PRACTICA # 6

CREANDO Y USANDO EL SISTEMA DE ARCHIVOS

OBJETIVO: Utilizar los comandos básicos para la creación y uso del sistema de archivos de Linux.

INTRODUCCIÓN:

La parte más importante de cualquier sistema de cómputo son los datos, es decir, la información que los programas almacenan y manipulan. Linux tiene un sistema de archivos cuyo trabajo es conservar toda la información que se almacene en la computadora, incluyendo programas, documentos, bases de datos, textos, etcétera.

En Linux el término "archivo" se refiere a cualquier fuente de entrada o destino de salida, no sólo a un depósito de datos. El espacio visible a los usuarios se basa en una estructura de árbol, con la raíz en lo alto. Los archivos y los directorios se colocan debajo de la raíz.

En realidad, algunos de los directorios en el árbol de archivos están físicamente ubicados en diferentes particiones del disco, sobre diferentes discos y eventualmente en diferentes computadoras. Cuando una de estas particiones del disco está ligado con el árbol de archivos en el directorio conocido como punto de montaje (*mount point*); al punto de montaje y a todos los directorios inferiores nos referiremos como el **SISTEMA DE ARCHIVOS.**

Linux tiene siete tipos de archivos y hablaremos de 3: los ordinarios, los directorios y los especiales. Los archivos ordinarios contienen datos y se almacenan en un disco. Un directorio se almacena en disco y contiene información que se usa para organizar y permitir el acceso a otros archivos.

Los archivos especiales o de dispositivo, son la representación interna de un dispositivo físico.

A. Para poder montar el sistema de archivos en Linux, primero debemos tener una partición física en el disco, un CD-ROM, o un floppy.

Linux usa el comando '`mount`' para montar el sistema de archivos. Las formas más utilizadas son:

```
/bin/mount [opciones] dispositivo  punto-de-montaje
/bin/mount punto-de-montaje
/bin/mount -at nfs
```

Donde *opciones* son las banderas que soporta el comando mount, *dispositivo* es el nombre del archivo de dispositivos (device file) de modo bloque que se desea montar y *punto-de-montaje* es el directorio donde se montará el sistema de archivos. Cabe mencionar que el *punto-de-montaje* es un sub-directorio que debe estar vacío, si éste sub-directorio tiene alguna información; ésta quedará oculta cuando se monte el sistema de archivos. El comando tiene varias banderas, estos son:

-f	Termina todos los procesos excepto la actual llamada al sistema.
-v	`mount` provee información adicional sobre lo que se trata de hacer.
-w	El sistema de archivos se montará con permisos de lectura y escritura.
-r	El sistema se montará con permisos de solo lectura.
-n	Se monta sin escribir las entradas en el archivo `/etc/mtab`
-t tipo	Especifica el tipo de sistema de archivos a instalar. Los tipos validos son: `minix,ext,ext2,xiafs,msdos,hpfs,proc,nfs, umsdos,sysv e iso9660.`
-*a*	Intenta montar todo el sistema de archivos, descritos en `/etc/fstab`
-o lista de opciones	El argumento `-o` le dice a `mount` que aplique las opciones que son especificadas para el sistema de archivos que se montará. Para más detalles ver la página de ayuda con el comando '`man mount`'

Podemos hacer que este tipo de montajes se realicen automáticamente cuando el sistema entra a modo multiusuario, editando el archivo de configuración llamado: `/etc/fstab`. Un archivo común es como el siguiente:

```
#
# /etc/fstab
#
# You should be using fstool (control-panel) to edit this!
#
# <device>    <mountpoint>   <filesystemtype> <options> <dump> <fsckorder>

/dev/hda2         /                         ext2   defaults 1 1
/dev/hdb1         /disco                    ext2   defaults 0 0
/dev/fd0          /mnt/floppy               ext2   defaults,users,noauto 0 0
/dev/hda1         none                      msdos  defaults 0 0

/proc             /proc                     proc   defaults
/dev/hda3         none                      swap   sw

cronos:/reas      /users                    nfs    rw
brahm:/u/alum     /users/alum1              nfs    rw
brahm:/u2/alum    /users/alum2              nfs    rw
```

B. El proceso contrario de montar es **desmontar** un sistema de archivos. Al igual que con el comando ‘`mount`’ tenemos tres distintas formas muy usuales del comando ‘`umount`’, estas son:

`#/bin/umount` *dispositivo* | *punto-de-montaje*
`#/bin/umount -a`
`#/bin/umount -t` *fstipo*

donde *dispositivo* es el nombre del dispositivo físico a desmontar y *punto-de-montaje* es el nombre del directorio donde fué montado. Solamente se necesita especificar una u otra opción. El comando tiene dos parámetros adicionales: `-a` desmonta todo el sistema de archivos, y `-t fs-tipo` actúa sólo en los sistemas de archivos especificado por fs-tipo.

```
#/bin/umount /dev/fd0

#/bin/umount /mnt
```

C. Eventualmente conectaremos un nuevo disco duro a nuestra computadora, cuando lo hacemos necesitaremos particionar el disco a fin de poder montar un sistema de archivos en él. La partición se realiza con el comando '`fdisk`'.

Para una referencia detallada de este comando puede revisar el manual de Linux sobre '`fdisk`'.

```
# fdisk drive
# fdisk /dev/hdb
```

las anteriores opciones son típicos para el comando.

D. Una vez que se ha realizado una partición en el disco con el comando '`fdisk`', debemos de crear el sistema de archivos antes de proceder a ocupar el disco para introducir los datos. Esto lo podemos realizar con el comando '`mkfs`'. La forma de usar el comando es la siguiente:

```
# mkfs [-V] [-t fs-tipo] [fs-opciones] sist_archivo [bloques]
```

sist_archivo	Es el nombre especial del sistema de archivos que se desea construir tal como /dev/hda1.
-V	Especificando esta opción inhibe más de una ejecución del sistema de archivos.
-v	*verbose.* Información extra.
-t fs-tipo	Especifica el tipo de sistema de archivo a construir. Ver el manual en línea de: fsck(8), mkfs.minix(8), mkfs.ext(8), mkfs.ext2(8), mkfs.xiafs(8). El tipo por omisión es minix.
fs-opciones	Las opciones con las que se creará el actual sistema de archivos.
-c	Verifica el dispositivo para los bloques malos antes de construir el archivo.
bloques	Especifica el número de bloques a usar por el sistema de archivos.

D.1 Un ejercicio práctico lo podemos realizar con nuestros discos flexibles utilizando el siguiente comando:

```
# /sbin/mkfs -t ext2 /dev/fd0 1440
```

E. En muchas ocasiones el sistema de archivos se daña. La causa más común es apagar la PC sin antes haber desmontado los sistemas de archivos. En los sistemas UNIX se ofrece el comando `'fsck'` (que a su vez es una liga a *e2fsck*)para reparar el sistema de archivos dañado. En Linux el comando tiene la siguiente sintaxis:

`# fsck` [-A] [-V] [-t *fs-tipo*] [-a] [-l] [-r] [-s] *sis_arch*

-A	Va a través del archivo /etc/fstab y trata de checar todo el sistema de archivos en una pasada.
-V	Imprime información adicional acerca de lo que '`fsck`' va haciendo.
-t fs-tipo	Especifica el tipo de sistema de archivo a verificar.
-a	Automáticamente repara cualquier problema que encuentra en el sistema de archivos sin preguntar. *Use esta opción con cuidado*
-l	Lista todos los nombres de archivos en el sistema de archivos.
-r	Pregunta la confirmación antes de reparar el sistema de archivos.
-s	Lista el superblock antes de checar el sistema de archivos.
sis_arch	Especifica el sistema de archivos a ser verificado.

El programa *e2fsck* está diseñado para correr rápidamente si es posible. Desde el chequeo del filesystem tienden que ser un disco atado, esto fue acabado para optimizar los algoritmos usando el *e2fsck* así esta estructurando el filesystem no son repetidos accesando desde el disco. En suma, el orden en el cual los inodos y los directorios son checados, ordenados por el número de bloques reducidos al importar el tiempo de discos buscados. Algunas de estas ideas son originalmente exploradas por [Bina y Emrath 1989] por esto tiene que ser refinados por los autores.

En el paso1, *e2fsck* itera sobre todos los inodos en los filesystem y checa sobre cada inodo con una no conectividad de objeto en el filesystem. Esto es, los chequeos no requieren unos sobrechequeos de los objetos del filesystem. Los ejemplos de cada chequeo incluye hechos seguros del archivo de modo legal, y esto es en todos los bloques en el inodo son número de bloques válidos. Durante el paso 1, los bitmaps indica cual bloque e inodo son en uso que están compilados.

Si *e2fsck* notifica datos de bloques cual son reclamados por mas de un inodo invoca el paso 1B por 1D que resuelve estos conflictos, uno a otro por clonar los bloques compartidos así esta cada inodo fue copiado el bloque compartido por desalojados uno o más de los inodos.

Paso 1 toma la longitud de tiempo a ejecutar, desde todos los inodos tienen que ser leído dentro de la memoria y checado. Reduce el tiempo de I/O necesario en futuros pases, la información del filesystem critico esta intercambiando en memoria. El mas importante ejemplo de esta técnica es la localización en el disco de todos los bloques en el filesystem. Esto obviamente necesita la re-lectura de la estructura de los inodos de directorios durante el paso 2 que contiene esta información.

El Paso 2 checa directorios como objetos sin conexión. Desde el directorio de entrada no resuelve bloques de disco, cada bloque de directorio puede checarse individualmente sin referencia a otros bloques de directorios. Esto aloja que el *e2fsck* ordena todos los bloques de directorios por el número de bloques y checa el directorio de bloques en orden ascendentes, así decrece la búsqueda del disco. Los bloques de directorios son checados a asegurar estas entradas de directorios que son válidos, y contiene referencia a los números de los inodos que están en uso (como se determinan por el paso 1).

Desde el primer directorio de bloque en cada inodo del directorio, el "." y ".." de entrada son checados asegurando, si este existe y este número del inodo de la entrada del "." Observando en el directorio corriente (El número del inodo de la entrada ".." no es checada hasta el paso 3). El paso 2 también checa la información concierte el directorio padre en cual cada directorio es ligado. (si un directorio es referente por más que en un directorio, la segunda referencia del directorio es negociado como una ilegal liga dura y es removida). Es notable notar que el fin del paso 2, casi todos los I/O del disco el cual el comando *e2fsck* necesita ejecutarse completo. Información de los pasos 3, 4 y 5 son intercambiados en memoria ; De aquí los pasos quedan en *e2fsck* que son largamente atados al CPU, y toma pérdidas que están al 5-10 % del tiempo total que corre el *e2fsck*.

En el paso 3, la conectividad del directorio es checado. Los trazos *e2fsck* del rutea de cada directorio atrasado a la raíz, usando la información que esta fue intercambiada durante el paso 2. Como estos tiempos, la entrada ".." de cada directorio es también checado a hacer seguro es válido. Algunos directorios cual no pueden ser trazados atrás del root son ligados al directorio */lost+found*.

En el paso 4, *e2fsck* checa la referencia de contar todos los inodos, por interacción sobre todos los inodos y compara las ligas contadas (cual fue intercambiando en el paso 1) fue contador interno compuesto durante el paso 2 y 3. Algunos archivos sin borrar con un contador de liga cero es también ligado al directorio */lost+found* durante este paso.

Finalmente en paso 5, *e2fsck* checa la validación del sumario de la información del filesystem. Compara el bloque y el bitmap del inodo cual fue construido durante los pasos previos contra los cuales bitmaps en el filesystem, y correctos las copias en disco si es necesario.

la forma más común de usar '`fsck`' es la siguiente:

```
# fsck /dev/hda3
```

DESARROLLO

1. Entre a sesión como super usuario.

2. Inserte un disco de 3.5" de alta densidad en la unidad de disco del servidor Linux. Ejecute el siguiente comando:

```
# /sbin/mkfs -t ext2 /dev/fd0 1440
```

3. Monte el sistema de archivos recién creado en el disco flexible en el sub-directorio `/mnt`. Y verifique que el kernel lo reconoce como parte del sistema de archivos. Ejecute los siguientes comandos:

```
# /bin/mount -t ext2 /dev/fd0 /mnt
# /bin/mount
```

4. Copie el archivo /etc/passwd al disco flexible. Ejecute el siguiente comando:

```
# cp /etc/passwd /mnt/passwd
```

5. Compruebe que el archivo se copió realmente. Ejecute los siguientes comandos:

```
# cd /mnt
# ls -l
```

6. Desmonte el sistema de archivos que se esta usando en el disco flexible. Ejecute el siguiente comando:

```
# umount /mnt
```

Una forma alternativa de hacerlo es: `umount /dev/fd0`

7. Verifique la integridad del sistema de archivos creado en el disco flexible: Ejecute el siguiente comando:

```
# fsck /dev/fd0
```

8. Monte nuevamente el sistema de archivos del disco flexible y ejecute el comando `'fsck'` sobre éste sistema de archivos. ¿ Qué sucede ? ¿ Por qué me pregunta si estoy seguro de continuar con la ejecución del comando ?

CUESTIONARIO:

1. ¿Cuál es el propósito del comando ‘`mount`’?
2. ¿Cuál es el propósito del comando ‘`umount`’?
3. ¿Cuál es el propósito del comando ‘`fdisk`’?
4. ¿Cuál es el propósito del comando ‘`mkfs`’?
5. ¿Cuál es el propósito del comando ‘`fsck`’?
6. ¿En que situaciones es conveniente utilizar el comando ‘`fsck`’?. Recuerde el punto 8 del desarrollo de la práctica. Anote sus comentarios y conclusiones al respecto.
7. ¿ El nombre del archivo de dispositivo que se le pasa como argumento a `‘fsck’`, es de tipo modo bloque o modo caracter ?
8. ¿ El nombre del archivo de dispositivo que se le pasa como argumento a `‘mount’` y `‘umount’`, es de tipo modo bloque o modo caracter ?
9. Investigue como montar un disco flexible que tiene formato de DOS y poder copiar archivos de UNIX a DOS y viceversa. Escriba el comando que utilizó para montar el disco flexible.
10. Se puede verificar una partición montada vía NFS con el comando fsck. ¿Sí o no? ¿Porque?

CONCLUSIONES: