PRACTICA #7

MANEJO DEL SISTEMA DE ARCHIVOS

OBJETIVO:

INTRODUCCIÓN:

Existen varios tipos de archivos:

1.- *Archivos ordinarios*. Son los más comunes, son los que almacenan datos, es decir, puede ser un programa, un archivo de texto, código fuente o cualquier cosa que pueda guardarse en cualquier lugar. El kernel soporta acceso secuencial y aleatorio en todos estos archivos.

2.- *Directorios* tienen en común con los archivos ordinarios en que también contienen datos, sólo que en este caso el dato es una lista de otros archivos.

3.- *Archivos especiales* se identifican porque cada uno tiene un número de dispositivo mayor y uno menor (major and minor device number). El número mayor identifica al manejador del dispositivo que necesita el kernel para acceder al dispositivo. El número menor significa un parámetro dependiente del manejador del dispositivo usado típicamente para diferenciar entre diversos tipos de dispositivos soportados por el manejador, o distintos modos de operación. Utilizan un i-nodo pero no bloques de datos. Representan dispositivos en los que se pueden leer o escribir cantidades arbitrarias de datos. Incluyen sistemas de archivos, puertos seriales, puertos paralelos, terminales y cintas. También se les conoce como *"raw devices"* debido a que no manipulan la I/O. Los discos duros y flexibles pueden ser accedidos de esta manera. Podemos ver archivos de este tipo en el directorio */dev*.

4.- *Entubamientos* (FIFO [**First In First Out**]). Son aquellos que permiten la comunicación entre dos procesos ejecutándose en el mismo nodo. Los entubamientos pueden ser creados con el comando `mknod` y eliminados con el comando `rm`.

5.- *Ligas Duras*, en realidad una liga no es un archivo, es un nombre adicional para otro archivo. Cada archivo tiene al menos una liga, usualmente el nombre bajo el cual fue originalmente creado. Cuando se hace una nueva liga hacia un archivo, un alias para este archivo es creado. Una liga es indistinguible del archivo al cual está ligado; LINUX mantiene el conteo de la cantidad de ligas que apuntan hacia un archivo en particular y no libera el espacio que ocupa el archivo hasta que la última liga es eliminada. La liga dura es una conexión directa entre archivos, por lo que ésta no puede existir a través de distintos sistemas de archivos.

6.- *Ligas Simbólicas* son archivos que simplemente contienen el nombre de otro archivo. Cuando el kernel trata de abrir o pasar a través de la liga, su atención es directamente hacia el archivo

que la liga simbólica apunta en vez de abrir la liga simbólica en sí. La diferencia entre las ligas, es que las duras son una referencia directa, mientras las simbólicas son una referencia a través de un archivo, las simbólicas son el archivo en sí, por lo tanto, tienen un propio dueño y permisos.

7.- Sockets son conexiones entre procesos que les permiten comunicarse de una manera más rápida y fácil. Existen varios tipos de sockets en UNIX, muchos de los cuales involucran el uso de la red. Los sockets son locales a un modo en particular y son referenciados a través de un objeto en el file system en vez de un puerto en la red. Los archivos de sockets son visibles a los demás procesos como entradas en el directorio, estas entradas no pueden ser leídas o escritas por procesos que no estén involucrados en la conexión del socket.

Los sockets en UNIX son creados con la llamada al sistema `socket()`y pueden ser eliminados con el comando `rm` con la llamada al sistema `unlink ().`

La llamada para abrir un canal bidireccional de comunicaciones es socket y se declara como sigue:

```
#include <sys/typesh.h>
#include <sys/socket.h>

int socket(af,type,protocol)
int af,type, protocol;
```

Socket crea un punto terminal para conectarse a un canal y devuelve un descriptor. El descriptor de socket devuelto se usará en llamadas posteriores a funciones de la interfaz.

Af (address family) especifica la familia de sockets o familia de direcciones que se desea emplear. Las distintas familias están definidas en el archivo de cabecera <*sys/socket.h*> y dependerán del fabricante del sistema y de la configuración del hardware. Las 2 familias siguientes suelen estar presentes en todos los sistemas:

`AF_UNIX` Protocolos internos UNIX. Es la familia de sockets empleada para comunicar procesos que se ejecutan en una misma máquina. Esta familia no requiere que esté presente un hardware especial de red, puesto que en realidad no realiza accesos a ninguna red.

`AF_INET` Protocolos Internet. Es la familia de sockets que se comunican mediante protocolos, tales como *TCP (Transmission Control Protocol),* desarrollado por la Universidad de California en Berkeley para *DARPA (Defense Advance Research Projects Agency)* ó *UDP (User Datagram Protocol).*

`AF_CCINT` Norma X.25 del CCITT.

`AF_NS` Protocolo NS de Xerox..

El argumento *type* indica la semántica de la comunicación para el socket. Puede ser:

`SOCK_STREAM` Socket con un protocolo orientado a conexión. Esto es lo que hemos estudiado como circuito virtual.

`SOCK_DGRAM` Socket con un protocolo no orientado a conexión o datagrama.

Protocol especifica el protocolo particular que se va usar en el socket. Normalmente, cada tipo de socket tienen sólo un protocolo, pero si hubiera más de uno, se especificaría mediante un argumento. *Protocol* puede valer cero, en cuyo caso la elección del protocolo se deja en manos del sistema. Si la llamada se ejecuta satisfactoriamente, devolverá un descriptor de fichero válido. En caso contrario, devolverá -1 y el *errno* estará codificado al error producido.

