Universidad Nacional de La Matanza
***Ingeniería en Informática*-Taller de GNU/Linux 2003**

TP N° 13
Enlaces duros y simbólicos (hard links & symlinks)

Objetivos:

- Comprender el significado y la diferencia entre enlaces duros y enlaces simbólicos.
- Utilizar los comando relacionados al manejo de enlaces.

Manejando enlaces de archivos

Los enlaces le permiten dar a un único archivo múltiples nombres. Los archivos son identificados en el sistema por su número de inodo, el cual es el único identificador del archivo para el sistema de archivos. Un directorio es una lista de números de inodo con sus correspondientes nombres de archivo. Cada nombre de archivo en un directorio es un enlace a un inodo particular.

Enlaces duros (Hard links)

El comando "ln" es usado para crear múltiples enlaces para un archivo. Por ejemplo, supongamos que tiene un archivo "foo" en un directorio. Usando "ls -i", veremos el número de inodo para el archivo.

```
# ls -i foo
  22192 foo
```

Aquí, el archivo foo tiene el número de inodo 22192 en el sistema de archivos. Podemos crear otro enlace a foo, llamado bar:

```
# ln foo bar
```

Con ls -i veremos que los dos archivos tienen el mismo inodo.

```
# ls -i foo bar
 22192 bar 22192 foo
```

Ahora, accediendo a foo o a bar accederemos al mismo archivo. Si hace cambios en foo, estos cambios también serán efectuados en bar. Para todos los efectos, foo y bar son el mismo archivo. Estos enlaces son conocidos como enlaces duros (hard links) porque directamente crean el enlace al inodo. Nótese que solo podemos crear enlaces duros entre archivos del mismo sistema de archivos. Los enlaces simbólicos (ver mas adelante) no tienen esta restricción. Cuando borra un archivo con rm, está solamente borrando un enlace a un archivo. Si usa el comando

```
# rm foo
```

La orden ls -i mostrará los números de inodo. Solo el enlace de nombre foo es borrado; bar todavía existirá. Un archivo es borrado definitivamente del sistema cuando no quedan enlaces a él. Usualmente, los archivos tienen un único enlace, por lo que el uso de rm los borra. Pero si el archivo tiene múltiples enlaces, el uso de rm solo borrara un único enlace; para borrar el archivo, deberá borrar todos los enlaces del archivo.
La orden ls -l muestra el número de enlaces a un archivo (entre otra información).

```
# ls -l foo bar
   -rw-r--r-- 2 root root 12 Aug 5 16:51 bar
   -rw-r--r-- 2 root root 12 Aug 5 16:50 foo
```

La segunda columna en el listado, "2", especifica el número de enlaces al archivo. Un directorio no es mas que un archivo que contiene información sobre la translación enlace a inodo. También, cada directorio tiene al menos dos enlaces duros en él: **"."** (un enlace apuntando a sí mismo) y **".."** (un enlace apuntando al directorio padre). En el directorio raíz (/), el enlace **".."** simplemente apunta a /.

Enlaces simbólicos

Los enlaces simbólicos son otro tipo de enlaces, diferente al enlace duro. Un enlace simbólico permite dar a un archivo el nombre de otro, pero no enlaza el archivo con un inodo. Dentro del enlace simbólico se guarda la ruta donde ubicar al archivo destino.
La orden "ln -s" crea un enlace simbólico a un archivo. Por ejemplo, si usamos la orden

```
# ln -s foo bar
```

crearemos un enlace simbólico bar apuntando al archivo foo. Si usamos ls -i, veremos que los dos archivos tienen inodos diferentes, en efecto.

```
# ls -i foo bar
 22195 bar 22192 foo
#
```

De cualquier modo, usando ls -l vemos que el archivo bar es un enlace simbólico apuntando a foo.

```
# ls -l foo bar
  lrwxrwxrwx 1 root root 3 Aug 5 16:51 bar -> foo
  -rw-r--r-- 1 root root 12 Aug 5 16:50 foo
#
```

Los bits de permisos en un enlace simbólico no se usan (siempre aparecen como rwxrwxrwx). En su lugar, los permisos del enlace simbólico son determinados por los permisos del archivo "apuntado" por el enlace (en nuestro ejemplo, el archivo foo).

Funcionalmente, los enlaces duros y simbólicos son similares, pero hay algunas diferencias. Por una parte, se puede crear un enlace simbólico a un archivo que no existe; lo mismo no es cierto para enlaces duros. Los enlaces simbólicos son procesados por el núcleo de forma diferente a los enlaces duros, lo cual es solo una diferencia técnica, pero a veces importante. Los enlaces simbólicos son de ayuda puesto que identifican al archivo al que apuntan; con enlaces duros no hay forma fácil de saber que archivo esta enlazado al mismo inodo.

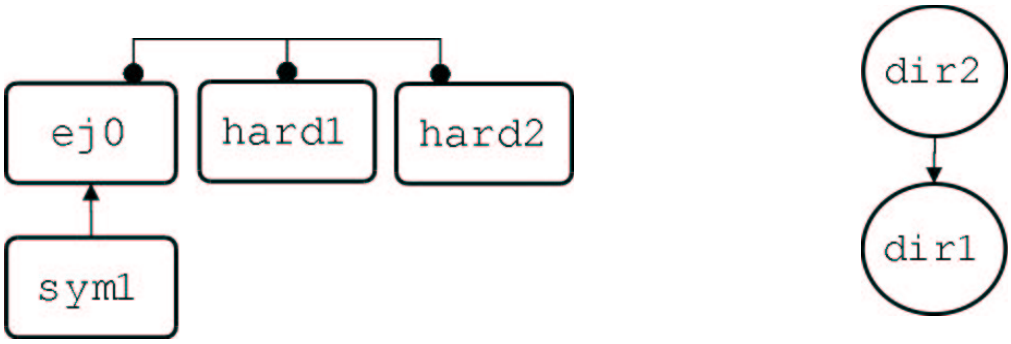
Los enlaces se usan en muchas partes del sistema Linux. Los enlaces simbólicos son especialmente importantes para el sistema de librerías compartidas en /lib.

