Sistemi Operativi Modulo I

Primo canale (A-L) e Teledidattica

A.A. 2021/2022

Corso di Laurea in Informatica

Il File System

Igor Melatti

Sapienza Università di Roma
Dipartimento di Informatica

Roadmap

- Visione d'insieme
- Le directory
- Gestione della memoria secondaria
- Gestione dei file in UNIX
- Gestione dei file su Windows

I File(s)

- Sono l'elemento principale per la maggior parte delle applicazioni
  - molto spesso, l'input di un'applicazione è un file; quasi altrettanto spesso, l'output è un file
  - i file "sopravvivono" ai processi
- Il file system è una delle parti del sistema operativo che sono più importanti per l'utente
- Proprietà desiderabili:
  - esistenza a lungo termine
  - condivisibilità con altri processi (tramite nome simbolici)
  - strutturabilità (directory gerarchiche)

Gestione dei File

- I file sono gestiti da un insieme di programmi e librerie di utilità
- Tali programmi costituiscono il File (Management) System, e vengono eseguiti come processi privilegiati (kernel mode)
- Le librerie, invece, vengono invocate come system call (sempre kernel mode)
- Hanno a che fare con la memoria secondaria (dischi, chiavi USB, ...)
  - in Linux, anche in RAM
- Forniscono un'astrazione sotto forma di operazioni tipiche
- Per ogni file vengono mantenuti degli attributi (o metadati), come proprietario, data di creazione, etc

Operazioni Tipiche sui File

- Creazione (con annessa scelta del nome)
  - al momento della creazione, il file è tipicamente vuoto
- Cancellazione
- Apertura: necessaria per poter leggere e scrivere
- Lettura: solo su file aperti (e non chiusi nel frattempo)
- Scrittura: idem
- Chiusura: necessaria per le performance

Terminologia

- Campo (*field*)
- Record
- File
- Database

Campi e Record

- Campi:
  - dati di base
  - contengono valori singoli
  - caratterizzati da lunghezza e tipo di dato (o con demarcazioni)
  - esempio tipico: carattere ASCII
- Record
  - insiemi di campi correlati
  - ognuno trattato come un'unità
  - esempio tipico: un impiegato è caratterizzato dal record nome, cognome, matricola, stipendio

File e Database

- File:
  - hanno un nome
  - insiemi di record correlati
    - nei SO generici moderni, ogni record è un solo campo con un byte...
  - ognuno trattato come un'unità con nome proprio
  - possono implementare meccanismi di controllo dell'accesso (alcuni utenti possono accedere ad alcuni file, altri ad altri)
- Database
  - collezioni di dati correlati
  - mantengono anche relazioni tra gli elementi memorizzati
  - realizzati con uno o più file
  - ci sono i DBMS, che sono tipicamente processi di un SO

Sistemi per la Gestione di File

- File Management Systems
- Forniscono servizi agli utenti e alle applicazioni per l'uso di file
  - e definiscono anche il modo in cui i file sono usati
- Sollevano i programmatori dal dover scrivere codice per gestire i file

Obiettivi per i File Management Systems

- Rispondere alle necessità degli utenti riguardo alla gestione dei dati (creazione etc)
- Garantire che i dati nei file sono validi
- Ottimizzare le prestazioni
  - sia dal punto di vista del SO (throughput) che dell'utente (tempo di risposta)
- Fornire supporto per diversi tipi di memoria secondaria
  - dischi magnetici, chiavi USB, CD, DVD...
- Minimizzare i dati persi o distrutti
- Fornire un insieme di interfacce standard per i processi utente
- Fornire supporto per l'I/O effettuato da più utenti in contemporanea

Requisiti per i File Management Systems

1. Ogni utente dev'essere in grado di creare, cancellare, leggere, scrivere e modificare un file
2. Ogni utente deve poter accedere, in modo controllato, ai file di un altro utente
3. Ogni utente deve poter leggere e modificare i permessi di accesso ai propri file
4. Ogni utente deve poter ristrutturare i propri file in modo attinente al problema affrontato
5. Ogni utente deve poter muovere dati da un file ad un altro
6. Ogni utente deve poter mantenere una copia di backup dei propri file (in caso di danno)
7. Ogni utente deve poter accedere ai propri file tramite nomi simbolici