Los *Permisos de Archivos* son el conjunto de nueve bits de permisos asociados a ellos, que controlan quien puede leerlos, escribirlos o ejecutarlos. Y junto con otros tres bits que afectan la forma de ejecutarlos, constituyen los llamados *bits de modo*, que son los permisos totales del archivo. Los doce bits de modo son guardados con otros 4 bits tipo de archivo son fijados cuando el archivo es creado y no pueden ser modificados, pero los otros 12 bits pueden ser alterados por el dueño o por el superusuario usando el comando `chmod`. El comando `/bin/ls -l` es utilizado para examinar el valor de estos bits.

Los bits correspondientes son el *setuid*, *setgid* y *sticky bit* y estos valores octales son 4000, 2000 y 1000 respectivamente.

Fijando el bit SUID (Set User ID) en un archivo, permite que los usuarios ejecuten los permisos del dueño del archivo.

Fijando el bit SGID (Set Group ID) en un archivo, permite que los usuarios ejecuten los permisos del grupo del archivo.

El *Sticky bit* (bit pegajoso). Es el bit con valor octal de 1000. Cuando éste bit es fijado a un archivo ejecutable, el bit le dice al sistema operativo que el archivo se ejecutará con mucha frecuencia y por lo tanto deberá ser retenido en la memoria principal aún cuando no sea ejecutado. Esto desperdicia memoria principal, pero reduce el tiempo de ejecución del programa significativamente.

Codificación de tipo de archivo usado por *ls*			
Tipo de archivo	Símbolo	Creado por	Eliminado por
Archivo regular	"-"	editores, cp , etc.	rm
Directorio	"d"	mdkir	rmdir, rm -r
Dispositivo de modo carácter	"c"	mknod	rm
Dispositivo de modo bloque	"b"	mknod	rm
Socket	"s"	socket(2)	rm
Named pipe	"P"	mknod	rm
Liga simbólica	"l"	ln -s	rm

DESARROLLO:

Veremos la diferencia del concepto del liga suave o dura. Compararemos estos archivos:

1) Crea un archivo llamado liga.txt, escribe lo que gustes y posteriormente crearemos una liga

```
#ln liga.txt liga.ln
#ls -li
```

Escribe la salida de este resultado

Comenta el resultado obtenido.

2) Ahora borraremos el archivo llamado *liga.txt* y veremos que pasa con el archivo *liga.ln*

```
#rm liga.txt
#cat liga.ln
#ls -li
```

Escribe el resultado

Que paso con la información de *liga.ln* y que concluyes con ello.

3)Ahora veremos la diferencia de la liga dura. Creemos otro archivo llamado *ligado.txt* y haremos la liga suave

```
#ln -s ligado.txt liga2.ln
#ls -li
```

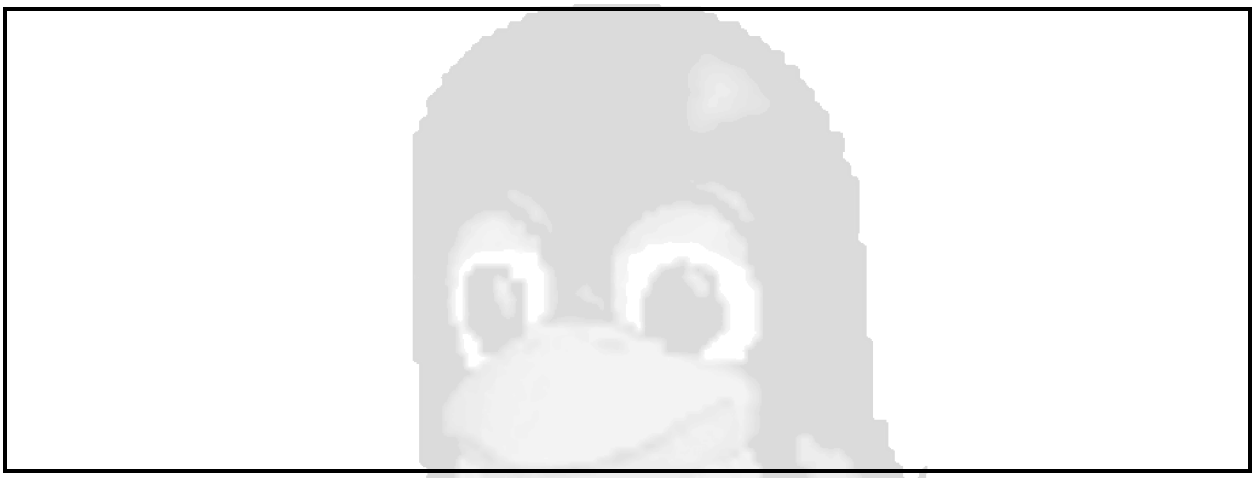
Escribe el resultado

Comenta la salida y que concluyes.

4)Ahora borremos el archivo *ligado.txt* y veremos la información que contiene.

```
#rm ligado.txt
#cat liga2.ln
#ls -li
```

Obten el resultado



Da tus conclusiones y que vez al respecto.

5) Ve otra aplicación de liga simbólica. Dirigete a tu directorio de casa y crearemos una liga suave :

```
#cd
#ln -s /tmp temporal
#cd temporal
#ls -la
#cd /tmp
# ls -l
```

Compara los resultado y escribe tus conclusiones.

CUESTIONARIO:

1.- ¿Cuántos tipos de archivos existen?
2.- ¿Qué es un archivo FIFO? ¿Cómo funciona?
3.- ¿Qué es una Liga Dura?
4.- ¿Qué es una Liga Simbólica?
5.- ¿Qué es un Entubamientos?
6.- ¿Qué se conoce como los Permisos de archivos?
7.- ¿Qué se conoce como el Sticky bit?

CONCLUSIONES: