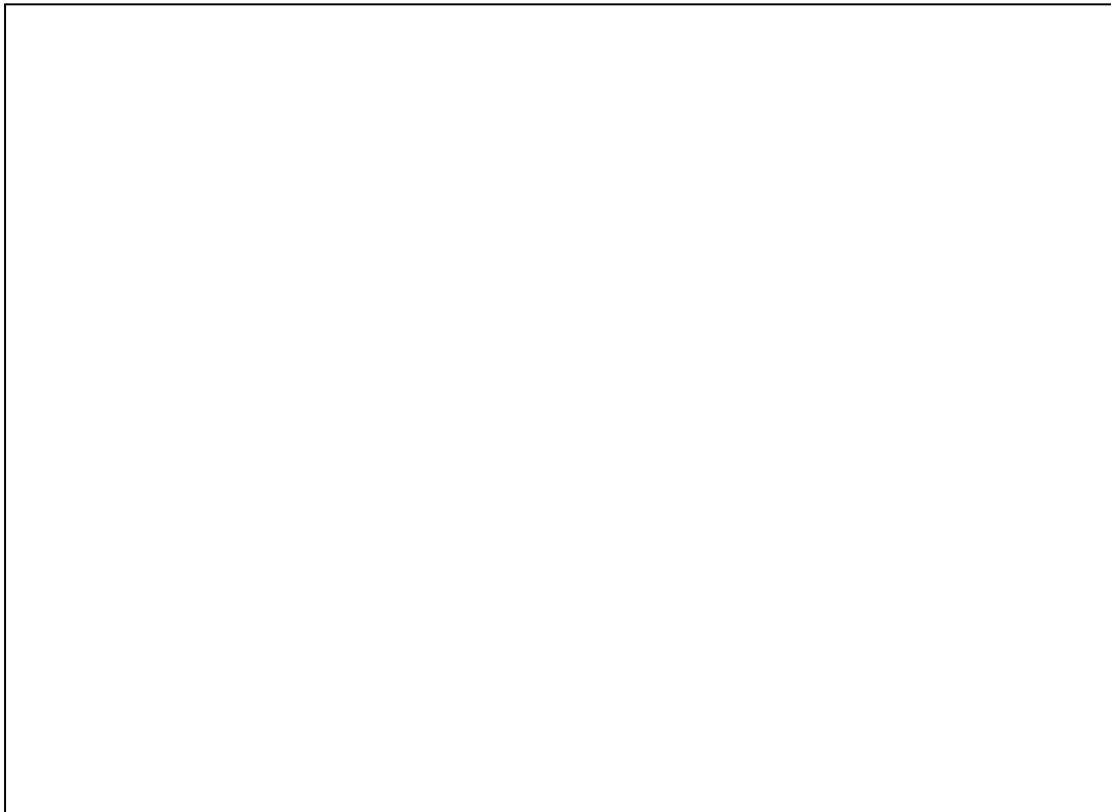
DNI: _____

3. (4 puntos) /dev/null

Queremos implementar el dispositivo lógico /dev/null. Este dispositivo lógico es un “agujero negro”: todo lo que se escribe en él, no se almacena en el sistema de ficheros sino que desaparece. Las lecturas a /dev/null siempre devuelven final de fichero (es decir, read devuelve 0). Las escrituras siempre devuelven el número de bytes escritos. Este dispositivo tiene un gestor asociado al cual solamente puede acceder el gestor de VFS.

Para contestar las preguntas, has de tener en cuenta que los nombres de las funciones dependientes del gestor de VFS siempre tienen el sufijo _vfs, que las del gestor de /dev/null tienen el sufijo _null y que las del driver de /dev/null tienen el sufijo _driver

- a) Haz un esquema donde se vea como se relaciona, la llamada al sistema read, el gestor de vfs, el de /dev/null y el del driver junto con las estructuras y/o código que se necesita



- b) Enumera las estructuras del sistema de ficheros, junto con sus campos, a las que accede sys_read



SOA2 (21/12/2022)

Nombre:

DNI:

c) Describe los campos que debería tener un iorb en una petición al VFS de lectura

d) ¿Cómo encuentra sys_read la dirección de la función "read_vfs"?

e) Escribe la sucesión de llamadas para hacer un read a /dev/null.

f) Escribe el código del write dependiente del gestor VFS.

```
int write_vfs(int fd, int offset, void *buffer, size_t size)
{

}

}
```

g) Escribe el código del write dependiente del gestor de /dev/null

```
int write_null(struct Inode* Inode, int offset, void *buffer, size_t size)
{

}

}
```

SOA2 (21/12/2022)

Nombre:

DNI:

h) Escribe el código del write del driver de /dev/null

```
int write_driver(struct opaque* data, int offset, void *buffer, size_t size)
{

}

}
```

i) Escribe el código del gestor de /dev/null

j) Dado el funcionamiento de este dispositivo, ¿sería necesario tener un gestor? En caso negativo indica qué funciones de las anteriores se tendrían que modificar y cómo se modificarían.