La operación 'rm' sobre un fichero simbólico no actua sobre el fichero apuntado sino sobre el propio enlace simbólico destruyéndolo.

Ejercicios

1) Crear un archivo de texto llamado **ej0** que contenga el NOMBRE y AELLIDO del alumno.
2) Crear un par de enlaces duros con **ej0** llamados **hard1** y **hard2.**
3) Crear un enlace simbólico a **ej0** llamado **sym1**
4) Crear un directorio **dir1** y crear un enlace llamado **dir2**.
5) Hacer un ls -l para ver los enlaces creados.

$
$
$
$
$



**nota: En el gráfico, los enlaces duros se simbolizan con líneas con terminación en círculo. Los enlaces simbólicos se grafican con líneas terminadas en flecha.*

Los enlaces rígidos no muestran nada particular listados con 'ls -l' pero los enlaces simbólicos vienen acompañados de una flecha que apunta a otro nombre de fichero.
Hay una columna de números a continuación de los permisos. Se trata de una columna que indica el número de enlaces rígidos que están asociados a un mismo fichero.

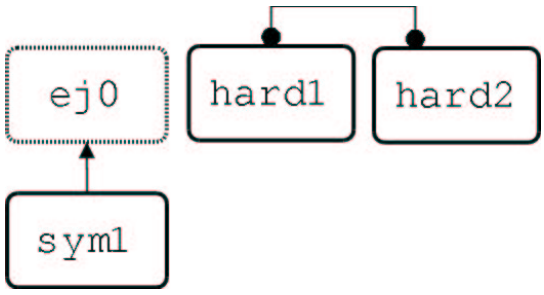
En el caso de **'ej0', 'hard1' y 'hard2'** aparece un 3 y está claro porque son enlaces creados por nosotros pero el directorio '**dir1**' tiene un 2. Esto significa que existe otro enlace rígido para ese directorio que nostros no hemos creado. Se ha creado automáticamente al crear '**dir1**'. Acuerdese que todos los directorios se crean con un par de entradas que son '.' y '..' El 2 por lo tanto en este caso se debe a la entrada '.' dentro del propio **'dir1'** y si dentro de **'dir1'** existieran directorios habría que contabilizar cada uno de los '..' de los directorios hijos como enlaces rígidos de '**dir1**'.

6) Ver el contenido de **ej0**.
7) Agregar (usando el operador >>) el texto DNI al archivo hard1.
8) Ver nuevamente el contenido de **ej0**.

$	$
$	$

La información es accesible desde distintos nombres de ficheros pero no son copias sino que se trata de la misma unidad de información.

9) Borrar **ej0**. Hacer un cat a **sym1**. Hacer un cat a hard1.

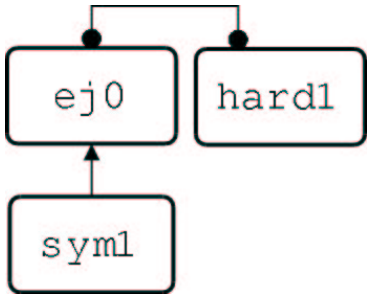


En el caso de los enlaces rígidos da igual cual es el enlace o fichero original. Son totalmente equivalentes y la información solo desaparecerá del sistema cuando el último enlace rígido sea eliminado. La diferencia con el enlace simbólico es que actua simplemente accediendo al nombre

del fichero que tiene almacenado en su interior. Por eso en el caso que acabamos de ver 'sym1' queda apuntando a 'ej0' que es un fichero que ya no existe.

10) Hacer un ls -li

La opción -i sirve para visualizar el número de inodo. Un inodo es una clave numérica para el acceso al sistema plano de ficheros donde cada almacén de información tiene una única clave y por eso los distintos enlaces rígidos contienen el mismo valor de inodo.



11) Cambiar el nombre de '**hard2**' por '**ej0**' usando el comando mv. Con esto el link simbólico debería funcionar nuevamente.

12) Verificar que **sym1** es válido (cat **sym1**)

13) Mover el enlace simbólico '**sym1**' a '**dir1**' usando el comando mv.
14) Verificar si funciona **sym1** (que está ahora dentro de **dir1**).

Hay error? Por que? :

Los enlaces simbólicos son relativos si no se especifica la ruta completa del target al momento de su creación.

15) Trasladar '**ej0**' a '**dir2**' y comprobar como queda todo y si **./dir1/sym1** es válido nuevamente

16) Ahora vamos acrear un symlink con ruta absoluta. Dentro de **dir1** creamos un symlink de nombre "**ultimo_symlink**", pero esta vez especificando el path completo del target **(/..../dir1/ej0)**. Una manera de hacerlo fácil para los casos donde el path completo sea muy largo:

```
ln -s `pwd`/ej0 ultimo_symlink
```

$

17) Verificar que ultimo_symlink funcione correctamente y luego moverlo a otro directorio. El symlink debería seguir apuntando correctamente a su target.

El texto es una adaptaciones de:
http://lucas.hispalinux.es/Manuales-LuCAS/LIPP/lipp-1.1-html-2/lipp.htm
Los gráficos fueron creados con "dia" versión 0.88.1
Mayor información sobre enlaces e inodos (en castellano):
Introducción al sistema de ficheros EXT2FS http://www.gui.uva.es/~vortex/