Guía de ejercicios.

Sistemas Operativos.

October 20, 2024

Abstract

Indicaciones para realizar la Guía:
★ => requerimiento de tipo parcial.
◆ => requerimiento de tipo opcional. (Se recomienda realizarlos para obtener una mejor comprensión de los temas)

1 File System.

1.1 Montaje de un sistema de archivos.

Sea un disco que posee 512 bloques de 4kb y un sistema operativo cuyos inodos son de 256 bytes, definir un sistema de archivos FFS.

★ Justifique en base a que, se toman todas las decisiones para el armado del FFS.

1.2 Capacidad máxima de un inodo.

Suponiendo que BigFS es una variante de FSS, el clásico sistema de archivos de unix, el cual posee 12 referencias inodos directas, 1 indirecta, 1 doble indirecta, una triple indirecta y una cuádruple indirecta. Asumiendo bloques de 4 Kb, 8 byte por puntero a bloques.

★ ¿Cuál es el máximo tamaño de archivo que se soporta? Expresar en Terabytes.
◆ Dibujar la distribución de los punteros y los bloques de punteros.

1.3 Accesos a inodos y datablocks.

Partiendo de un filesystem vacío se ejecutan una serie de comandos en la shell en el órden indicado.

Para los comandos en **negrita**, indicar la cantidad de accesos a inodos y accesos a bloques de datos que el sistema operativo debe realizar.

Notas:

- Asuma que no existe ningún mecanismo de caché ú optimización, y el sistema operativo siempre tiene que acceder a todos los inodos y bloques de datos necesarios en cada comando.

```
# mkdir /dir    /dir/s
# echo 'hola' > /dir/x
# echo 'mundo' > /dir/s/y
#
# ls /dir/x     Inodos: ---    Blq. datos: ---
# cat /dir/s/y      Inodos: ---    Blq. datos: ---
#
# touch /dir/h /dir/y
# ln /dir/x /dir/h
# ln -s /dir/s/y /dir/y
# rm /dir/x
```

```
#
# cat /dir/h    Inodos: ---    Blq. datos: ---
# cat /dir/y    Inodos: ---    Blq. datos: ---
```

★ Justifique los pasos que el sistema operativo necesita hacer para completar cada comando, y que resulten en la cantidad de accesos que previamente ha indicado.

◆ Dibuje como queda el file system en cada paso.

1.4 Idem anterior (variante de parcial).

a. El superbloque de un sistema de archivos indica que el (3) inodo correspondiente al directorio raíz es el #43. En la siguiente secuencia de comandos, y siempre partiendo de ese directorio raíz, se pide indicar la cantidad de inodos y bloques de datos a los que se precisa acceder (leer) para resolver la ruta dada a `cat(1)` o `stat(1)`.

```
# mkdir /dir    /dir/s    /dir/s/w
# touch /dir/x /dir/s/y
#
# stat /dir/s/w/x     Inodos: ---    Blq. datos: ---
# stat /dir/s/y       Inodos: ---    Blq. datos: ---
#
# touch /dir/y
# ln /dir/s/x /dir/h
# ln -s /dir/s/y /dir/y
#
# cat /dir/h    Inodos: ---    Blq. datos: ---
# cat /dir/y    Inodos: ---    Blq. datos: ---
```

Ayuda: Todos los directorios ocupan un bloque. La idea es que describan como `stat` llega a los archivos.

★ Justifique los pasos que el sistema operativo necesita hacer para completar cada comando, y que resulten en la cantidad de accesos que previamente ha indicado.

◆ Dibuje como queda el file system en cada caso.

2 Memoria Virtual

2.1 Cantidad de direcciones.

Explicar el mecanismo de address translation memoria virtual paginada de tres niveles de indirección de 32 bits. Indique la cantidad de direcciones de memoria que provee, una virtual address:

7 bits	7 bits	6 bits	12 bits

◆ Dibuje la distribución de las páginas y como se conectan:

- Page Directory.
- Page Table #1.
- Page Table #2.
- Page Frame.

2.2 Capacidad de memoria en Kbytes.

Cual es la cantidad de Kbytes que se pueden almacenar en un esquema de memoria virtual de 48 bits con 4 niveles de indirección, en la cuál una dirección de memoria se describe como sigue: 9 bits para el directorio de página, 9 bits para cada tabla de páginas y 12 bits para el offset.

★ Desarrolle las cuentas que se deben realizar para encontrar la cantidad de Kbytes que se pueden almacenar.

◆ ¿Cuánta memoria es en Gbytes? ¿Y en Tbytes?.

2.3 Traducción de direcciones.

Suponga a las virtuales address con las siguientes características:

- 4 bit para el segment number.
- 12 bit para el page number.
- 16 bit para el offset.

Segment table	Page Table A	Page Table B
0 Page B	0 ZASPEH	0 F000
1 Page A	1 DEAD	1 D8BF
X Invalid	2 BEEF	2 3333
	3 BA11	X Invalid

Traducir las siguientes direcciones virtuales a físicas: [0000000], [20022002], [10222002], [00015555]

★ Desarrolle como se realizan las traducciones y cual número de cada dirección indica las distintas tablas.

◆ Dibujar como se acceden a las distintas tablas y páginas.

2.4 Accesos a memoria.

Considere un sistema x86 de memoria virtual paginada de dos niveles con un espacio de direcciones de 32 bits, donde cada página tiene un tamaño de 4096 bytes. Un entero ocupa 4 bytes y se tiene un array de 50,000 enteros que comienza en la dirección virtual 0x01FBD000.

El arreglo se recorre completamente, accediendo a cada elemento una vez. En este proceso, **¿a cuántas páginas distintas** (no la cantidad total de accesos) necesita acceder el sistema operativo para conseguir esto? Recuerde contar las tablas de páginas intermedias, no solo las páginas que contienen los elementos del array.

★ Desarrolle como se realizan los accesos y cuantos enteros se ocupan por página.

◆ dibuje como se realizan los accesos a las distintas páginas. (Realizar un dibujo conceptual que muestre como se usa la memoria).

2.5 Idem anterior (variante de parcial).

Dado un espacio de direcciones virtuales con direcciones de 8 bits y páginas de 16 bytes, asume un array de 12 enteros (cada uno de 4 bytes) comenzando en la dirección virtual 100. Calcula el patrón de aciertos y fallos en la TLB cuando se accede a todos los elementos del array en un bucle. Asume que inicialmente, la TLB está vacía.

Pasos:

1. Determina el VPN y el desplazamiento para cada elemento del array.
2. Identifica si ocurre un acierto o un fallo en la TLB para cada acceso.

★ Desarrolle como se realizan los accesos y cuantos enteros se ocupan por página.

◆ dibuje como se realizan los accesos a las distintas páginas. (Realizar un dibujo conceptual que muestre como se usa la memoria).

2.6 EJERCICIO OPCIONAL.

◆ Dibuje como se distribuye la memoria en las distintas formas de implementar el Virtual Address Space, minimamente:

1. Dynamic Reallocation.
2. Tabla de segmentos.
3. Memoria Paginada.

3 Scheduling

3.1 Ejecución de un proceso MLFQ.

Sea 1 proceso, cuyo tiempo de ejecución total es de 40 ms, el *time slice* por cola es de 2 ms/c pero el mismo se incrementa en 5 ms por cola.

★ ¿Cuántas veces se interrumpe y en qué cola termina su ejecución?

◆ Dibuje las colas detalladamente, mostrando los pasos y cómo el proceso va perdiendo prioridad mientras se ejecuta.

4 Código.

4.1 Concurrencia / Threads.

Utilice la API de los Threads para crear un programa que use 5 threads para incrementar una variable compartida por todos en 7 unidades/thread (cada thread suma 7).

★ La suma debe cortar al llegar o superar las 1000 unidades.

◆ Dar ejemplos de: race-condition; dead-lock.

4.2 Shell primitiva.

Escriba un programa en C que simule el funcionamiento de una shell primitiva.

★ ¿Qué debería agregarse para poder encadenar dos comandos por las salidas y las entradas?

◆ Programar la shell junto con todas las validaciones necesarias y comentar el código explicando que sucede en cada paso.

4.3 Ping Pong.

Escriba un programa en C que permita jugar a dos procesos al ping pong, la pelota es un entero, cada vez que un proceso recibe la pelota debe incrementar en 1 su valor.

★ El programa debe cortar por overflow o cambio de signo.

4.4 EJERCICIOS OPCIONALES.

Escriba las siguientes funciones en el lenguaje de programación C:

1. Implementar la función LS. (./a.out)
2. Implementar la función Copy. (./a.out fileold filenew)
3. Implementar la función Find (./a.out /path/to/search "filename).