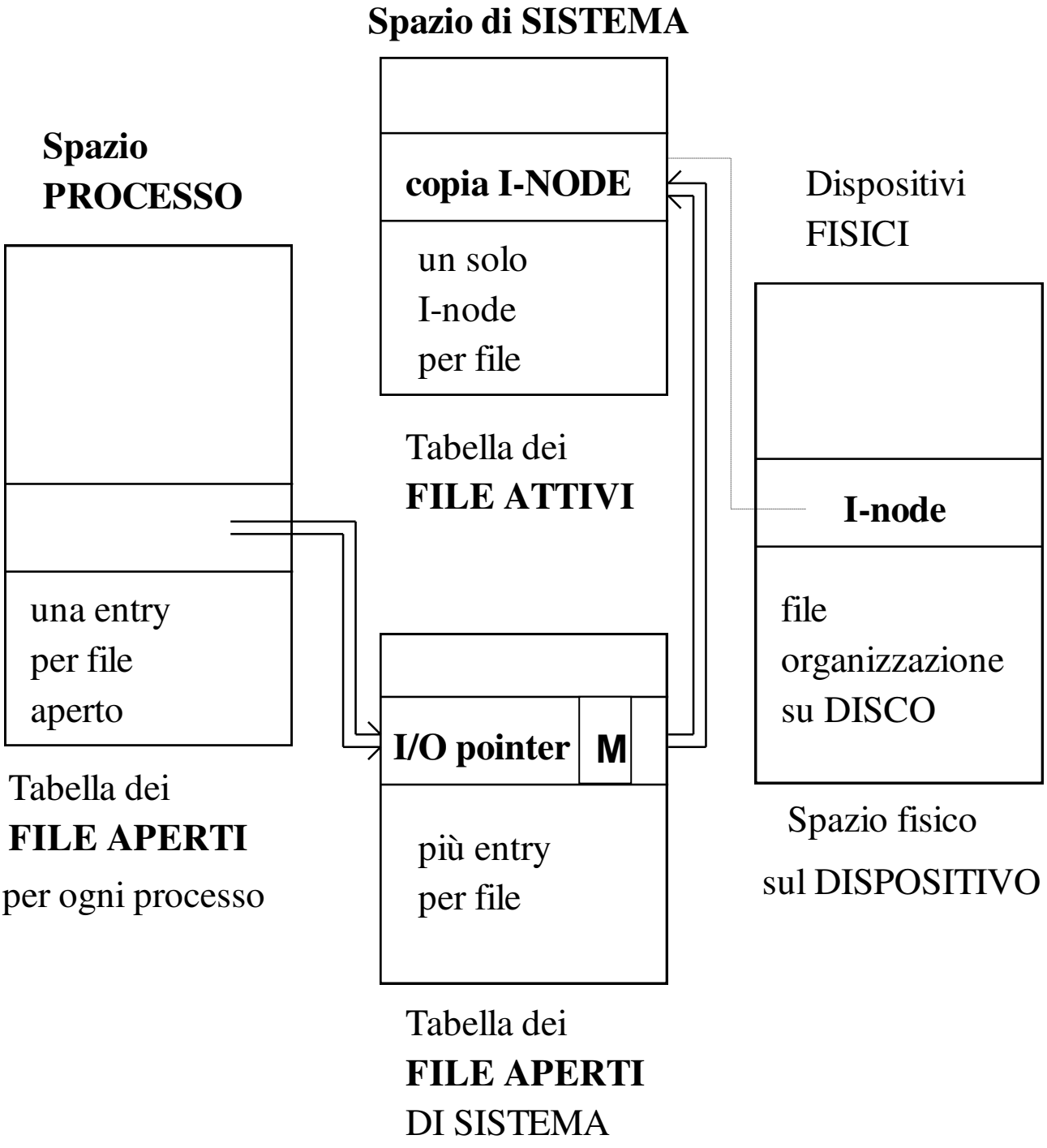
TABELLE DI SISTEMA PER ACCEDERE AI FILE UNIX (punto di vista interno)



Oltre ad una tabella dei FILE APERTI per ogni processo, a livello di sistema (KERNEL) ci sono **2 tabelle globali:**

- una **tabella di FILE APERTI di sistema** e
- una **tabella dei FILE (o I_NODE) ATTIVI**

(segue TABELLE DI SISTEMA - FILE UNIX)

Tabella dei file aperti di processo ⇨ un elemento (entry) significativo per ogni sessione di apertura di file per quel processo (fd ➔ indice nella tabella)
Sono possibili più entry che fanno riferimento allo stesso file

Ogni elemento di questa tabella contiene un puntatore al corrispondente elemento nella tabella dei file aperti di sistema

Tabelle globali

Tabella file attivi (**i-node attivi**) ⇨ un numero di elementi (entry) significativi pari al numero di file attivi
Ogni elemento di questa tabella contiene la ***COPIA*** dell'I-NODE (memorizzato in memoria secondaria) del file attivo

Tabella dei file aperti di sistema ⇨ un elemento (entry) significativo per ogni sessione di apertura di file
Sono quindi possibili più entry che fanno riferimento allo stesso file
Ogni elemento di questa tabella contiene:

- un puntatore al corrispondente elemento nella tabella dei file attivi (copia **i-node)**;
- un puntatore (**I/0 pointer**) che punta al byte "corrente" del file a partire dal quale verranno effettuate le operazioni;
- la modalità di apertura con cui è stata effettuata l'apertura del file.

L'apertura di un file (system call open()) provoca:

- l'inserimento di un elemento (individuato da un file descriptor) nella prima posizione libera della Tabella dei file aperti del processo
- l'inserimento di un'entry nella Tabella file aperti di sistema
- la copia del suo i-node nella tabella dei file attivi (se il file non è già stato aperto da qualunque processo, anche il medesimo)

STRUTTURA DI UN FILE SYSTEM FISICO

Un disco (o una sua partizione) viene suddiviso ("formattato") in parti di dimensione fissa composti da **blocchi fisici** di lunghezza fissa ⇨ in UNIX, da 512 a 4/8 kbyte

blocco 0	**boot-block**
blocco 1	**"superblock"** (descrive il resto del disco)
blocchi da 2 a n	**"i-list"** (lista degli **i-node**)
blocchi da n+1 a N-1	**blocchi occupati** dai dati e blocchi **liberi**

BOOT-BLOCK ===> contiene un programma di boot che serve per fare il boot (significativo solo per un FS fisico)

SUPERBLOCK ===> descrive lo stato del file system: la sua grandezza (in blocchi), quanti file può contenere, etc. Inoltre, contiene DUE LISTE: la lista dei numeri di BLOCCHI LIBERI e la lista degli i-number LIBERI

Il superblock viene COPIATO in memoria CENTRALE ===> riduce il tempo per allocare lo spazio per memorizzare i file

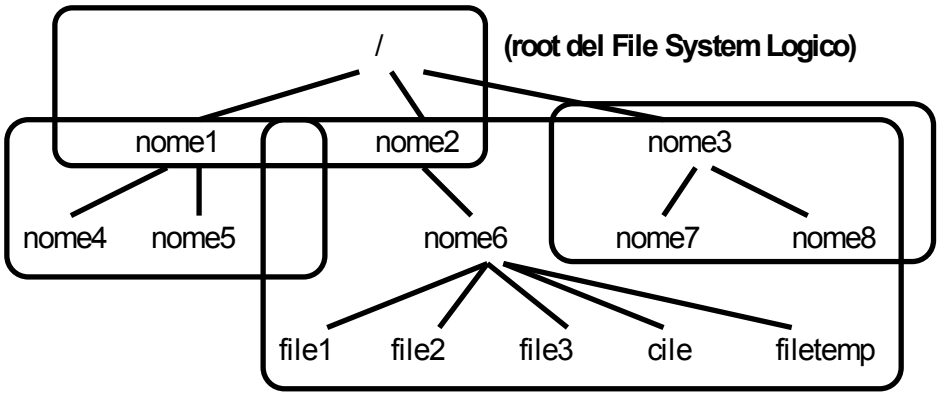
I-LIST ===> lista degli I-NODE
gli i-node sono numerati da 1 in su

BLOCCHI ===> sia blocchi allocati che blocchi liberi

Ogni File System Fisico deve essere **MONTATO** (comando/primitiva ***mount***) nel File System LOGICO che presenta una unica radice (*root* ⇨ */*) ⇨ comando **df** per vedere i File System fisici montati (con il punto di mount)

FILE SYSTEM LOGICO

Un File System Logico è in generale composto da più File System Fisici: uno di partenza (quello che contiene la root del File System Logico) e altri che devono essere montati in modo che la loro root sia un nodo e quindi una directory del grafo totale



