Capitolo 10 **Interfaccia del file system**

Per la maggior parte degli utenti il file system è l’aspetto più visibile di un sistema operativo.

Esso fornisce il meccanismo per la registrazione e l’accesso in linea a dati e programmi appartenenti al sistema operativo e a tutti gli utenti del sistema di calcolo.

Il file system consiste di due parti distinte: un insieme di *file,* ciascuno dei quali contenente dati correlati, e una *struttura della directory*, che organizza tutti i file nel sistema e fornisce le informazioni relative.

**10.1 Concetto di file.**

I calcolatori possono memorizzare le informazioni su diversi supporti, come dischi, nastri magnetici e dischi ottici.

Il sistema operativo associa i file ai dispositivi fisici, di solito non volatili, in modo che il loro contenuto non vada perduto a causa delle interruzioni dell’alimentazione elettrica e dei riavvii del sistema.

Un file è un insieme di informazioni, correlate e registrate in memoria secondaria, cui è stato assegnato un nome. Dal punto di vista dell’utente, un file è la più piccola porzione di memoria secondaria logica; i dati si possono cioè scrivere in memoria secondaria soltanto all’interno di un file.

In genere un file è formato da una sequenza di bit, byte, righe o *record* il cui significato è definito dal creatore e dall’utente del file stesso.

Le informazioni contenute in un file sono definite dal suo creatore e possono essere di molti tipi: **programmi sorgente, programmi oggetto, dati numerici, testo, dati contabili, immagini, registrazioni sonore, e cosi via.**

**10.1.1 Attributi dei file**

Per comodità degli utenti, ogni file ha un nome che si usa come riferimento.

Una volta ricevuto il nome, il file diviene indipendente dal processo, dall’utente, e anche dal sistema da cui è stato creato. Un file ha altri attributi che possono variare secondo il sistema operativo, ma che tipicamente comprendono i seguenti.

♦ **Nome.** Il nome simbolico del file e l’unica informazione in forma umanamente leggibile.

**♦ Identificatore.** Si tratta di un’etichetta unica, di solito un numero, che identifica il file all’interno del file system; è il nome impiegato dal sistema per il file.

♦ **Tipo**. Questa informazione è necessaria ai sistemi che gestiscono tipi di file diversi.

**♦ Locazione.** Si tratta di un puntatore al dispositivo e alla locazione del file in tale dispositivo.

**♦ Dimensione.** Si tratta della dimensione corrente del file (in byte, parole o blocchi) ed eventualmente della massima dimensione consentita.

**♦ Protezione.** Le informazioni di controllo degli accessi controllano chi può leggere, scrivere o far eseguire il file.

**♦ Ora, data e identificazione dell’utente.** Queste informazioni possono essere relative alla creazione, l’ultima modifica e l’ultimo uso. Questi dati possono essere utili ai fini della protezione e per controllarne l’uso.

Le informazioni sui file sono conservate nella **struttura della directory**, che risiede a sua volta in memoria secondaria. Di solito un elemento di directory consiste di un nome di file e di un identificatore unico, che a sua volta individua gli altri attributi del file.

**10.1.2 Operazioni sui file**

Un file è un **tipo di dato astratto.** Per definire adeguatamente un file è necessario considerare le operazioni che si possono eseguire su di esso. Il sistema operativo può offrire chiamate di sistema per creare, scrivere, leggere, spostare, cancellare e troncare un file.

**♦ Creazione di un file.** Per creare un file è necessario compiere due passaggi. In primo luogo si deve trovare lo spazio per il file nel file system.

Secondariamente, per il file si deve creare un nuovo elemento nella directory in cui registrare il nome del file, la sua posizione nel file system ed eventualmente altre informazioni.

**♦ Scrittura di un file.** Per scrivere in un file è indispensabile una chiamata di sistema che specifichi il nome del file e le informazioni che si vogliono scrivere. Dato il nome del file, il sistema cerca la sua posizione nella directory. Il file system deve mantenere un puntatore di ***scrittura*** alla locazione nel file in cui deve avvenire l’operazione di scrittura successiva. Il puntatore si deve aggiornare ogniqualvolta si esegue una scrittura.

**♦ Lettura di un file.** Per leggere da un file è necessaria una chiamata di sistema che specifichi il nome del file e la posizione in memoria dove collocare il successivo blocco del file. Anche in questo caso si cerca l’elemento corrispondente nella directory e il sistema deve mantenere un puntatore di ***lettura*** alla locazione nel file in cui deve avvenire la successiva operazione di lettura. Una volta completata la lettura, si aggiorna il puntatore.

Di solito un processo legge o scrive in un file, e la posizione corrente è mantenuta come un **puntatore alla posizione corrente del file.** Sia le operazioni di lettura sia quelle di scrittura adoperano lo stesso puntatore, risparmiando spazio e riducendo la complessità del sistema.

**♦ Riposizionamento in un file.** Si ricerca l’elemento appropriato nella directory e si assegna un nuovo valore al puntatore alla posizione corrente nel file. Il riposizionamento non richiede operazione di l/O. Questa operazione è anche nota come ***posizionamento*** o ***ricerca (seek)*** nel file.

**♦ Cancellazione di un file.** Per cancellare un file si cerca l’elemento della directory associato al file designato, si rilascia lo spazio associato al file e si elimina l’elemento della directory.

**♦ Troncamento di un file.** Si potrebbe voler cancellare il contenuto di un file, ma mantenere i suoi attributi. Questa funzione consente di mantenere immutati gli attributi (a esclusione della lunghezza del file) pur azzerando la lunghezza del file e rilasciando lo spazio occupato.

Queste 6 operazioni di base comprendono **l’insieme minimo delle operazioni richieste per i file**. La maggior parte delle operazioni sopra citate richiede una ricerca dell’elemento associato al file specificato nella directory. Per evitare questa continua ricerca, molti sistemi richiedono l’impiego di una chiamata di sistema open ( ) la prima volta che si adopera un file in maniera attiva.

Il sistema operativo mantiene una piccola tabella contenente informazioni riguardanti tutti i file aperti (detta, per l’appunto, **tabella dei file aperti**)**.** Quando si richiede un’operazione su un file, questo viene individuato tramite un indice in tale tabella, in questo modo si evita qualsiasi ricerca. Quando il file non è più attivamente usato viene *chiuso* dal processo, e il sistema operativo rimuove l’elemento a esso associato dalla tabella dei file aperti. L’operazione

open ( ) riceve il nome del file, lo cerca nella directory e copia l’elemento a esso associato nella tabella dei file aperti.

La chiamata di sistema open ( ) riporta di solito un puntatore all’elemento nella tabella dei file aperti; questo puntatore si adopera al posto dell'effettivo nome del file in tutte le operazioni di I/O, evitando cosi successive operazioni di ricerca e semplificando l’interfaccia delle chiamate di sistema.

La realizzazione della open e close è più complicata nei sistemi multiutente quando più utenti possono accedere allo stesso file. Quindi il SO introduce due tabelle, una per ciascun processo e una per il sistema.

Ciascun elemento della tabella associata a ciascun processo punta a sua volta a una tabella

di sistema dei file aperti, contenente le informazioni indipendenti dai processi come la posizione dei file nei dischi, le date degli accessi e le dimensioni dei file.

In genere, la tabella dei file aperti ha anche un *contatore delle aperture* associato a ciascun file, indicante il numero di processi che hanno aperto quel file. Ogni close( ) decrementa questo *contatore*; quando raggiunge il valore zero il file non è più in uso e si elimina l’elemento corrispondente dalla tabella dei file aperti.

A ciascun file aperto sono associate le seguenti informazioni:

**♦ Puntatore al file.** Nei sistemi che non prevedono lo scostamento come parte delle chiamate di sistema read( ) e write ( ), il sistema deve tener traccia dell’ultima posizione di lettura e scrittura sotto forma di un puntatore alla posizione corrente nel file. Questo puntatore è unico per ogni processo che opera sul file e quindi deve essere tenuto separato dagli attributi del file residenti nel disco.

**♦ Contatore dei file aperti.** Questo contatore tiene traccia del numero di open ( ) e close ( ), e raggiunge il valore zero dopo l’ultima chiusura, momento in cui il sistema può rimuovere l’elemento della tabella.

**♦ Posizione nel disco del file.** La maggior parte delle operazioni richiede al sistema di modificare i dati contenuti nel file. L’informazione necessaria per localizzare il file nel disco è mantenuta in memoria, per evitare di doverla prelevare dal disco a ogni operazione.

**♦ Diritti d’accesso.** Ciascun processo apre un file in una delle modalità d’accesso. Questa informazione è contenuta nella tabella del processo in modo che il sistema operativo possa permettere o negare le successive richieste di I/O.

Quando un processo intende proteggere un file dall’accesso concorrente di altri processi, si serve dei **lock**. L’utilità dei lock dei file emerge nel caso di file condivisi da diversi processi: un file di log, per esempio, può subire modifiche da parte di molti processi.

Un **lock condiviso** è assimilabile, per funzionamento, ai lock di lettura: entrambi consentono a più processi concorrenti di appropriarsene. Un **lock esclusivo** mostra invece analogie con i lock di scrittura, perchè un solo processo per volta può acquisire questo tipo di lock. Si noti bene che non in tutti i sistemi operativi è possibile scegliere tra i due tipi di lock; alcuni sistemi forniscono solamente lock esclusivi dei file.

Inoltre, il sistema operativo può fornire meccanismi di protezione dei file **obbligatori,** oppure **consigliati.**

Se un lock è obbligatorio, il sistema operativo impedirà a qualunque altroprocesso di accedere al file interessato.

Qualora invece il lock sia solo consigliato, il sistema operativo non impedirà l’accesso dell’editor a system. log. Tuttavia, per poter accedere al file, l’editor deve essere scritto in modo tale da acquisire esplicitamente il lock.

**10.1.3 Tipi di file**

Nella progettazione di un file system si può considerare la possibilità di riconoscere e quindi gestire più specificatamente i tipi di file.

Una tecnica comune per realizzare la gestione dei tipi di file consiste nell’includere il tipo nel nome del file.

Il nome è suddiviso in due parti, un nome e un’ *estensione,* di solito separate da un punto ; in questo modo l’utente e il sistema operativo possono risalire al tipo del file semplicemente esaminandone il nome. Il sistema usa l’estensione per stabilire il *tipo* del file e le operazioni che si possono eseguire su tale file.

**10.1.4 Struttura dei file**

**10.1.5 Struttura interna dei file**

Uno scostamento all'interno di un file può essere complicato da gestire per l'OS.

Nei dischi la dimensione dei blocchi è ben definita e dipende dalla dimensione di un settore.

La dimensione del record fisico invece non è prestabilita ed una soluzione diffusa per gestirlo all'interno di un blocco consiste nell'**impaccamento** di un certo numero di file in blocchi.

La dimensione dei record logici, quella dei blocchi fisici e la tecnica d'impaccamento, determinano il numero dei record logici all'interno di ogni blocco fisico.

Poichè lo spazio nel disco è sempre assegnato in blocchi, una parte del blocco di ogni file in genere è sprecata; maggiore è la dimensione dei blocchi, maggiore è la frammentazione interna.

**10.2 Metodi d'accesso**

I file memorizzano informazioni , e al momento dell’uso è necessario accedere a queste informazioni e trasferirle in memoria.

**10.2.1 Accesso sequenziale**

Il più semplice metodo d’accesso è **l’accesso sequenziale**: le informazioni del file si elaborano ordinatamente, un record dopo l’altro; questo metodo d’accesso è di gran lunga il più comune, ed è usato, ad esempio, dagli editor e dai compilatori.

Le più comuni operazioni che si compiono sui file sono le letture e le scritture: un’operazione di lettura legge la prima porzione e fa avanzare automaticamente il puntatore del file che tiene traccia della locazione di I/O; analogamente, un’operazione di scrittura fa un’aggiunta in coda al file e avanza fino alla fine delle informazioni appena scritte, che costituisce la nuova fine del file.

**10.2.2 Accesso diretto**

Un altro metodo è **l’accesso diretto** (o accesso relativo). Un file è formato da elementi logici (record) di lunghezza fissa; ciò consente ai programmi di leggere e scrivere rapidamente tali elementi senza un ordine particolare.

Il file si considera come una sequenza numerata di blocchi o record che si possono leggere o scrivere in modo arbitrario.

I file ad accesso diretto sono molto utili quando è necessario accedere immediatamente a grandi quantità di informazioni.

Il numero del blocco fornito dall’utente al sistema operativo è normalmente un **numero di blocco relativo**.

Si tratta di un indice relativo all’inizio del file, quindi il primo blocco relativo del file è 0.

L’uso dei numeri di blocco relativi permette al sistema operativo di decidere dove posizionare il file e aiuta a impedire che l’utente acceda a porzioni del file system che possono non far parte del suo file. Alcuni sistemi iniziano la numerazione dei blocchi relativi da 0, altri da 1.

10.2.3 Si possono costruire sulla base dei precedenti altri metodi d'accesso più complessi.

**10.3 Struttura della directory e del disco**

Un dispositivo di memorizzazione può essere interamente utilizzato per un file system, ma può anche essere suddiviso per un controllo più raffinato. Un disco può ad esempio essere **partizionato** e ogni partizione può contenere un file system. I dispositivi di memorizzazione possono essere raccolti in insiemi RAID che proteggono dal fallimento di un singolo disco.

La suddivisione in partizioni è utile anche per limitare la dimensione dei file system individuali, per mettere sullo stesso dispositivo tipi diversi di file system, oppure per liberare ad altri scopi una parte del dispositivo, come nel caso dello spazio di swap *(avvicendamento)* o dello spazio su disco non formattato **(raw).**

Le partizioni sono note anche come **suddivisioni** *(slice)* o **minidischi** (nel mondo **IBM).**

Un file system può essere installato su ciascuna di queste parti del disco. Ogni entità contenente un file system è generalmente nota come **volume.**

Il volume può essere un sottoinsieme di dispositivi, un dispositivo intero o dispositivi multipli collegati in **RAID.** Ogni volume può essere pensato come un disco virtuale.

I volumi possono inoltre contenere diversi sistemi operativi, permettendo al sistema di avviare ed eseguire più di un sistema operativo.

Ogni volume contenente un file system deve anche avere in sè le informazioni sui file presenti nel sistema. Tali informazioni risiedono in una **directory del dispositivo** o **indice** **del volume.**

**10.3.2 Generalità sulla directory**

La directory si può considerare come una tabella di simboli che traduce i nomi dei file negli elementi in essa contenuti. Da questo punto di vista si capisce che la stessa directory si può organizzare in molti modi diversi.

Deve essere possibile inserire nuovi elementi , cancellarne di esistenti e cercare un elemento.

* **Ricerca di un file.** Deve esserci la possibilità di scorrere una directory per individuare l’elemento associato a un particolare file
* **Creazione di un file.** Deve essere possibile creare nuovi file e aggiungerli alla directory
* **Cancellazione di un file.** Quando un file non serve più deve essere possibile rimuoverlo.
* **Elencazione di una directory.** Deve esistere la possibilità di elencare tutti i file di una directory e il contenuto degli elementi della directory associati ai rispettivi file nell’elenco
* **Ridenominazione di un file.** Poichè un nome di un file rappresenta per i suoi utenti il contenuto del file, questo nome deve poter essere cambiato
* **Attraversamento del file system.** Si potrebbe voler accedere a ogni directory e a ciascun file contenuto di una directory. Per motivi di affidabilità è opportuno salvare il contenuto e la struttura dell'intero file system a intervalli regolari

**10.3.3 Directory a livello singolo**

La struttura più semplice per una directory è quella a **livello singolo**.

Tutti i file sono contenuti nella stessa directory, facilmente gestibile e comprensibile.

Una directory a livello singolo presenta però limiti notevoli che si manifestano all’aumentare del numero dei file in essa contenuti, oppure se il sistema è usato da più utenti. Poichè si trovano tutti nella stessa directory, i file devono avere nomi unici; se due utenti attribuiscono lo stesso nome al loro file di dati si viola la regola del nome unico.

Anche per un solo utente, con una directory a livello singolo, con l’aumentare del loro numero diventa difficile ricordare i nomi dei file.

**10.3.4 Directory a due livelli**

La **directory a due livelli** è la soluzione più diretta per creare una separazione fra utenti. In questo

modo infatti non si pone il problema dell'esistenza di file omonimi, in quanto se presenti in directory diverse (ognuna corrispondente quindi a un utente diverso) non entrano in conflitto.

La struttura prevede una directory principale (*master file directory*) a cui fanno riferimento tutte le sottodirectory figlie (a cui possiamo rispettivamente associare un utente).

Ipotizzando di vedere questo tipo di directory come un albero, la sua radice è la *master file directory*, i suoi diretti discendenti sono le directory d'utente e le foglie sono i file all'interno di ognisottodirectory (introduzione del concetto di *path name*, nome utente/nome file).

Per gestire la divisione in directory si necessita della presenza dei file di sistema con tutte le informazioni necessarie in ogni sottodirectory.

La struttura della directory a due livelli presenta ancora dei problemi. In effetti questa struttura isola un utente dagli altri. Questo isolamento può essere un vantaggio quando gli utenti sono completamente indipendenti ma uno svantaggio quando gli utenti vogliono cooperare e accedere a file di altri utenti .

Un altro problema si riferisce al fatto delle chiamate di sistema e i file di sistema che per non sprecare spazio nel copiare ogni file di sistema in ciascun utente, si definisce una speciale directory utente contente i file di sistema. Ogni volta che si indica un file da caricare, il sistema operativo lo cerca innanzitutto nella directory utente locale e se lo trova, lo usa. Se non lo trova il sistema cerca automaticamente nella speciale directory utente contenente i file di sistema. La sequenza delle directory in cui è stata fatta la ricerca avviata dal riferimento a un file è detta percorso di ricerca .

**10.3.5 Directory con struttura ad albero**

La corrispondenza strutturale tra **directory a due livelli** e **albero a due livelli** permette di generalizzare facilmente il concetto.

Questa generalizzazione permette agli utenti di creare proprie sottodirectory e di organizzare i file di conseguenza. L’albero ha una directory radice (*root directory*), e ogni file del sistema ha un unico nome di percorso.

Una directory, o una sottodirectory, contiene un insieme di file o sottodirectory. Le directory sono semplicemente file, trattati però in modo speciale. La distinzione tra file e directory è data dal bit, rispettivamente 0 e 1.

Normalmente, ogni utente dispone di una directory corrente.

Quando si fa un riferimento a un file, si esegue una ricerca nella directory corrente; se il file non si trova in tale directory, l’utente deve specificare un nome di percorso oppure cambiare la directory corrente facendo diventare tale la directory contenente il file desiderato. Per cambiare directory corrente si fa uso di una chiamata di sistema che preleva un nome di directory come parametro e lo usa per ridefinire la directory corrente.

La directory corrente iniziale di un utente è stabilita all’avvio del lavoro d’elaborazione dell’utente. Nel file di contabilizzazione è memorizzato un puntatore alla (oppure il nome della) directory iniziale dell’utente. Tale puntatore viene copiato in una variabile locale per l’utente che specifica la sua directory corrente iniziale. Dalla shell dell’utente si possono poi avviare altri processi: la loro directory corrente è solitamente la directory corrente del processo genitore al momento della creazione del figlio.

**I** nomi di percorso possono essere di due tipi: **nomi di percorso assoluti** e **nomi di percorso relativi.**

Un *nome di percorso assoluto* comincia dalla radice dell’albero di directorye segue un percorso che lo porta fino al file specificato indicando i nomi delle directory checostruiscono le tappe del suo cammino.

