File System
II parte

Cosa è il FS

E' quella parte del Sistema Operativo che fornisce i meccanismi di accesso e memorizzazione delle informazioni (programmi e dati) allocate in memoria di massa.

Realizza i concetti:
- di **file**: unità logica di memorizzazione
- di **directory**: insieme di file (e directory)
- di **partizione**: insieme di file associato ad un particolare dispositivo fisico (o porzione di esso)

N.B. Le caratteristiche di file, direttorio e partizione sono del tutto indipendenti dalla natura e dal tipo di dispositivo utilizzato.

Tabelle Unix

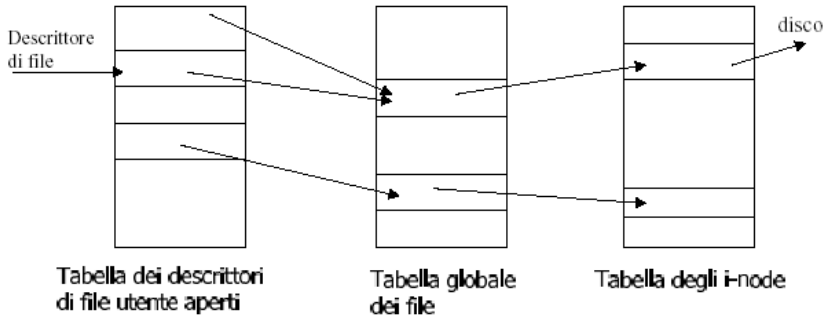
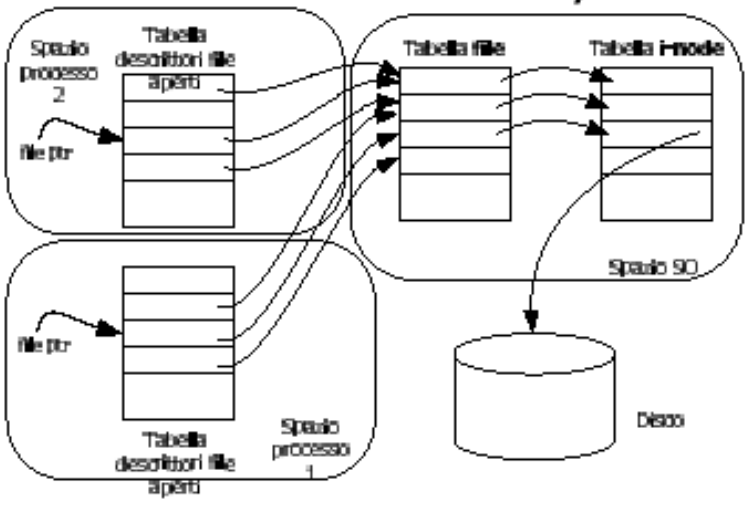


Tabelle Unix

- Tabella dei descrittori di file utente:
  - tabella associata ad ogni processo utente contenente una riga per ogni file aperto dal processo con l'indirizzo della riga della tabella globale dei file aperti relativa al file
- Tabella globale dei file aperti:
  - tabella del sistema operativo che contiene una riga per ogni file aperto nel sistema
  - Ogni riga contiene l'indirizzo del corrispondente i-node nella tabella degli i-node, l'indicatore della posizione corrente del file e il contatore al numero di riferimenti da parte dei processi a questo file
- Tabella degli i-node:
  - Le righe contengono la copia in memoria degli i-node del volume per maggiore efficienza nei riferimenti

Tabelle Unix

Strutture dati del file system



Contenuto volume

Blocco di boot	Super block	Lista i-node	Blocchi dati

- Blocco 0 o blocco di boostrap: contenuto tipicamente nel primo settore contiene il codice di inizializzazione del sistema operativo
- Superblock: descrive lo stato del file system (es. dimensioni, spazio libero…)
- Lista degli i-node: dimensione definita in fase di configurazione del sistema operativo, accessibile dalla tabella degli i-node

Il FS Ext2

Il FS Extended 2 (ext2) è quello più diffuso in Linux, quello che viene usato normalmente per formattare le partizioni in cui viene installato il sistema.

