***File & Directory***

*st\_mode* nella struttura ***stat*** include anche 9 bit che regolano i permessi di accesso al file:

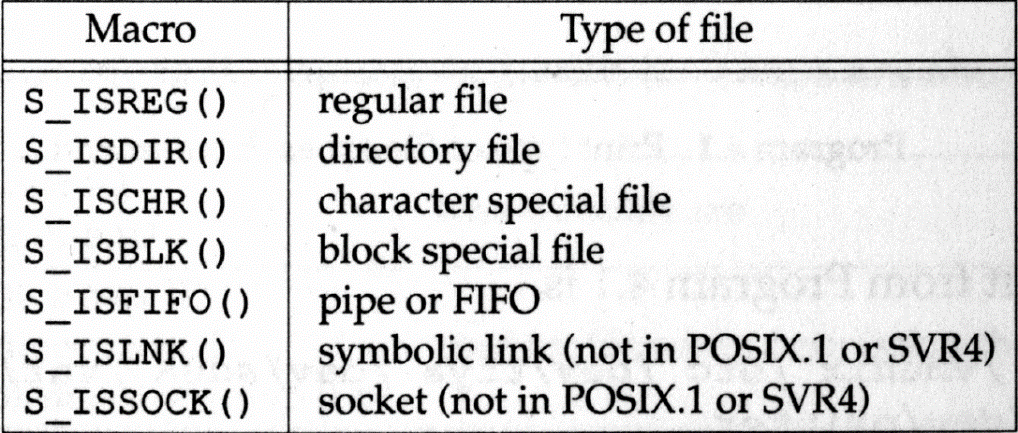
**PRELEVARE INFORMAZIONI DA FILE:**

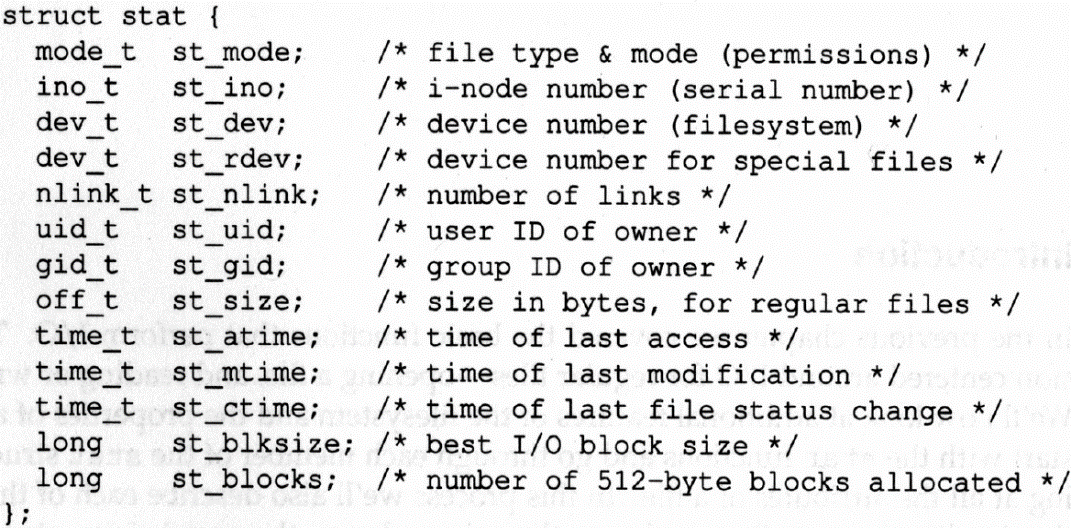
* int ***stat*** (const char \**pathname*, struct stat \**buf*);
* fornisce una struttura di info relative al file del primo argomento.
* int ***fstat*** (int *fd*, struct stat \**buf*);
* come prima, ma il file cui si riferisce è già aperto e quindi prende come argomento il file descriptor.
* int ***lstat*** (const char \**pathname*, struct stat \**buf*);
* le info ottenute sono relative al link simbolico (e non al file a cui esso si riferisce)

*Descrizione*: danno informazioni sul file 1° argomento, bisogna fornire un puntatore ad una struttura (chiamata “stat”) che viene poi riempita durante lo svolgimento della funzione.

*Restituiscono*: 0 se OK, -1 in caso di errore.

Le ***macro*** seguenti sono *funzioni booleane* che aiutano ad identificare il tipo di un file verificando ciò che è contenuto nel campo *st\_mode* della struttura stat del file:





Il campo *st\_uid (st\_gid)* della struttura stat contiene l’ID dell’utente (gruppo) possessore del file.

Ogni processo ha degli ID associati: real u/g ID, effective u/g ID, saved set-u/g-ID.

* ***real*** : chi siamo realmente, presi dal file /etc/passwd al login time;
* ***effective*** : determina i permessi di accesso ai file;
* ***saved set*** : contengono copie dell'effective quando è eseguito un programma (exec).

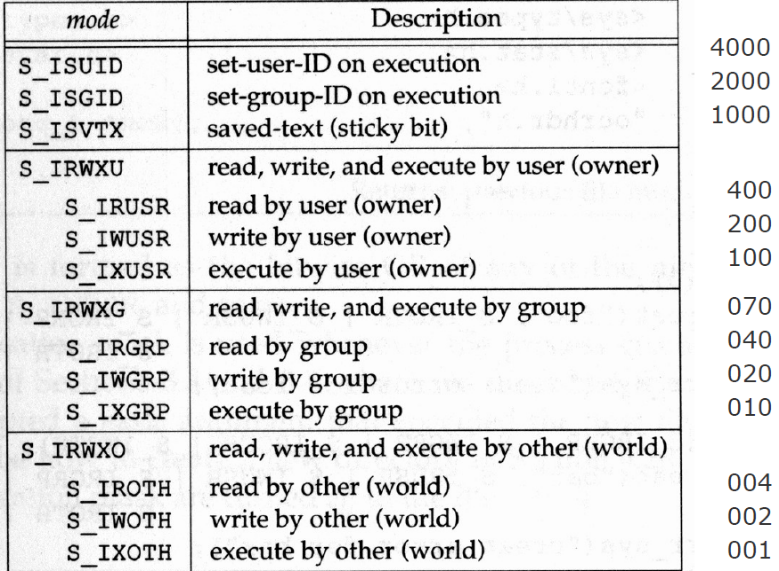
**MODIFICA DEI PERMESSI AI FILE:**

* int ***access*** (const char \**pathname*, int *mode*);

***Descrizione***: verifica se il real ID ha accesso al file 1°argomento nella modalità specificata da mode.

***Restituisce***: 0 se OK, -1 in caso di errore.

* int ***chmod*** (const char \**pathname*, mode\_t *mode*);
* int ***fchmod*** (int *fd,* mode\_t *mode*);



***Descrizione***: cambiano i bit di permesso del file 1° argomento.

***Restituiscono***: 0 se OK, -1 in caso di errore.

**CAMBIARE PROPRIETARIO AI FILE:**

* int ***chown*** (const char \**pathname*, uid\_t *owner*, gid\_t *group*);
* int ***fchown*** (int *fd*, uid\_t *owner*, gid\_t *group*);
* int ***lchown*** (const char \**pathname*, uid\_t *owner*, gid\_t *group*);

*Descrizione*: cambiano il proprietario ed il gruppo del file 1° argomento e li settano uguale a owner e group.

*Restituiscono*: 0 se OK, -1 in caso di errore.

***i-nodo*** ha un contatore(*st\_nlink*) di *link*(detti *hard link*) che contiene il num di *directory entry* che lo puntano e contiene tutte le *info* che riguardano il file:

* tipo del file;
* bit di permesso;
* size del file;
* puntatori ai blocchi di dati del file.

***directory block***: è una lista di record aventi almeno due campi:

* numero dell'i-nodo
* nome del file

**MODIFICA DEI LINK:**

* int ***link*** (const char \**path*, const char \**newpath*);

*Descrizione*: crea una nuova *directory entry* *newpath* che si riferisce a path, crea un hard link aggiuntivo (con la corrispondente directory entry) ad un file esistente incrementa anche di uno il contatore dei link *st\_nlink*.

*Restituisce*: 0 se OK, -1 in caso di errore (anche se *newpath* già esiste).

* int ***unlink*** (const char \**pathname*);

*Descrizione*: rimuove la directory entry specificata da pathname e se pathname è un hard link allora decrementa il contatore dei link del file cui il link si riferisce.

*Restituisce*: 0 se OK, -1 in caso di errore.

* è consentita se si ha il permesso di scrittura ed esecuzione nella directory dove è presente la *directory entry*, solo il *superuser* può rimuovere un *hard link* ad una *directory*
* int ***remove*** (const char \**pathname*);

*Descrizione*: rimuove il file specificato dal *pathname*.

*Restituisce*: 0 se OK, -1 in caso di errore.

* int ***rename*** (const char \**oldname*, const char \**newname*);

*Descrizione*: assegna un nuovo nome newname ad un file od ad una directory data come 1° argomento.

*Restituisce*: 0 se OK, -1 in caso di errore.

***link simbolici,*** sono dei puntatori indiretti ad un file.

* int ***symlink*** (const char \**path*, const char \**sympath*);

*Descrizione*: crea un link simbolico sympath che punta a path.

*Restituisce*: 0 se OK, -1 altrimenti.

* int ***readlink*** (const char \**pathname*, char \**buf*, int *bufsize*);

*Descrizione*: legge dal link simbolico 1° argomento e ne mette il contenuto in buf la cui taglia è bufsize, legge il contenuto del link e non del file cui esso si riferisce.

*Restituisce*: il numero di byte letti se OK, -1 in caso di errore.

**MODIFICA DEGLI ACCESSI AI LINK:**

* int ***mkdir*** (const char \**pathname*, mode\_t *mode*);

*Descrizione*: crea una directory i cui permessi di accesso vengono determinati da mode e dalla mode creation mask del processo.

*Restituisce*: 0 se OK, -1 in caso di errore.

* int ***rmdir*** (const char \**pathname*);

*Descrizione*: viene decrementato il numero di link al suo i-node, se esso =0 si libera la memoria solo se nessun processo ha quella directory aperta.

*Restituisce*: 0 se OK, -1 in caso di errore.

* int ***chdir*** (const char \**pathname*);
* int ***fchdir*** (int *fd*);

*Descrizione*: cambiano la *cwd* del processo chiamante a quella specificata come argomento.

*Restituiscono*: 0 se OK, -1 in caso di errore.

* char \****getcwd*** (char \**buf*, size\_t *size*);

*Descrizione*: ottiene in buf il path assoluto della cwd.

*Restituisce*: buf se OK, NULL in caso di errore.