Un *nome di percorso relativo* definisce un percorsoche parte dalla directory corrente.Una decisione importante relativa alla strutturazione ad albero delle directory riguardail modo di gestire la cancellazione di una directory. Se una directory è vuota, è sufficientecancellare l’elemento che la designa nella directory che la contiene. Tuttavia se la directoryda cancellare non è vuota, ma contiene file oppure sottodirectory, è possibile procedere indue modi. Alcuni sistemi, come l’MS-DOS, non cancellano una directory a meno che non sia vuota; per cancellarla l’utente deve prima cancellare i file in essa contenuti. Se esiste qualche sottodirectory, questa procedura si deve applicare anche alle sottodirectory. Questo metodo può richiedere una discreta quantità di lavoro.

In alternativa, come nel comando rm di UNIX, si può avere un’opzione che, alla richiesta di cancellazione di una directory, cancelli anche tutti i file e tutte le sottodirectory in essa contenuti.

**10.3.6 Directory con struttura a grafo aciclico**

La struttura ad albero non ammette la condivisione di file o directory.

Un **grafo aciclico** permette alle directory di avere sottodirectory e file condivisi. Lo stesso file o la stessa sottodirectory possono essere in due directory diverse. Un grafo aciclico, cioè senza cicli, rappresenta la generalizzazione naturale dello schema delle directory con struttura ad albero.

Il fatto che un file o una directory siano condivisi non significa che ci siano due copie, perciò tutte le modifiche sono immediatamente visibili ad ogni utente che vi accede.

Per riuscire a rendere questa cosa possibile i sistemi operativi usano i collegamenti (*link*), cioè concettualmente dei “puntatori” localizzati in punti diversi rispetto alla locazione del file o della directory originali, ma che conoscono il percorso per risalirvi e richiamarli anche da quella posizione.

Esistono 2 tipi di collegamenti utilizzati:

• **Link simbolici**: con questo criterio la locazione di ogni file o directory è unica, ma sono presenti dei file contenenti il percorso per risalire ad essi in ogni punto del sistema li si voglia richiamare. Con questo metodo la cancellazione dei file o delle directory originali fa perdere di significato ai vari collegamenti posizionati nei vari punti del sistema e la

rimozione di essi può diventare abbastanza onerosa. (E' il criterio che si segue nella famiglia dei sistemi Windows)

• **Hard link**: essenzialmente è un nuovo nome dello stesso file. Creare un **hard link** significa aggiungere un nuovo nome al file in modo tale da poter accedere ad esso da qualsiasi locazione in cui si trovi uno dei suoi nomi; materialmente viene aggiunto un nuovo elemento all'elenco di riferimenti al file, cioè al contatore degli *hard link* dal quale dipende la cancellazione del file stesso. Tale contatore infatti viene incrementato ad ogni nuova assegnazione di nome e permette l'eliminazione del file soltanto se esso è uguale a 0. (E' il criterio che si segue in Unix).

**10.3.7 Directory con struttura a grafo generale**

Aggiungendo nuovi file e nuove sottodirectory alla directory con struttura ad albero, la natura di quest'ultima persiste. Tuttavia, quando si aggiungono dei collegamenti a una directory con struttura ad albero, tale struttura si trasforma in una semplice struttura a grafo; ciò potrebbe causare dei problemi nel caso in cui a causa dei collegamenti alle directory, il grafo diventi ciclico, e il sistema vada in loop infinito durante l'attraversamento del file system.

Esistono dei modi per evitare questo problema:

• Permettere il collegamento di soli file e non directory

• Usare la *garbage collection*, cioè controllare l'accessibilità di tutti i file attraversando il file system (metodo notevolmente costoso)

• Usare algoritmi di controllo d'esistenza di un ciclo ad ogni creazione di *link* (metodo notevolmente costoso).

**10.4 Montaggio di un file system**

Cosi come si deve *aprire* un file per poterlo usare, per essere reso accessibile ai processi di un sistema, un file system deve essere *montato*. La struttura delle directory può ad esempio essere composta di volumi, che devono essere montati affinchè siano disponibili nello spazio dei nomi di un file system. La procedura di montaggio è molto semplice: si fornisce al sistema operativo il nome del dispositivo e la sua locazione (detta punto di montaggio) nella struttura di file e directory alla quale agganciare il file system.

Di solito, un punto di montaggio è una directory vuota cui sarà agganciato il file system che deve essere montato.