Principali caratteristiche di **ext2**:

Dimensione massima del filesystem	4 TB
Dimensione massima dei files	2 GB
Lunghezza massima dei nomi	255 caratteri
Fast symbolic links	Si
Supporta ctime, mtime e atime	Si
Spazio riservato per root	Si
Attributi estesi dei files	Si
Parametri modificabili	Si

Come tutti i filesystem unix anche l'ext2 è organizzato in una struttura comprendente super_block, inodes, directory e files.

Super-Block

Lo spazio all'interno di un filesystem unix è organizzato logicamente come un array lineare di blocchi di dimensioni uguali.

Dimensione blocco logico: indipendente dalla dimensione dei blocchi del dispositivo fisico - di solito di 1024 bytes, ma può essere fissata anche in 2048 o 4096 bytes.

Dimensione blocco fisico: HD è di 512 bytes, quindi un blocco logico occupa di solito 2 blocchi fisici.

Il fs ext2 è descritto da un blocco particolare detto **super_block**
è memorizzato in posizione fissa all'inizio del fs stesso e ne descrive le caratteristiche, dimensioni, struttura, stato ecc.

Super-Block

ext2_super_block è descritto dalla seguente struttura:

```
struct ext2_super_block {
        __u32   s_inodes_count;         /* Inodes count */
        __u32   s_blocks_count;         /* Blocks count */
        __u32   s_r_blocks_count;       /* Reserved blocks count */
        __u32   s_free_blocks_count;    /* Free blocks count */
        __u32   s_free_inodes_count;    /* Free inodes count */
        __u32   s_first_data_block;     /* First Data Block */
        __u32   s_log_block_size;       /* Block size */
        __s32   s_log_frag_size;        /* Fragment size */
        __u32   s_blocks_per_group;     /* # Blocks per group */
        __u32   s_frags_per_group;      /* # Fragments per group */
        __u32   s_inodes_per_group;     /* # Inodes per group */
        __u32   s_mtime;                /* Mount time */
        __u32   s_wtime;                /* Write time */
        __u16   s_mnt_count;            /* Mount count */
        __s16   s_max_mnt_count;        /* Maximal mount count */
        __u16   s_magic;                /* Magic signature */
        __u16   s_state;                /* File system state */
        __u16   s_errors;               /* Behaviour when detecting
errors */
        __u16   s_pad;
        __u32   s_lastcheck;            /* time of last check */
        __u32   s_checkinterval;        /* max. time between checks */
        __u32   s_creator_os;           /* OS */
        __u32   s_rev_level;            /* Revision level */
        __u16   s_def_resuid;           /* Default uid for reserved
blocks */
        __u16   s_def_resgid;           /* Default gid for reserved
blocks */
        __u32   s_reserved[235];        /* Padding to the end of the
block */
};
```

Super-Block

Sono contenuti tre tipi fondamentali di informazioni:

- **Caratteristiche del filesystem** (dimensioni, struttura, magic number, ecc.) stabilite al momento della creazione e non possono più essere modificate, per es. la **dimensione dei blocchi**, il **n.ro di blocchi**, il **n.ro di inodes**.

- **Parametri modificabili** (maximal mount count, error behaviour, ecc.) possono essere cambiati dal `superuser` con il programma **`tune2fs`**, che permette di variare ad es. il numero massimo di volte che il fs può essere rimontato prima di forzare un **`fsck`** automatico, oppure il comportamento in caso di errore. Un parametro modificabile è la % di spazio che può essere riservata ad un utente privilegiato, normalmente `root`. Questo consente al sistema di continuare a disporre di spazio su disco anche quando il fs si riempie per l'uso smodato degli utenti.

- **Variabili di stato** (file system state, free blocks, mount count, ecc.) descrivono invece lo stato del fs e vengono aggiornate automaticamente man mano che il fs viene usato, per esempio il numero di blocchi liberi, lo stato (montato o meno), il numero di volte che esso è stato montato, ecc.

