RECUPERATORIO 19/06/2024

GRILLA DE RESPUESTAS A PREGUNTAS

	1	2	3	4	5	6	7	8	9	10
A	X				X					
B			X			X	X			
C		X		X						
D										

Ejercicio 8

`ls /dir/x`
Inodos: 2 accesos (3 también se toma como bien)
Datos: 2 accesos

`cat /dir/s/y`
Inodos: 4 accesos
Datos: 4 accesos

`cat /dir/h`
Inodos: 3 accesos
Datos: 3 accesos

`cat /dir/y`
Inodos: 7 accesos
Datos: 7 accesos

Ejercicio 9

Cantidad de paginas: 51

Ejercicio 10

1 superbloque
1 bitmap de inodos
1 bitmap de datos
129 bloques de inodos
1917 bloques de datos (2049 - 129 - 1 - 1 - 1)

Desarrollo

Ejercicio 8

`ls /dir/x`

1. Acceso a inodo '/
2. Leer el bloque de datos con el contenido de /, encuentra la entrada para ‘dir’
3. Acceso al inodo dir
4. Leer el bloque de datos con el contenido de dir, encuentra la entrada para ‘x’

No accede a los datos del archivo.

Inodos: 2 accesos (1, 3)
Datos: 2 accesos (2, 4)

Nota: Sin parámetros ls no muestra metadata del archivo. Como esto no estaba especificado en el enunciado, consideramos bien 3 accesos a inodos en este ejercicio si el alumno asume que se lee la metadata

`cat /dir/s/y`

1. Acceso a inodo /
2. Leer el bloque de datos con el contenido de /, encuentra la entrada para ‘dir’
3. Acceso al inodo ‘dir’
4. Leer el bloque de datos con el contenido de ‘dir’, encuentra la entrada para ‘s’
5. Acceso al inodo ‘s’
6. Leer el bloque de datos con el contenido de ‘s’, encuentra la entrada para ‘y’
7. Acceso al inodo ‘y’
8. Leer el contenido de ‘y’ y mostrarlo en pantalla

Inodos: 4 accesos (1, 3, 5, 7)
Datos: 4 accesos (2, 4, 6, 8)

`cat /dir/h`

/dir/h se creo como un hard link. Esto significa que si el archivo original se elimina, el hard link sigue funcionando normalmente.

1. Acceso a inodo /
2. Leer el bloque de datos con el contenido de /, encuentra la entrada para ‘dir’
3. Acceso al inodo ‘dir’
4. Leer el bloque de datos con el contenido de ‘dir’, encuentra la entrada para ‘h’
5. Acceso al inodo ‘h’

6. Leer el contenido de ‘h’ y mostrarlo en pantalla

Inodos: 3 accesos (1, 3, 5)
Datos: 3 accesos (2, 4, 6)

```
cat /dir/y
```

/dir/y es un soft link. Esto significa que es simplemente un archivo que contiene el path del archivo asociado. Esto implica que el sistema operativo tiene que leer el path del archivo, y el path contenido en el soft link, uno después del otro.

1. Acceso a inodo /
2. Leer el bloque de datos con el contenido de /, encuentra la entrada para ‘dir’
3. Acceso al inodo ‘dir’
4. Leer el bloque de datos con el contenido de ‘dir’, encuentra la entrada para ‘y’
5. Acceso al inodo ‘y’
6. Leer el contenido de ‘y’. Obtiene el path original: /dir/s/y
7. Acceso a inodo /
8. Leer el bloque de datos con el contenido de /, encuentra la entrada para ‘dir’
9. Acceso al inodo ‘dir’
10. Leer el bloque de datos con el contenido de ‘dir’, encuentra la entrada para ‘s’
11. Acceso al inodo ‘s’
12. Leer el bloque de datos con el contenido de ‘s’, encuentra la entrada para ‘y’
13. Acceso al inodo ‘y’
14. Leer el contenido de ‘y’ y mostrarlo en pantalla

Inodos: 7 accesos (1, 3, 5, 7, 9, 11, 13)
Datos: 7 accesos (2, 4, 6, 8, 10, 12, 14)

Ejercicio 9

Cantidad total de bytes que ocupa el array: 50,000 x 4 = 200,000

Posición del primer byte: 0x01FBD000 (definido en el enunciado)

Posición del último byte: 0x01FBD000 + (50000 x 4 - 1) = 0x01FEDD3F

Componentes de la dirección virtual del primer y último byte utilizados (separadas en grupos de 10, 10 y 12 según la paginación de x86):

- `0x01FBD000 = 0000000111 1110111101 000000000000`
- `0x01FEDD3F = 0000000111 1111101101 110100111111`

Los primeros dos grupos de 10 bits indican índices dentro de las tablas de páginas. Estos índices en decimal son:

Indice del page directory:

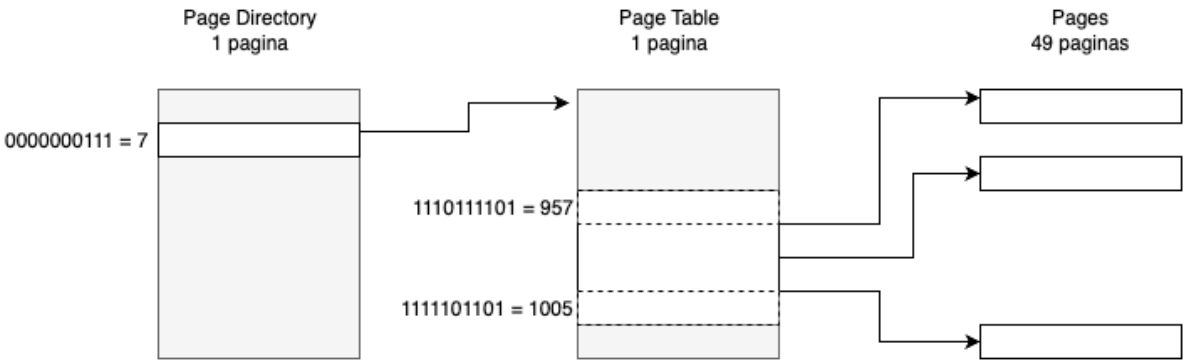
- `0000000111 = 7`

En el primer nivel (el directorio de páginas), se usa un solo registro, en el índice 7. Este contiene la dirección (y metadata) de la **única tabla de páginas** que será necesaria en el segundo nivel. Si se requirieran más tablas de páginas de segundo nivel, este índice cambiaría para algunas direcciones involucradas en el rango. Esto no ocurre para el rango propuesto en este ejercicio.

Índices de las page table

- `1110111101 = 957`
- `1111101101 = 1005`

En el segundo nivel, se utilizan varios registros, cada uno contiene la dirección de las páginas donde están los datos del array. Concretamente, se usan los registros desde el 957 hasta el 1005, es decir, 49 registros, lo que implica 49 páginas de datos.



En el tercer nivel, cada página contiene 4096 bytes de datos. Si queremos guardar 200,000 bytes entonces necesitamos 48.8 páginas, es decir, necesitamos utilizar 49 páginas. Esto es consistente con el cálculo que hicimos antes a partir de las direcciones de los extremos del rango de bytes del array.

Resultado: 1 page directory (por definición de la arquitectura x86) + 1 page table (justificado anteriormente) + 49 páginas de datos (justificado anteriormente) = **51 páginas en total** que el sistema operativo necesita utilizar para guardar y acceder a este array.