

# CAPITOLO 10

## I LINK

Pietro Buffa

Spesso succede che un utente abbia la necessità di mettere lo stesso file in più di una directory. Copiando il file in questione in più directory, occuperete inutilmente molto spazio nel disco (pensate ad un file contenente un filmato), se invece lo spostate, perdete l'ordinamento iniziale. E' a questo punto che vengono in vostro soccorso i **link** (collegamenti), che costituiscono un metodo particolare per indicare al filesystem che volete che un file compaia in directory diverse, senza doverne copiare i dati. Con i link è possibile quindi associare più nomi ad uno stesso file e creare così delle scorciatoie molto utili. Per chiarire meglio cosa sono i link ed i differenti tipi di link che possiamo creare sotto GNU/Linux, bisogna prima chiarire il concetto di i-node.

### 10.1 ANCORA SUI FILE: gli I-NODE

Ad ogni file esistente in un filesystem è intimamente associata una particolare struttura che contiene i cosiddetti metadati. Questa struttura prende il nome di **index-node** (nodo indice) o, in forma abbreviata, **i-node**. Ogni struttura i-node è contraddistinta da un numero e contiene circa una ventina di metadati, tra cui:

**uid** e **gid** che rappresentano rispettivamente l'utente ed il gruppo proprietario del file.

**mode** che descrive il tipo di file ed i suoi permessi.

**size** la dimensione in byte del file e i **block**, il numero di blocchi occupati dal file.

**atime**, **ctime** e **mtime** che rappresentano rispettivamente il tempo di ultimo accesso al file, il tempo di ultima modifica dell'i-node e quello di ultimo accesso dei contenuti del file.

**link\_count** rappresenta il numero di hard link associati al file (come vedremo nel corso di questo capitolo).

**block** riferimento ai data block che contengono i dati veri e propri: senza questo parametro il sistema non sarebbe in grado di recuperare i contenuti di un file.

Ogni i-node occupa circa 128 byte, il che significa che se avete nel vostro sistema 2 milioni di files (un numero assolutamente normale di questi tempi), le tabelle degli i-node occuperanno in tutto più di 200MB di spazio.

Quando creiamo un file dandogli un bel nome, in effetti il sistema sta collegando (linkando) il nome del file alla sua struttura i-node, che verrà creata ed assegnata al file in questione dal sistema stesso e che rappresenta per il sistema la collocazione fisica di quel file all'interno del filesystem. Sotto questa nuova luce possiamo dunque immaginare una directory come un elenco di numeri i-node associati ai rispettivi nomi dei files.

Poniamo che abbiate un file di nome "relazione\_2005" all'interno di una data directory, usando il comando `ls` che già conosciamo, con la corretta opzione:

```
> ls -li relazione_2005
```

Possiamo vedere il "numero di i-node" associato a quel file. L'output sarà qualche cosa di simile a questo:

```
22192 relazione_2005
```

GNU/Linux mette a disposizione dell'utente due diversi tipi di link: **hard link** e **symbolic link**, vediamo di cosa si tratta e come utilizzarli.

### 10.2 HARD LINK

Possono essere creati con il comando:

```
> ln nomefile nomelink
```

Tramite questo comando viene aggiunto alla directory dove si vuole creare “nomelink” un riferimento alla struttura **i-node** del file, incrementando il **link count** del file originale (che normalmente è uguale ad 1). Quest’ultima operazione serve ad evitare che un file condiviso tra più utenti venga cancellato quando un utente cancella il suo link.

Quando si cancella un link ad un determinato file, ad esempio usando il comando **rm**, il sistema in effetti non fa altro che decrementare il link count del file originale: solo se questo scenderà a 0, il file originale verrà cancellato.

Per spiegare meglio questo concetto, si fa spesso riferimento all’esempio del cane con molti guinzagli: il cane è il vostro file ed i guinzagli sono tutti gli hard link al file stesso. Il cane non può scappare finché almeno una persona lo tiene con un guinzaglio, il che vale a dire che un file non viene cancellato finché rimane almeno un link.

Proviamo ad esempio a creare un hard link al file “relazione\_2005” :

```
bash:~> ln relazione_2005 relazione_finale
```

```
22192 relazione_2005 22192 relazione_finale
```

Da questo momento, il riferirsi a “relazione\_2005” oppure a “relazione\_finale” è la stessa cosa, avendo i due file lo stesso numero di i-node. Se adesso apportate delle modifiche a “relazione\_finale” queste appariranno anche in “relazione\_2005” perché di fatto, si tratta dello stesso file.

Gli hard link però, hanno due limitazioni importanti:

- Non è possibile creare collegamenti ad una intera directory.
- E’ possibile creare link ad un file, soltanto nella partizione del disco in cui questo risiede.

Proprio per superare questi problemi, GNU/Linux mette a disposizione i cosiddetti symbolic link .

## 10.3 SIMBOLIC LINK

Possono essere creati semplicemente utilizzando l’opzione -s nel precedente comando:

```
> ln -s nomefile nomelink
```

A differenza di un hard link, un symbolic link non è altro che un nuovo file, di tipo particolare, che contiene il path (percorso) del file “puntato”, o file originale. Facciamo lo stesso esempio di prima usando un symbolic link, proviamo ad esempio a creare un symbolic link al file “relazione\_2005” :

```
bash:~> ln -s relazione_2005 relazione_finale
```

```
22192 relazione_2005 22195 relazione_finale
```

Con il comando **ls -i**, possiamo evidenziare che i due file hanno effettivamente i-node differenti.

Inoltre usando il comando **ls -l**, la tabella evidenzierà come il file “relazione\_finale” sia un symbolic link al file “relazione\_2005”:

```
lrwxrwxrwx 1 bios bios 12 Aug 5 16:51 relazione_finale -> relazione_2005
```

↑  
↑  
numero di link al file

la lettera “l”, che precede tutte le altre lettere  
ci dice che il file è un symbolic link

↑  
la freccia ci dice a quale file  
è collegato.

I permessi, ossia tutti quegli strani simboli iniziali che descriveremo approfonditamente nel capitolo 12, su un symbolic link non sono usati, ma appaiono comunque nella forma **lrwxrwxrwx**, in realtà al symbolic link vengono applicati i permessi del file originale a cui esso è collegato perciò, se è vietata la scrittura sul file originale, questa sarà vietata anche nel suo symbolic link.

L'accesso ai symbolic link è più lento di quello agli hard link (il sistema deve prima aprire il file contenente il path, leggerlo, effettuarne il parsing e cercare il file originale) e l'aggiunta di un symbolic link non incrementa il link count di un file (il che vuol dire che è possibile che nel vostro disco si trovino dei "broken link", cioè dei symbolic link che puntano a file che sono stati cancellati), ma la maggiore flessibilità di questo approccio rispetto agli hard link lo rende, di fatto, la soluzione più comunemente usata.

## 10.4 OLTRE I "REGULAR FILES"

Come abbiamo visto, non tutti i file di un sistema Linux hanno lo scopo di contenere dati. Alcuni di essi servono come collegamenti ad altri file (link), altri fungono da contenitori (directory). Esistono anche tipi di file ancora più "esotici", come i file di device (dispositivi) che, come abbiamo accennato nel capitolo 6, si trovano all'interno della directory di sistema */dev* e si dividono in file a caratteri o a blocchi a seconda del metodo di accesso al dispositivo.

Adesso cominciamo a prendere coscienza del perché molti articoli tecnici sui sistemi UNIX/Linux spesso utilizzino l'espressione: "*In UNIX everything is a file*", tutto è rappresentato da un file.