Super-Block

Manca informazione sul tipo e sulla struttura fisica del device o della partizione in cui esso è ospitato. Il fs è totalmente indipendente da queste informazioni, che sono importanti per poter accedere fisicamente alla partizione o al device.
Si demanda al **device driver** tutte le operazioni necessarie per accedere al singolo blocco logico. Boot record del dos:

```
struct msdos_boot_sector {
        __s8    ignored[3];     /* Boot strap short or near jump */
        __s8    system_id[8];   /* Name - can be used to special case
                                   partition manager volumes */
        __u8    sector_size[2]; /* bytes per logical sector */
        __u8    cluster_size;   /* sectors/cluster */
        __u16   reserved;       /* reserved sectors */
        __u8    fats;           /* number of FATs */
        __u8    dir_entries[2]; /* root directory entries */
        __u8    sectors[2];     /* number of sectors */
        __u8    media;          /* media code (unused) */
        __u16   fat_length;     /* sectors/FAT */
        __u16   secs_track;     /* sectors per track */
        __u16   heads;          /* number of heads */
        __u32   hidden;         /* hidden sectors (unused) */
        __u32   total_sect;     /* number of sectors (if sectors == 0)
*/
};
```

I Cylinder Groups

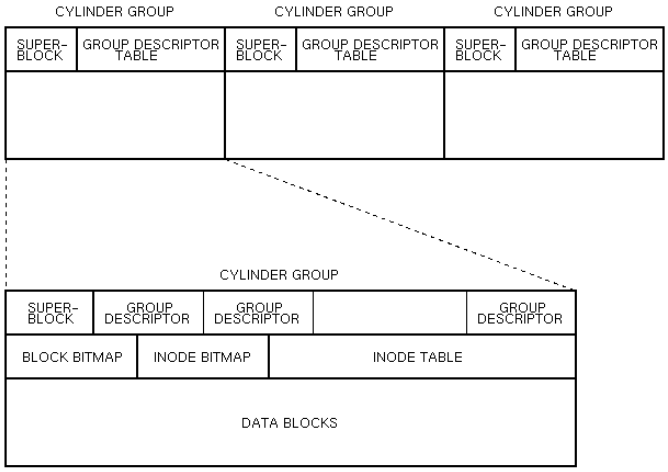
ext2 è suddiviso logicamente in più parti, dette **cylinder groups** i **cg** vengono gestiti come **entità separate** ed **autonome** pur facendo parte dello stesso fs.

Duplice scopo:
1. si vogliono minimizzare le conseguenze di eventuali errori, cioè fare sì che se alcuni dati di un **cg** risultano corrotti il danno resti limitato all'interno del **cg** stesso e non si propaghi a tutto il fs.

2. la tendenza a localizzare i files nell'intorno delle loro directory per ridurre i tempi di accesso, cosa che viene ottenuta cercando di allocare inodes e blocchi nello stesso cylinder group delle directory.

Per ridurre ulteriormente le possibilità che un intero fs venga corrotto a causa di eventuali errori, sia il **super_block** che le **group descriptor tables** vengono duplicati in ogni **cylinder group**, come mostrato in figura.

I Cylinder Groups



Se uno dei *super_block* o *group descriptor* viene corrotto a causa di errori esso può essere ripristinato a partire da una delle sue copie. Abbiamo quindi una struttura altamente ridondante che permette la *recovery* di eventuali errori.

I Cylinder Groups

Lo spazio all'interno del fs è gestito separatamente ed autonomamente per ciascun cylinder-group.

Ciascun cg contiene una ***block bitmap***, che indica quali blocchi del cg sono stati allocati a files o directories, una ***inode bitmap***, che indica analogamente quali inode risultano allocati, ed una ***inode table*** che contiene gli inodes appartenenti al cg.