File System: Organizzazione del codice

- Directory Management: da nomi di file a identificatori di file; tutte le operazioni utente che hanno a che fare con i file (crearli, cancellarli, spostarli, ...)
- File System: struttura logica ed operazioni (apri, chiudi, leggi, scrivi, ...)
- Organizzazione fisica: da identificatori di file a indirizzi fisici su disco; allocazione/deallocazione
- Scheduling & Control: ovviamente è qui che ci sono i vari SCAN e compagnia bella

Roadmap

- Visione d'insieme
- Le directory
- Gestione della memoria secondaria
- Gestione dei file in UNIX
- Gestione dei file su Windows

Cosa Contengono

- Informazioni sui file
  - attributi
  - posizione (dove sono i dati)
  - proprietario
- Una directory è essa stessa un file (speciale)
- Fornisce il mapping tra nomi dei file e file stessi

Operazioni su una Directory

- Ricerca
- Creazione file
- Cancellazione file
- Lista del contenuto della directory
- Modifica della directory

Elementi delle Directory: Informazioni di Base

- Nome del file
  - nome scelto dal creatore (utente o processo)
  - unico in una directory data
- Tipo del file
  - eseguibile, testo, binario, ...
- Organizzazione del file
  - per sistemi che supportano diverse possibili organizzazioni

Elementi delle Directory: Informazioni sull'Indirizzo

- Volume
  - indica il dispositivo su cui il file è memorizzato
- Indirizzo di partenza
  - ad es.: da quale settore o traccia di disco
- Dimensione attuale
  - in byte, word o blocchi
- Dimensione allocata
  - dimensione massima del file

Elementi delle Directory: Controllo di Accesso

- Proprietario
  - può concedere/negare i permessi ad altri utenti, e può anche cambiare tali impostazioni
- Informazioni sull'accesso
  - potrebbe contenere username e password per ogni utente autorizzato
- Azioni permesse
  - per controllare lettura, scrittura, esecuzione, spedizione tramite rete

Elementi delle Directory: Informazioni sull'Uso

- Data di creazione
- Identità del creatore
- Data dell'ultimo accesso in lettura
- Data dell'ultimo accesso in scrittura
- Identità dell'ultimo lettore
- Identità dell'ultimo scrittore
- Data dell'ultimo backup
- Uso attuale
  - lock, azione corrente, ...

Una Semplice Struttura per le Directory

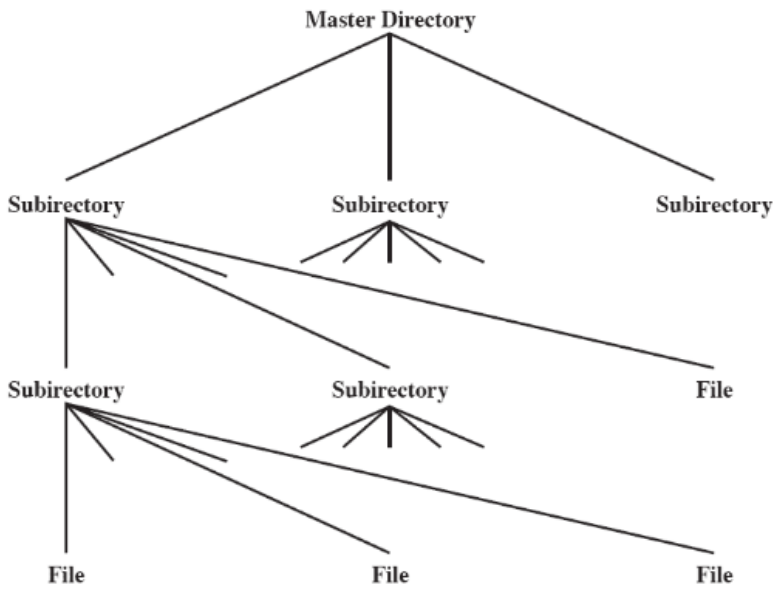
- Il metodo usato per memorizzare le informazioni di cui sopra varia molto da sistema a sistema
- Quello più semplice è fare una lista di entry, una per ogni file
  - file sequenziale con il nome del file a far da chiave
  - non aiuta nell'organizzare i file
  - non si può dare lo stesso nome a due file diversi

Schema a Due Livelli per le Directory

- Una directory per ogni utente, più una (*master*) che le contiene
  - la master contiene anche l'indirizzo e le informazioni per il controllo dell'accesso
- Ogni directory utente è solo una lista dei file di quell'utente
  - non offre struttura per insiemi di files

Schema Gerarchico ad Albero per le Directory

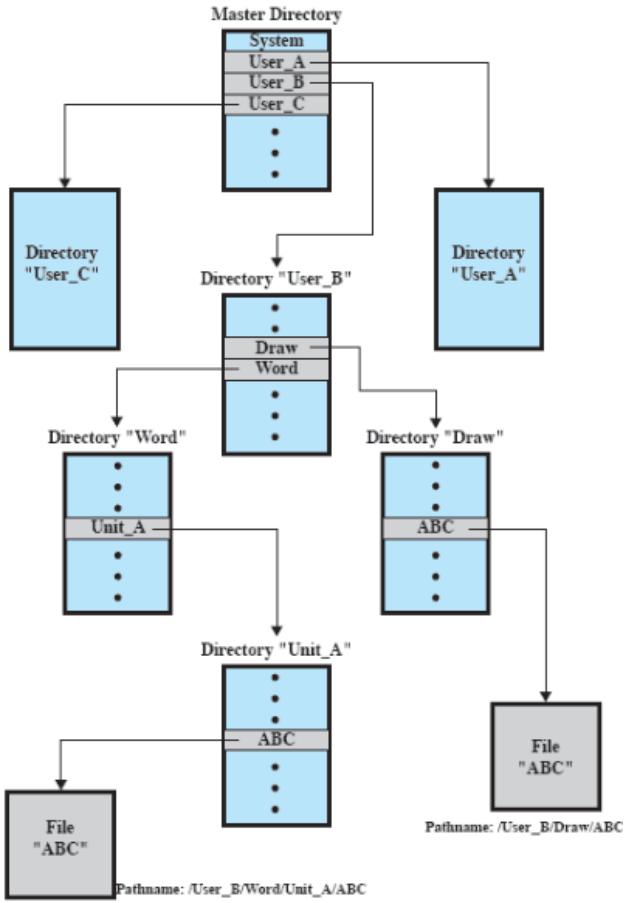
- Una directory *master* che contiene le directory utente
- Ogni directory utente può contenere file oppure altre directory utente
- Ci sono anche sottodirectory di sistema, sempre dentro la directory master



Nomi

- Gli utenti devono potersi riferire ad un file usando solo il suo nome
  - i nomi devono essere unici, ma un utente può non aver accesso a tutti i file dell'intero sistema
- La struttura ad albero permette agli utenti di trovare un file seguendo un percorso nell'albero (*directory path*)
  - nomi duplicati sono possibili purché con path diversi
  - ovvero, in directory diverse

Directory ad Albero, Esempio



Directory di Lavoro

- Dover dare ogni volta il path completo prima del nome del file può essere lungo e noioso
- Solitamente, gli utenti o i processi interattivi hanno associata una *directory di lavoro o corrente*
  - tutti i nomi di file sono dati relativamente a questa directory
  - è sempre possibile dare esplicitamente l'intero percorso, se necessario

Roadmap

- Visione d'insieme
- Le directory
- Gestione della memoria secondaria
- Gestione dei file in UNIX
- Gestione dei file su Windows

Gestione della Memoria Secondaria

- Il SO è responsabile dell'assegnamento di blocchi a file
- Due problemi correlati
  - occorre allocare spazio per i file, e mantenerne traccia una volta allocato
  - occorre tener traccia dello spazio allocabile
  - l'un problema influenza l'altro
- I file si allocano in "porzioni" o "blocchi"
  - l'unità minima è il settore del disco
  - ogni porzione o blocco è una sequenza *contigua* di settori

Allocazione di Spazio per i File

- Vari problemi da affrontare:
  - preallocazione vs. allocazione dinamica
  - porzioni di dimensione fissa o dinamica, e quanto grandi
    - si usa "porzioni" per quelle di dimensione dinamica, "blocco" per quelle di dimensione fissa
  - metodo di allocazione: contiguo, concatenato o indicizzato
  - gestione della file allocation table
    - per ogni file, mantiene le informazioni su dove sono, sul disco, le porzioni che lo compongono

Preallocazione vs. Allocazione Dinamica

- Preallocazione: occorre che la massima dimensione sia dichiarata a tempo di creazione
- La dimensione è facilmente stimabile in alcune applicazioni
  - es.: risultato di compilazioni, file che forniscono sommari su dati
- Difficile in molte altre: utenti ed applicazioni sovrastimano la dimensione
  - così da poter effettivamente memorizzare le informazioni desiderate nel file
- Risultato: spreco di spazio su disco, a fronte di un modesto risparmio di computazione
- Allocazione dinamica quasi sempre preferita
  - dimensione aggiustata in base alle `append` o alle `truncate`

Dimensione delle Porzioni

- Due possibilità agli estremi:
  - si alloca una porzione larga a sufficienza per l'intero file
    - efficiente per il processo che vuole creare il file: l'accesso sequenziale è il più veloce
  - si alloca un blocco alla volta
    - efficiente per il SO, che deve gestire tanti file
    - ciascun blocco è una sequenza di n settori contigui, con n fisso e piccolo (spesso $n = 1$)
- Si cerca un punto d'incontro (*trade-off*) tra efficienza del singolo file ed efficienza del sistema
  - sarebbe ottimo, per le prestazioni di accesso al file, fare porzioni contigue
  - porzioni piccole vuol dire grandi tabelle di allocazione, e quindi grande overhead
  - ma vuol anche dire maggior facilità di riuso dei blocchi
  - da evitare porzioni fisse grandi: frammentazione interna
  - frammentazione esterna sempre possibile: i file possono venire cancellati...

Dimensione delle Porzioni

Alla fine, 2 possibilità (valide sia per preallocazione che per allocazione dinamica):

- Porzioni grandi e di dimensione variabile
  - ogni singola allocazione è contigua
  - tabella di allocazione abbastanza contenuta
  - complicata la gestione dello spazio libero: servono algoritmi ad hoc
- Porzioni fisse e piccole
  - tipicamente, 1 blocco per 1 porzione
  - molto meno contiguo del precedente
  - spazio libero: basta guardare una tabella di bit...

Dimensione delle Porzioni

- Preallocazione + porzioni grandi e di dimensione variabile
  - se si usa questa combinazione, niente tabella di allocazione: per ogni file basta l'inizio e la lunghezza
  - ogni file è un'unica porzione
  - come per il porzionamento della RAM: best fit, first fit, next fit
  - ma qui non c'è un vincitore, troppe variabili
  - inefficiente per lo spazio libero: necessita periodica compattazione
  - e compattare il disco è mooooooolto più oneroso che compattare la RAM
    - è un I/O...

