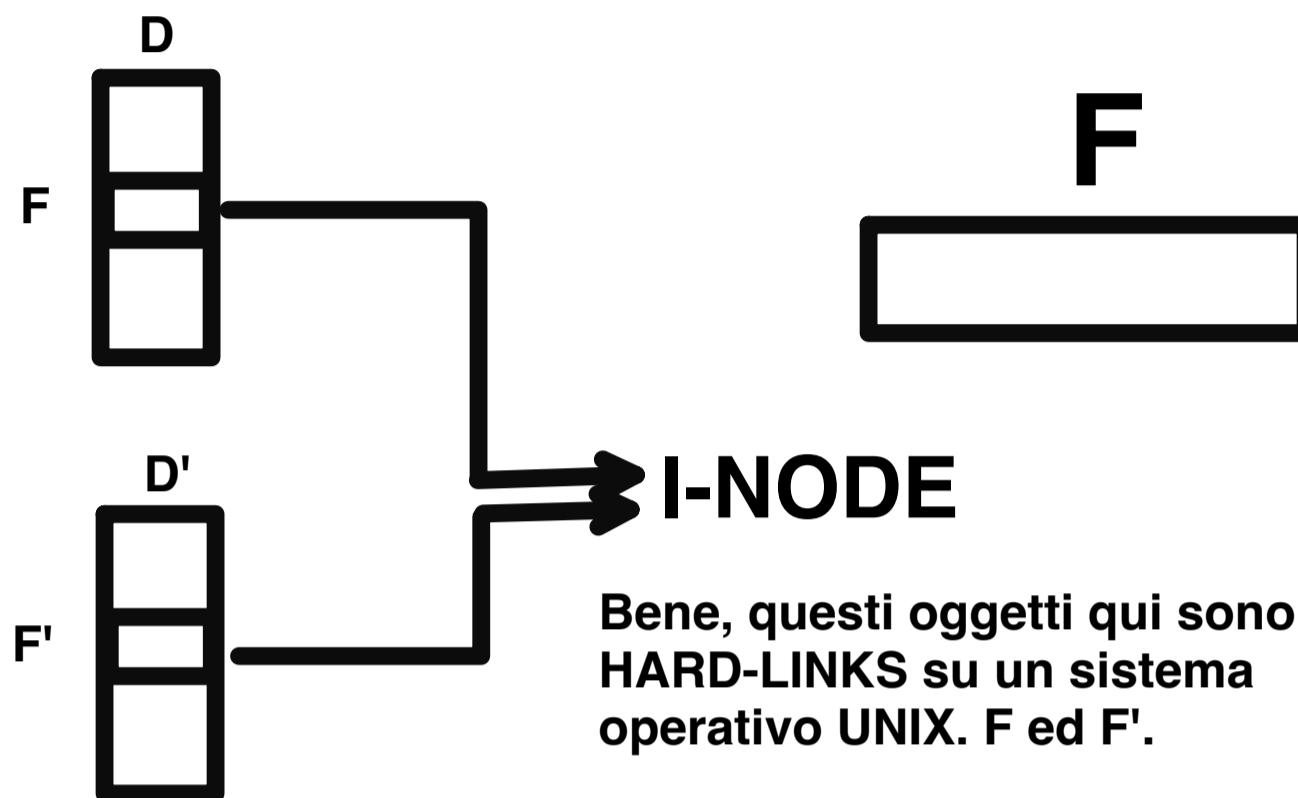


Hard links e rimozione

Supponiamo di avere un oggetto di I/O che sia un file, ma in generale questa cosa è vera anche se non sono file, e supponiamo che questo file sia associato ad uno specifico I-NODE che noi abbiamo all'interno del sistema. Chiaramente questo file potrà esistere all'interno di qualche directory: potremmo avere una directory D il cui interno del file speciale ci dice che esiste un file che si chiama F e il suo I-NODE è questo.

Noi potremmo avere un'altra directory D' con un file speciale che la rappresenta e ci potrà essere all'interno una entry che dice che all'interno di questa directory D' abbiamo un file che si chiama F' che è esattamente lo stesso di prima. Quindi arriviamo sullo stesso I-NODE.



Quindi sono due collegamenti diretti verso lo stesso I-NODE.