Bitmap organizzata come una sequenza di bits:
ogni bit indica se il corrispondente inode o blocco è ***libero*** o ***occupato***.
ogni bitmap occupa esattamente un blocco e quindi il numero massimo di inodes o di blocchi che possono essere contenuti in un cg è data dalla dimensione in bytes di un blocco moltiplicata per 8.
Ad es. un fs con blocchi da **1K** $\Rightarrow$ 8192 (1024 * 8) blocchi per cg, poiché la ***block bitmap*** può contenere al max 8192 bits.

I Cylinder Groups

Posizione di un blocco o inode all'interno del fs a partire dal suo numero (nr):
- individuare il cylinder group: nr/num(oggetti per cg)
- individuare l'oggetto all'interno del cg utilizzando come indice il resto di tale divisione.

Esempio: volendo accedere all'inode **17654** di un fs con **1016** inodes per group si ottiene:

group = 17654 / 1016 = 17
index = 17654 % 1016 = 382

si deve cercare l'inode 382 del cylinder group 17.

Dal punto di vista dell'utente il filesystem è visto come un tutt'uno e la sua suddivisione in cylinder-groups è totalmente trasparente.

Group Descriptor

Ciascun cylinder-group è descritto da un apposito blocco di controllo detto **group descriptor**, che ha la seguente struttura:

```
struct ext2_group_desc {
        __u32   bg_block_bitmap;        /* Blocks bitmap block */
        __u32   bg_inode_bitmap;        /* Inodes bitmap block */
        __u32   bg_inode_table;         /* Inodes table block */
        __u16   bg_free_blocks_count;   /* Free blocks count */
        __u16   bg_free_inodes_count;   /* Free inodes count */
        __u16   bg_used_dirs_count;     /* Directories count */
        __u16   bg_pad;
        __u32   bg_reserved[3];
};
```

Il group descriptor contiene, oltre ai puntatori alle bitmap di allocazione ed alla tabella degli inodes, anche dei contatori (**bg_free_blocks_count, bg_free_inodes_count e bg_used_dirs_count**) che sono aggiornati ed utilizzati dalle routines di allocazione del fs.

File: attributi

A seconda del sistema operativo, i file possono avere attributi diversi.

Solitamente:
- **tipo**: stabilisce l'appartenenza a una classe (eseguibili, batch, testo, etc)
- **indirizzo**: puntatore/i a memoria secondaria
- **dimensione**: numero di byte contenuti nel file
- **data e ora** (di creazione e/o di modifica)

In S.O. multiutente anche:
- utente **proprietario**
- **protezione**: diritti di accesso al file per gli utenti del sistema

inode

Ad ogni file è associata una piccola tabella, detta **inode** ("index-node"), contenente

gli **attributi** del file
gli **indirizzi** dei primi blocchi del disco su cui è memorizzato il file
l'indirizzo di un **blocco a singola indirezione** contenente gli indirizzi di ulteriori blocchi di dati su disco
l'indirizzo di un **blocco a doppia indirezione** contenente gli indirizzi di blocchi a singola indirezione

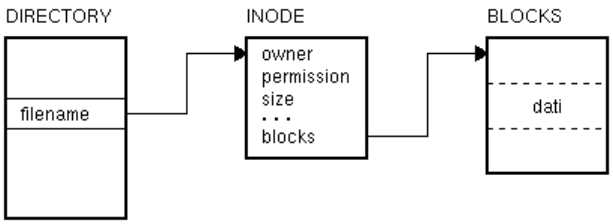
Attraverso l'i-node, il sistema operativo può rintracciare il file nel file system
Ogni i-node è identificato da un **i-number**

inode

L'inode è la risorsa principale di un fs unix.

Ad ogni **file** o **directory** è associato **univocamente** un **inode** che ne identifica le caratteristiche ed indica dove sono memorizzati fisicamente i dati.

Tutte le operazioni su un file o una directory vengono effettuate tramite il suo inode, che contiene tutte le informazioni sul file stesso, esclusi i dati veri e propri, come sintetizzato dalla seguente figura:



directory

E' un file come tutti gli altri, con l'unica differenza che **i dati in esso contenuti sono le informazioni sui files nella directory**, e viene pertanto gestito in modo particolare dal fs e dal kernel.

Ciascuna **entry** di directory è:

un record di lunghezza variabile allineata alla word (4 bytes) contente solamente il nome del file ed il suo numero di inode.

Tutte le altre informazioni sul file non hanno niente a che fare con la directory in cui esso è contenuto e sono pertanto memorizzate nel suo inode.

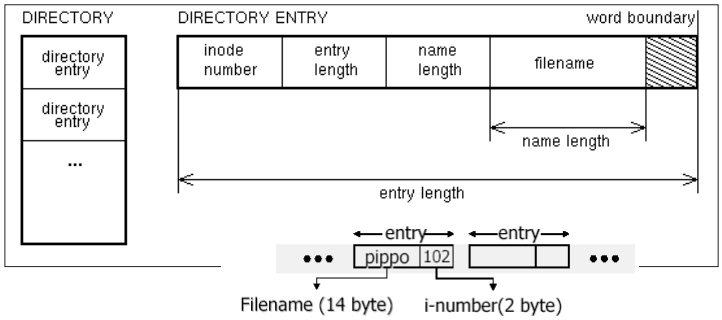
La directory serve solo a collegare

il nome del file col suo inode.

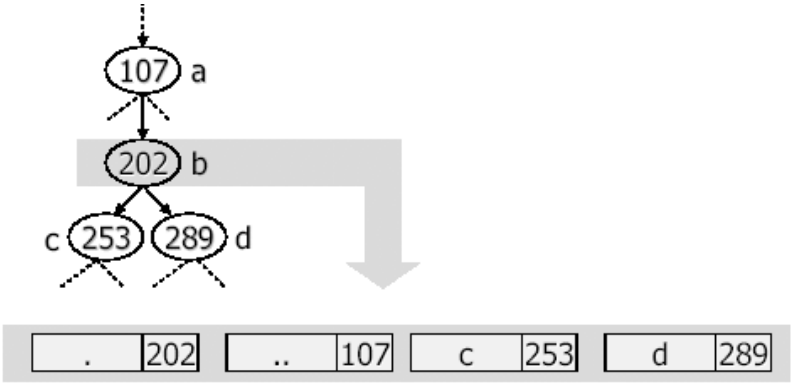
directory

La directory entry di ext2 è descritta dalla seguente struttura:

```
#define EXT2_NAME_LEN 255
struct ext2_dir_entry {
        __u32   inode;                  /* Inode number */
        __u16   rec_len;            /* Directory entry length */
        __u16   name_len;                /* Name length */
        char    name[EXT2_NAME_LEN];    /* File name */
};
```



directory



Ogni directory ha almeno 2 entry:
- "." la directory stessa
- ".." la directory padre

Inode: contenuto

Gli inodes sono numerati progressivamente da 1 in su ed il loro numero è l'informazione utilizzata dal kernel per accedere all'inode e quindi ai dati del file corrispondente.

La struttura di un inode di un fs ext2 è la seguente:

```
struct ext2_inode {
        __u16   i_mode;         /* File mode */
        __u16   i_uid;          /* Owner Uid */
        __u32   i_size;         /* Size in bytes */
        __u32   i_atime;        /* Access time */
        __u32   i_ctime;        /* Creation time */
        __u32   i_mtime;        /* Modification time */
        __u32   i_dtime;        /* Deletion Time */
        __u16   i_gid;          /* Group Id */
        __u16   i_links_count;  /* Links count */
        __u32   i_blocks;       /* Blocks count */
        __u32   i_flags;        /* File flags */
        ...
};
```