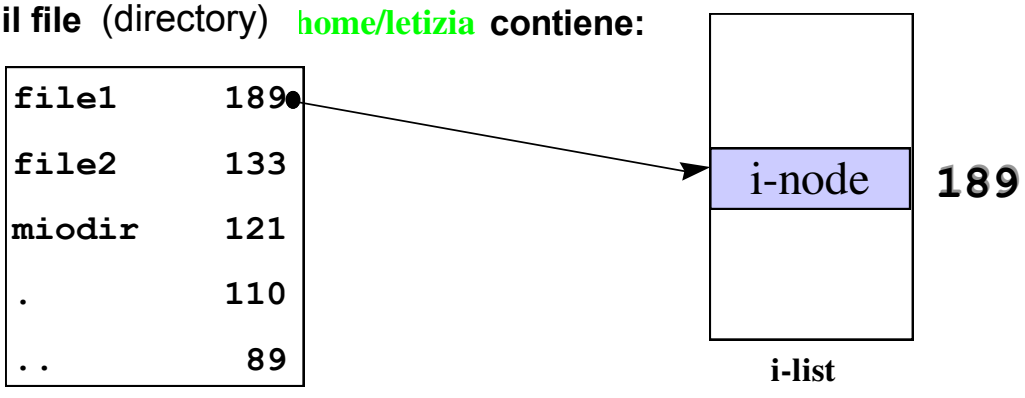
Partendo quindi dalla radice del File System Logico, la locazione fisica di ogni file **F** viene recuperata tramite le informazioni contenute nel suo I-NODE, cui si arriva tramite l'I-NUMBER contenuto nella directory ove è memorizzato il nome relativo semplice del file **F**

⇨ la numerazione degli I-NODE è unica per ogni File System FISICO!!!

In una directory per ogni file abbiamo:

nomefile (relativo semplice)	**i-number**

il file (directory) **home/letizia** **contiene:**



I-NODE

Rivediamone la struttura

per file *normali/directory*

- ***tipo*** del file:
(ordinario, directoryo o special file);

- i ***bit*** SUID, SGID, e 'sticky';

- ***diritti*** read/write/execute per utente, gruppo e altri;

- numero dei ***link*** del file;

- ***identificatori*** user, group (UID e GID);

- ***dimensione*** del file

- ***tempi di accesso*** (lettura, scrittura file e I-node)

- ***indirizzi*** di tredici blocchi ⌫ **NOTA BENE**
per recuperare i blocchi di dati

per i ***dispositivi***

numero IDentificativo:
numero maggiore driver di dispositivo
(per una tabella di configurazione)
numero minore quale dispositivo

classe del dispositivo
a blocchi / a caratteri

RITROVARE i blocchi fisici del FILE

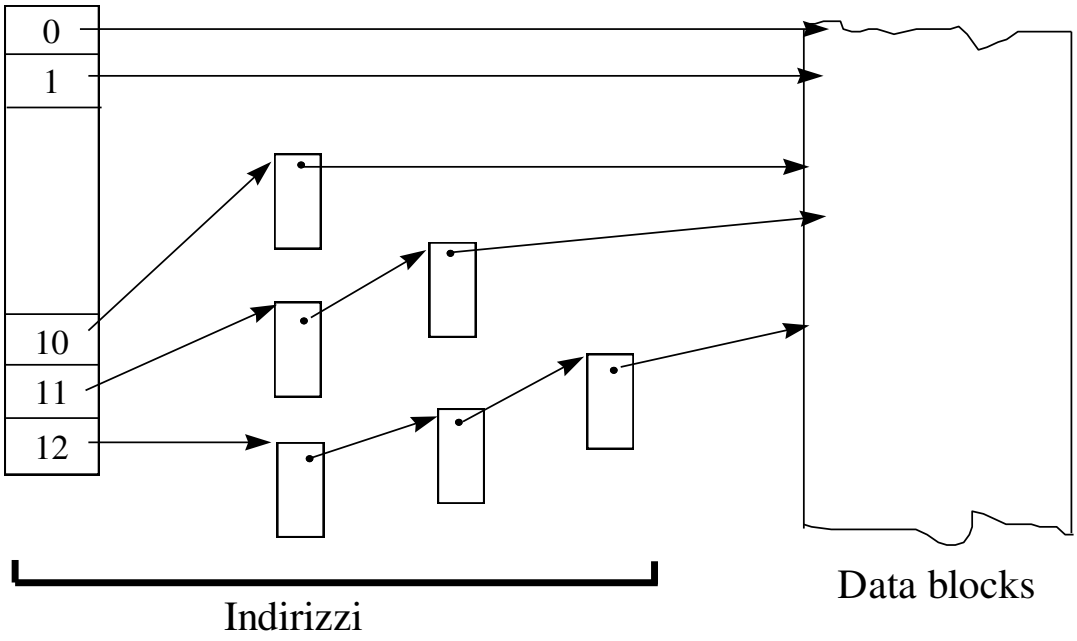
I primi dieci indirizzi **diretti**
(**Ad esempio**: se un blocco è di 512 byte ⇨
Lunghezza massima file: *10*512 byte = 5 KB)*

L'***undicesimo*** indirizzo è **indiretto**
cioè l'indirizzo di un blocco che contiene gli indirizzi di altri blocchi
(**Segue esempio**: se gli indirizzi sono di 4 byte ⇨ in un blocco,128 indirizzi di blocchi dati: *128 * 512 byte = 64 KB*
Lunghezza massima file: *5 + 64 = 69 KB)*