Il passo successivo consiste nella verifica da parte del sistema operativo della validità del file system contenuto nel dispositivo. La verifica si compie chiedendo al driver del dispositivo di leggere la directory di dispositivo e controllando che tale directory abbia il formato previsto. Infine, il sistema operativo annota nella sua struttura della directory che un certo file system è montato al punto di montaggio specificato.

**10.5 Condivisione di file**

Una volta che più utenti possono condividere file, l’obiettivo diventa estendere la condivisione a più file system. Infine ci possono essere diverse interpretazione delle azioni conflittuali su file condivisi

**10.5.1 Utenti multipli**

Se un SO permette l’uso del sistema da parte di più utenti, diventano particolarmente rilevanti i problemi relativi alla condivisione dei file, alla loro identificazione tramite nome e alla loro protezione. Il sistema può permettere a ogni utente di accedere ai file degli altri utenti oppure chiedere i permessi di accesso.

Per realizzare i meccanismi di condivisione e protezione, il sistema deve memorizzare e gestire gli attributi di directory e file. La maggior parte dei sistemi ha adottato i concetti di *proprietario, gruppo* e altri.

Il *proprietario* è l’utente che può cambiare gli attributi di un file, concedere l'accesso.

L’attributo *gruppo* invece si definisce il sottoinsieme di utenti autorizzati a condividere l’accesso. Gli identificatori del gruppo e del proprietario sono memorizzati insieme con altri attributi del file.

**10.5.2 File system remoti**

L’avvento delle reti ha permesso la comunicazione tra calcolatori separati dagrandi distanze.

I metodi con i quali i file si condividono in una rete sono cambiati molto, seguendo l’evoluzione della tecnologia delle reti e dei file.

Un primo metodo consiste nel trasferimento dei file richiesto in modo esplicito dagli utenti, attraverso programmi come l’**ftp**. Un secondo metodo, molto diffuso, è quello del file system distribuito *(distributed file system*, DFS), che permette la visibilità nel calcolatore locale delle directory remote. Il terzo metodo, il World Wide Web.

**10.5.2.1 Modello client-server**

I file system remoti permettono il montaggio di uno o più file system di uno o più calcolatori remoti in un calcolatore locale. Il calcolatore contenente i file si chiama server, mentre quelli che ricevono sono chiamati client. La relazione tra client e server è piuttosto comune tra i calcolatori in rete.

Il server in genere specifica i file disponibili su di un volume. Tra le soluzioni per l’autenticazione c'è quella dell’invio comune di chiavi cifrate. Nel caso di UNIX e del suo file system di rete NFS, l’autenticazione avviene tramite le informazioni di connessione relative al client.

I protocolli NFS permettono relazioni da molti a molti, cioè più server possono fornire dati a più client. Una volta montato il file system remoto, le richieste delle operazioni su file sono inviate al server usando il protocollo DFS.

Per il montaggio può applicare la stessa semantica dei file system locali. Nella richiesta di apertura di un file, il client invia il nome utente e la richiesta che verrà esaminata dal server per gestire i permessi.

**10.5.2.2 Sistemi informativi distribuiti**

Per semplificare la gestione dei servizi client-server i sistemi informativi distribuiti sono stati concepiti per porre un accesso unificato alle informazioni necessarie per il calcolo remoto. Il sistema dei nomi di dominio DNS fornisce le traduzioni dai nomi dei calcolatori agli indirizzi di rete per il web. Altri sistemi informativi distribuiti forniscono uno spazio identificato da nomeutente/pwd/identificatoreutente/idenitficatoregruppo/ per un servizio distribuito.

Nel caso delle reti microsoft CIFS le informazioni di rete si usano insieme con gli elementi di autenticazione dell’utente per creare un nome utente di rete che il server usa per decidere se permettere o negare l'accesso. L’industria si sta orientando verso il protocollo LDAP come meccanismo sicuro per la comunicazione distribuita.

Se il protocollo avrà successo, un’organizzazione potrà usare una singola dir LDAP per memorizzare le informazioni su tutti gli utenti e le risorse di tutti i calcolatori.

**10.5.2.3 Malfunzionamenti**

I file system locali possono presentare malfunzionamenti per varie cause: \*\*problemi dei dischi che li contengono, alterazione dei dati relativi alle strutture delle directory o a informazioni necessarie alla gestione dei dischi, malfunzionamenti dei controllori dei dischi.\*\*

Molte cause portano al crollo dei sistema, all’emissione di una condizione d’errore e alla necessità di un intervento umano per risolver il problema. L’uso di file system remoti implica maggior possibilità di malfunzionamenti. Nel caso delle reti si possono verificare interruzioni del collegamento tra due calcolatori, dovute a malfunzionamenti o a improprie configurazioni. Nel caso di un crollo del server o a un malfunzionamento della rete si porta all’inaccessibilità del file system remoto. Invece di comportarsi come la perdita di un file system locale, deve provvedere a terminare tutte le operazioni sul server e attendere che la connessione ritorni. Per mantener questo tipo di recupero dal malfunzionamento è necessario tenere delle informazioni sullo stato dei client e dei server.

**10.5.3 Semantica della coerenza**

La *semantica della coerenza* è un importante criterio per la valutazione di qualsiasi file system che consenta la condivisone dei file.

* **Semantica unix.** Le scritture in un file aperto da parte di un utente sono immediatamente visibili ad altri utenti che hanno aperto contemporaneamente lo stesso file.
* **Semantica delle sessioni.** Le scritture in un file aperto da un utente sono visibili immediatamente ad altri utenti ma una volta chiuso le modifiche sono visibili solo agli utenti che apriranno il file successivamente.

**Semantica dei file condivisi immutabili.** Un altro metodo è quello dei file condivisi immutabile: una volta che un file è stato dichiarato condiviso, non può essere modificato.

**10.6 Protezione**

La salvaguardia delle informazioni contenute in un sistema di calcolo da i danni fisici e da accessi impropri è fondamentale. Generalmente l’affidabilità è assicurata da più copie dei file. Molti calcolatori hanno programmi di sistema che programmano a intervalli regolari la creazione di copie che potrebbero essere cancellati per sbaglio o a causa di danneggiamenti dei dischi.

**10.6.1 Tipi d'accesso**

I sistemi che non permettono l’accesso ai file di altri utenti non richiedono protezione; quindi si può ottenere una completa protezione proibendo l’accesso. In alternativa si può permettere un accesso totalmente libero senza alcuna protezione. Questi orientamenti sono entrambi eccessivi, quindi non si possono applicare in generale; ciò che serve in realtà è un **accesso controllato.**

Gli accessi si permettono o si negano secondo diversi fattori, innanzitutto i tipi d’accesso richiesti. Si possono controllare distinte operazioni:

**♦ Lettura.** Lettura da file.

**♦ Scrittura.** Scrittura o riscrittura di file.

**♦ Esecuzione.** Caricamento di file in memoria ed esecuzione.

**♦ Aggiunta.** Scrittura di nuove informazioni in coda ai file.

♦ **Cancellazione.** Cancellazione di file e liberazione del relativo spazio per un possibile

riutilizzo.

**♦ Elencazione.** Elencazione del nome e degli attributi dei file.

**10.6.2 Controllo degli accessi**

Il problema della protezione comune è il controllo degli accessi. Lo schema generale per realizzare tale lista è quello dell’ACL a ogni file e dir ; in tale lista sono specificati i nomi degli utenti e i relativi tipi di accesso consentiti.

Questo ha il vantaggio di permettere complessi metodi d’accesso ma ha lo svantaggio della costruzione di una lista per ogni file Tale compito è noioso. Per risolvere questo problema può essere introdotta una versione diversa della lista dove vengono specificati il **proprietario, il gruppo e l’universo**. Affinché questo schema funzioni correttamente è necessario uno stretto controllo dei permessi e delle liste di controllo.

**10.6.3 Altri metodi di protezione**

Un altro metodo consiste nell’associare una parola chiave, password, a ciascun file. Tener un buon livello di sicurezza delle password consiste nell’avere password complesse e che cambino ogni tot di tempo. Questo presenta un grande svantaggio che è quello di memorizzare numerose password.