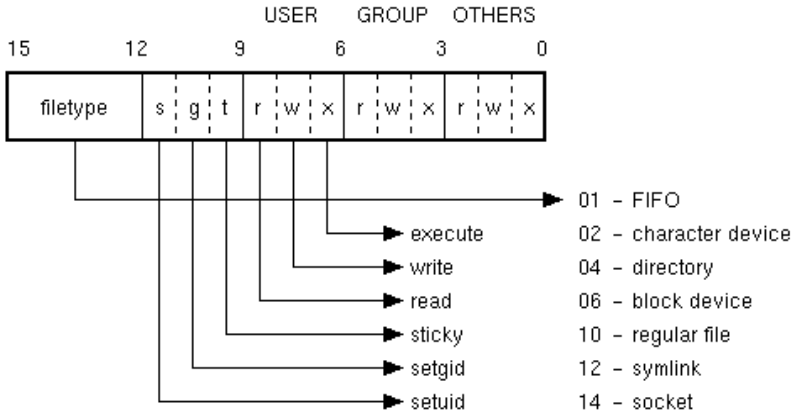
Inode: contenuto

Gli attributi contenuti bell'i-node di un file sono:

- **Tipo**: ordinario, directory, speciale
- **Posizione**: dove si trova
- **Dimensione**: quanto è grande
- **Numero di links**: quanti nomi ha
- **Permessi**: chi può usarlo e come
- **Creazione**: quando è stato creato
- **Modifica**: quando è stato modificato di recente
- **Accesso**: quando è stato l'accesso più recente

i-mode

Flag di 16 bit che memorizza i permessi di accesso ed esecuzione associati al file.



i-uid; i-gig; i-size

I campi ***i_uid*** e ***i_gig*** contengono l'userid ed il groupid del proprietario del file, rappresentati come interi di 16 bit.

Il campo ***i_size*** contiene la dimensione in bytes del file oppure, nel caso si tratti di un device, un numero di ***16 bit*** che specifica il **major** e il **minor number** del device stesso.

Es: le partizioni del primo harddisk ide hanno major number 3 e minor number che va da 1 a 9:

```
$ ls -l /dev/hda[1-9]
  brw-rw----   1 root     disk    3,  1 Apr 28  1995 /dev/hda1
  brw-rw----   1 root     disk    3,  2 Apr 28  1995 /dev/hda2
  brw-rw----   1 root     disk    3,  3 Apr 28  1995 /dev/hda3
  brw-rw----   1 root     disk    3,  4 Apr 28  1995 /dev/hda4
  brw-rw----   1 root     disk    3,  5 Apr 28  1995 /dev/hda5
  brw-rw----   1 root     disk    3,  6 Apr 28  1995 /dev/hda6
  brw-rw----   1 root     disk    3,  7 Apr 28  1995 /dev/hda7
  brw-rw----   1 root     disk    3,  8 Apr 28  1995 /dev/hda8
  brw-rw----   1 root     disk    3,  9 Apr 28  1995 /dev/hda9
```

i-atime; i-ctime; i-mtime; i-dtime

I quattro campi ***i_atime***, ***i_ctime***, ***i_mtime***, e ***i_dtime*** contengono rispettivamente i tempi di

ultimo accesso, creazione, modifica e cancellazione del file,

espressi in secondi dall'1-1-1970 secondo lo standard unix.

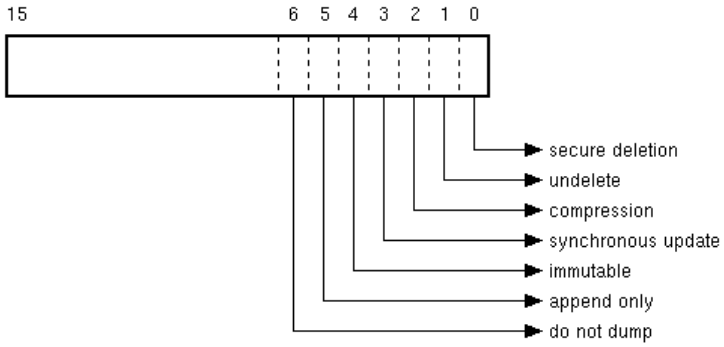
i-links_count; i-blocks

Il campo ***i_links_count*** contiene il numero di hard-links che puntano all'inode. Il suo valore è di solito 1, ma può essere maggiore di 1 se vengono creati degli ulteriori hard-links all'inode.