Il ***dodicesimo*** indirizzo comporta **due livelli** di indirettezza:
(**Segue esempio**: 128 indirizzi di blocchi di indirizzi a blocchi dati: *128 * 128 * 512 byte = 8 MB*
Lunghezza massima file: *69 KB + 8 MB*)

Il ***tredicesimo*** indirizzo comporta **tre livelli** di indirettezza fino a Gbyte:
(**Segue esempio**: 128 indirizzi di blocchi di indirizzi a blocchi di indirizzi: *128 * 128 * 128 * 512 byte = 4 GB*
Lunghezza massima file: *69 KB + 8 MB + 4 GB*)

Si favoriscono file di **media lunghezza**
File di dimensioni teoricamente **illimitata**



La primitiva SYNC

UNIX, per velocizzare le operazioni sui dischi, mantiene in **memoria centrale**

* una copia dei SUPERBLOCK di tutti i file system montati nel sistema

* dei buffer per le operazioni di I/O (read/write)

PROBLEMA:

Le informazioni su disco possono risultare non aggiornate rispetto a quelle mantenute in memoria
===> se il sistema va in crash (ad esempio, manca la corrente) la situazione sui dischi potrebbe essere non corretta

SOLUZIONE:

Periodicamente (in genere, 30 secondi),
UNIX chiama una primitiva che fa il FLUSH di tutte le informazioni in memoria centrale sui dischi

SYNC sync();

Questa operazione viene SEMPRE eseguita quando si effettua lo shutdown di un sistema UNIX

Kernel di Linux (versione 2.0)

La struttura files_struct presente nel descrittore di un processo contiene, in particolare, la **tabella dei file aperti**

```
struct files_struct {
        ...
/* Maschera di bit di tutti i file descriptor usati (fino a 256) */
        fd_set open_fds;
/* Array di file descriptor */
        struct file * fd[NR_OPEN];
};
```

dove (in fs.h)

```
#define NR_OPEN 256
```

L'array **fd** è la **tabella dei file aperti del processo**: ogni elemento è un puntatore alla seguente struttura (ancora in fs.h):

```
struct file {
/* modalità di accesso (read only, read+write, write only) */
        mode_t f_mode;
/* file pointer all'interno del file (64 bit ⇨ file > di 2 Giga) */
        loff_t f_pos;
/* ulteriori flag di open */
        unsigned short f_flags;
/* contatore di riferimenti */
        unsigned short f_count;
/* puntatore alla copia dell'i-node del file */
        struct inode * f_inode;
        ...
};
```

segue Kernel di Linux (versione 2.0)

Vediamo ora la struttura **inode** (sempre in fs.h):

```
struct inode {
/* descrizione del dispositivo (partizione) */
        kdev_t          i_dev;
/* numero dell'inode all'interno del dispositivo */
        unsigned long   i_ino;
/* diritti di accesso */
        umode_t         i_mode;
/* numero di hard link */
        nlink_t         i_nlink;
/* user id e group id del proprietario */
        uid_t           i_uid;
        gid_t           i_gid;
/* dimensione (in byte) */
        off_t           i_size;
/* tempo di ultimo accesso, modifica, modifica dell'i-node */
        time_t          i_atime;
        time_t          i_mtime;
        time_t          i_ctime;
        .....
/* informazioni per lo specifico file system */
        union {
                struct minix_inode_info minix_i;
                struct ext_inode_info ext_i;
                struct ext2_inode_info ext2_i;
                struct hpfs_inode_info hpfs_i;
                struct msdos_inode_info msdos_i;
                ...
        } u;
};
```

segue Kernel di Linux (versione 2.0)

Consideriamo il caso di fs ext2 (file Ext2_fs.h)

```
/* numero di blocchi indirizzati direttamente */
#define    EXT2_NDIR_BLOCKS    12

/* indici dei puntatori ai blocchi che contengono indirizzi dei blocchi indiretti */
#define    EXT2_IND_BLOCK EXT2_NDIR_BLOCKS
#define    EXT2_DIND_BLOCK      (EXT2_IND_BLOCK + 1)
#define    EXT2_TIND_BLOCK (EXT2_DIND_BLOCK + 1)

/* numero di elementi della tabella dei riferimenti ai blocchi */
#define    EXT2_N_BLOCKS    (EXT2_TIND_BLOCK + 1)

struct ext2_inode {
        ...
/* indirizzi dei blocchi */
        __u32      i_block[EXT2_N_BLOCKS];
        ...
};
```

Tabella i_block

Posizione	Contenuto
0 → (EXT2_NDIR_BLOCKS -1)	12 riferimenti diretti
EXT2_IND_BLOCK	Riferimento indiretto 1° livello
EXT2_DIND_BLOCK	Riferimento indiretto 2° livello
EXT2_TIND_BLOCK	Riferimento indiretto 3° livello