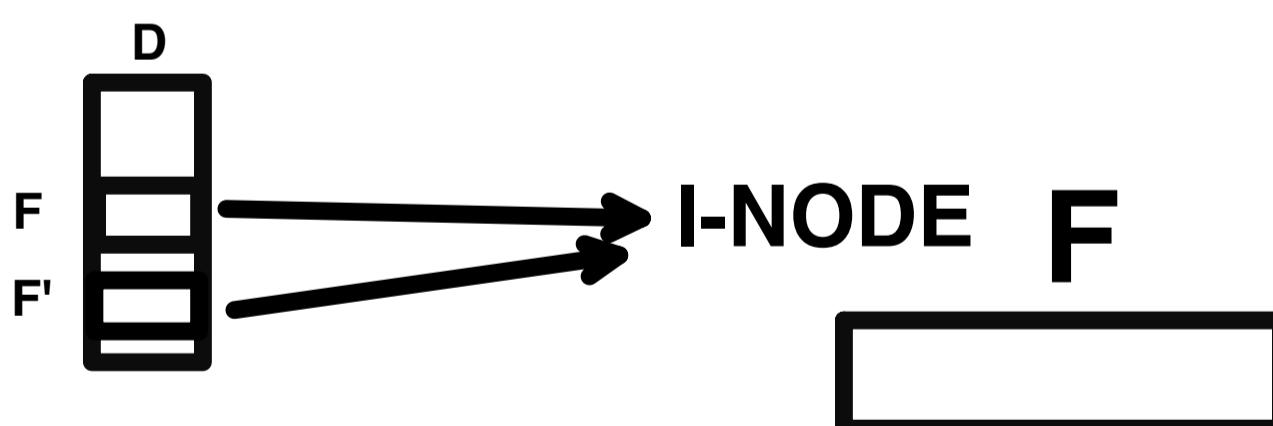
Se noi consideriamo gli attributi che abbiamo all'interno di un I-NODE, E SE CI FACCIAMO CASO, tra i metadati dell'i-node non c'è il "nome del file". Infatti su un sistema operativo UNIX, un file può avere più di un nome. Uno è F, e l'altro è F', ma sempre di un file si tratta.

F è un hard-link nella directory D, ed F' è un hard-link nella directory D'.

Ma F = F'!

Quindi al posto di F' possiamo mettere anche F, è lo stesso.

Ma posso avere anche due file F ed F' all'interno della stessa directory D, che puntano allo stesso I-NODE.



Su unix possiamo Hard-Linkare elementi dello stesso file system: Per creare un hard link basta chiamare la system call link:

```
int link(const char *oldpath, const char * newpath)
```

ritorna -1 in caso di fallimento

Passiamo il puntatore al vecchio nome - quindi all'hard-link già esistente - e ovviamente definiamo qual è il nuovo nome verso lo stesso contenuto.

Ovviamente il nuovo nome potrà definire anche qual è la nuova directory dove noi vogliamo collocare questo nuovo hard-link verso lo stesso contenuto.

Il file verrà rimosso quando non ci saranno più i collegamenti. La rimozione di file su un sistema operativo Linux si basa sulla rimozione degli hard links verso il file.

Se eliminiamo solo F stiamo eliminando solo un hard-link, e il contenuto del file ancora persiste all'interno del file system e raggiungibile utilizzando un altro nome, quando chiamiamo una open e andiamo in un'altra directory ad aprire un file in quella directory e da quella directory otteniamo l'indicazione di qual è l'I-

NODE e quindi sappiamo tutto.

Il contenuto del file e il rispettivo i-NODE viene rimosso quando rimuoviamo tutti gli hard-links verso quell'i-NODE.

Esempio di funzione per rimuovere un file:

```
int unlink(const char *pathname)
```

ritorna -1 in caso di fallimento

Questo lo fa il comando rm su linux.

```
uffer-overflow> ls -lai
total 60
11667096 drwxr-x 2 francesco users 4096 Apr 15 12:35 .
11666968 drwxr-xr-x 4 francesco users 4096 Mar 18 13:14 ..
11667007 -rw-rxr-xr-x 1 francesco users 13112 Mar 11 12:33 a.out
11667101 -rw-r--r-- 1 francesco users 781 Mar 18 2020 exploit.c
11667167 -rw-r--r-- 1 francesco users 7 Apr 15 12:13 f
11667103 -rw-r--r-- 1 francesco users 489 Mar 14 2018 Makefile
11667168 -rw-r--r-- 1 francesco users 0 Apr 15 12:35 pippo
11667030 -rwxr-xr-x 1 francesco users 12552 Mar 11 12:33 print-string
11667100 -rw-r--r-- 1 francesco users 905 Mar 2 2019 print-string.c
11667102 -rw-r--r-- 1 francesco users 136 Mar 2 2019 README
```

Numero degli hard-links
verso questo specifico I-
NODE

Se è 1 significa che il file è raggiungibile da un solo nome all'interno del virtual file system.

Esso è il contatore degli hard link che arrivano su uno specifico I-NODE.

Esso prende il nome anche di "reference Counter" ossia quanti sono i riferimenti intesi come hard-link verso questo i-node.