Il campo ***i_blocks*** indica il numero di blocchi occupati dal file, compresi anche i blocchi di indirizzamento indiretto che verranno illustrati più avanti.

i-flags

Il campo ***i_flags*** contiene dei flags aggiuntivi che indicano particolari caratteristiche del file:



Fast Symbolic Link

link simbolici ad un file: utilizzare un file con un certo *path* per fare in realtà riferimento ad un file che sta da un'altra parte, al limite su un altro disco o su un altro computer.
link fisici: non sono altro che diverse entry di directory che puntano allo stesso inode.

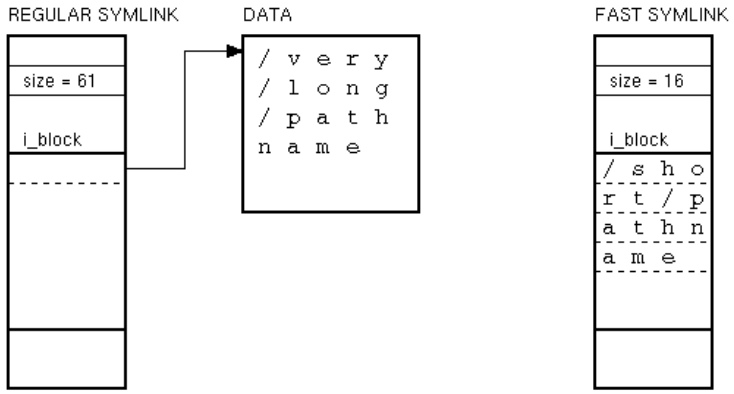
Un link simbolico a un file è di solito ***implementato*** memorizzando il **pathname** del file destinazione in un blocco di dati associato **all'inode** del link stesso.

Fast Symbolic Link

In ext2 non è sempre così: se il pathname destinazione è lungo non più di 59 bytes esso viene memorizzato direttamente nell'array ***i_blocks*** al posto dei puntatori ai blocchi di dati (si risparmia l'allocazione al blocco di dati e l'accesso al disco necessario per leggere il pathname del target, dato che questo è già disponibile nell'inode.

I **symlinks** di questo tipo sono per questo motivo chiamati ***Fast Symbolic Link*** e ciascuno di essi occupa un inode ma nessun blocco di dati.
Se invece il *pathname* del target è più lungo di **60 bytes** deve essere allocato un blocco di dati come per un file normale.
Nella figura seguente sono rappresentati i due tipi di symlink:

Fast Symbolic Link



Mount e Umount

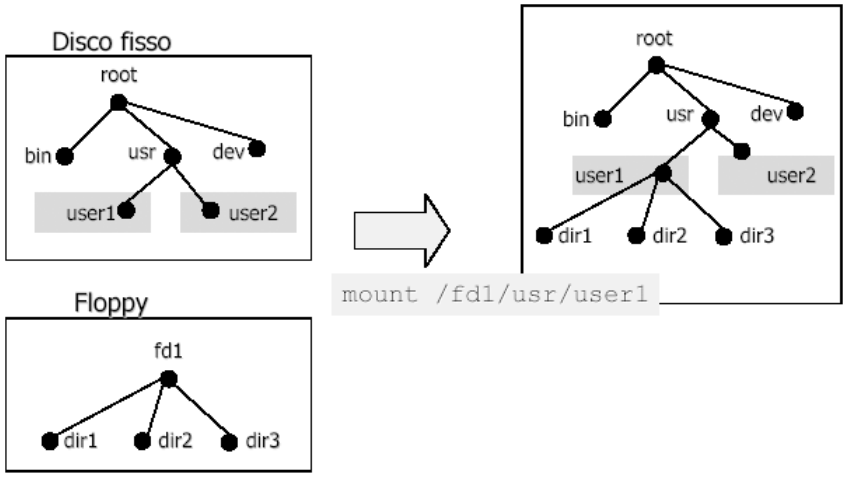
Un file system Unix è sempre **unico**, ma può avere parti residenti su device rimuovibili (es. dischetti)

Queste parti devono essere:
• "montate" prima di potervi accedere (**mount**)
• "smontate" prima di rimuovere il supporto (**umount**)

La system call **mount** ha due argomenti:
• **nome di un file ordinario**
• **nome di un file speciale** relativo ad un device removibile contenente un file system (fd1, fd2)

Dopo la mount non c'è più distinzione tra il file system presente sul disco rimovibile e quello presente sul disco fisso
La stessa tecnica si usa per suddividere il file system fra diversi device, anche se non rimuovibili

Effetto della Mount



/dev

Nome	Descrizione	Dos
/dev/fd0	prima unità a dischetti	A:
/dev/fd1	seconda unità a dischetti	B:
/dev/hda	primo disco fisso IDE/EIDE	
/dev/hdb	secondo disco fisso (o CD-ROM) IDE/EIDE	
/dev/hdc	terzo disco fisso (o CD-ROM) IDE/EIDE	
…		
/dev/sda	primo disco SCSI	
/dev/sdb	secondo disco SCSI	
…		

mount

Per poter accedere a dati contenuti su una diversa partizione del proprio disco bisogna **montare** il filesystem in quello che in gergo si chiama **mountpoint**, che poi non è altro che una comune directory.

Per identificare le partizioni Linux usa degli speciali file situati nella directory **/dev**.
Digitando il comando: **ls /dev/hd***
otterremo una lunga lista di file con nomi simili a hda1, hdb1.

mount

Creiamo una directory che servirà come

mount-point:

di solito le directory adibite a questo compito si trovano in `/mnt` ma nulla vieta di crearla altrove.

mkdir /mnt/discoibm

Con il comando **mount /dev/hda1 /mnt/discoibm** si monterà la partizione hda1 sotto la directory /mnt/discoibm.

Per vedere tutto quella che c'è in questa partizione basterà digitare il comando **ls -lAF /mnt/discoibm**

mount

Per fare in modo di non dover digitare il comando **mount** ad ognio avvio del O.S.:

Per fare ciò si può agire in tre modi:
1- inserendo la riga **insmod vfat** all'intemo del file **/etc/rc.d/rc.local**
2- editiamo il file **/etc/modules.conf**
3- Oppure ci si può affidare alle utility apposite sviluppate dalle vane distnibuzioni (COAS di OpenLinux 2.2, Lisa di OpenLinux 1.3, YaST di SuSE, Linuxconf di Red Hat).

Editandoo il file **/etc/fstab** verrà visualizzata una struttura del tipo:

Mounting del fs

- mount [<opz>] [<dev>] [<dir>]

-a *(utilizza /etc/fstab per montarli tutti)*
-t [no]<tipo-di-fs> [,…]
-o <opz-specifiche-di-fs> [,…]
es.
```
mount -t vfat /dev/fd0 /floppy
```

- umount [<opz>] [<dev>] [<dir>]

es.
```
umount /mount/cdrom
```

/etc/fstab

```
# nome Innesto Tipo Opzioni Dump Check
/dev/hda3 /         ext2  defaults 1         1
/dev/hdb1 /home ext2  defaults 0         2
proc /proc          proc defaults  0         0
/dev/hda2           none swap    sw
/dev/hda1 /mnt/dosc vfat quiet,umask=000 0 0
/dev/sda /mnt/dosd vfat user,noauto,quiet 0 0
/dev/sda1 /mnt/scsimo ext2 user,noauto 0 0
/dev/cdrom /mnt/cdrom iso9660 ro,user,noauto 0 0
roggen.brot.dg:/ /mnt/roggen nfs ro,user,noauto 0 0
/dev/fd0 /mnt/dosa vfat user,noauto,quiet 0 0
```

L'elenco dei filesystem attualmente in uso è contenuto in `/etc/mtab`