Come Allocare Spazio per i File

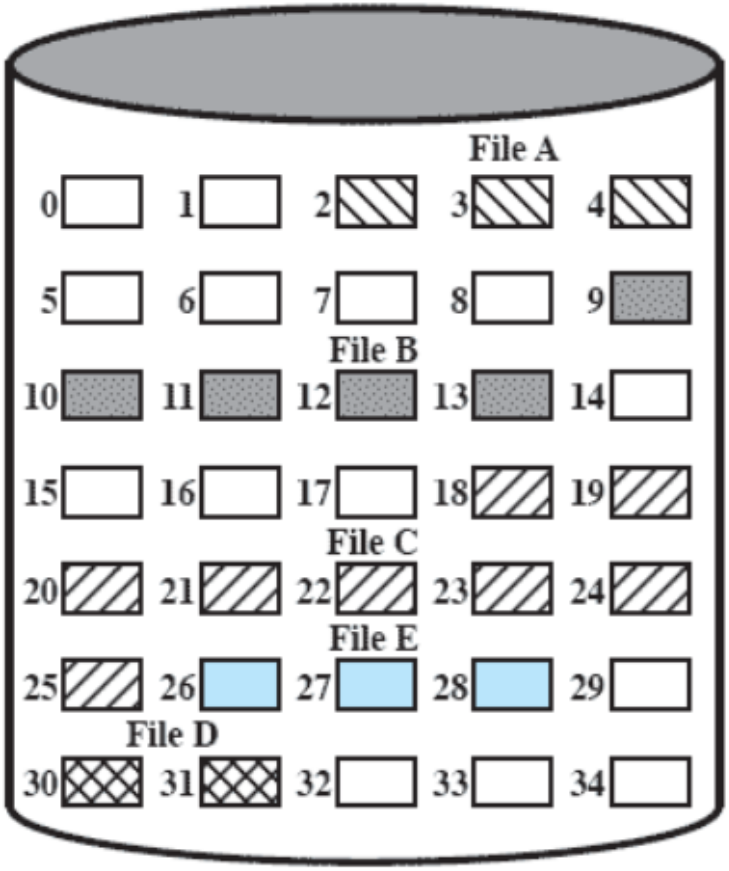
- Tre metodi:
  - contiguo
  - concatenato
  - indicizzato
- Serve comunque una tabella di allocazione dei file

	Contiguous	Chained	Indexed	
Preallocation?	Necessary	Possible	Possible	
Fixed or variable size portions?	Variable	Fixed blocks	Fixed blocks	Variable
Portion size	Large	Small	Small	Medium
Allocation frequency	Once	Low to high	High	Low
Time to allocate	Medium	Long	Short	Medium
File allocation table size	One entry	One entry	Large	Medium

Allocazione Contigua

- Un insieme di blocchi viene allocato per il file quando quest'ultimo viene creato
  - preallocazione necessaria, occorre sapere quanto lungo, al massimo, sarà il file
  - altrimenti, se un file può crescere oltre il limite massimo, potrebbe incontrare blocchi già occupati, e niente contiguità
- È necessaria una sola entry nella tabella di allocazione dei file
  - blocco di partenza e lunghezza del file
- Ci sarà frammentazione esterna
  - con conseguente necessità di compattazione

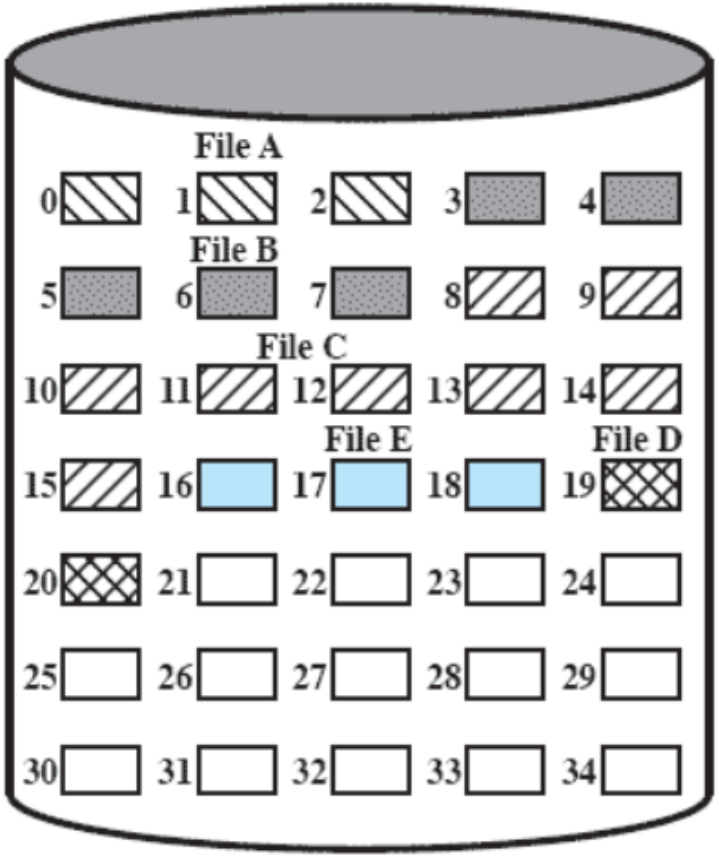
Allocazione Contigua



File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

Compattazione



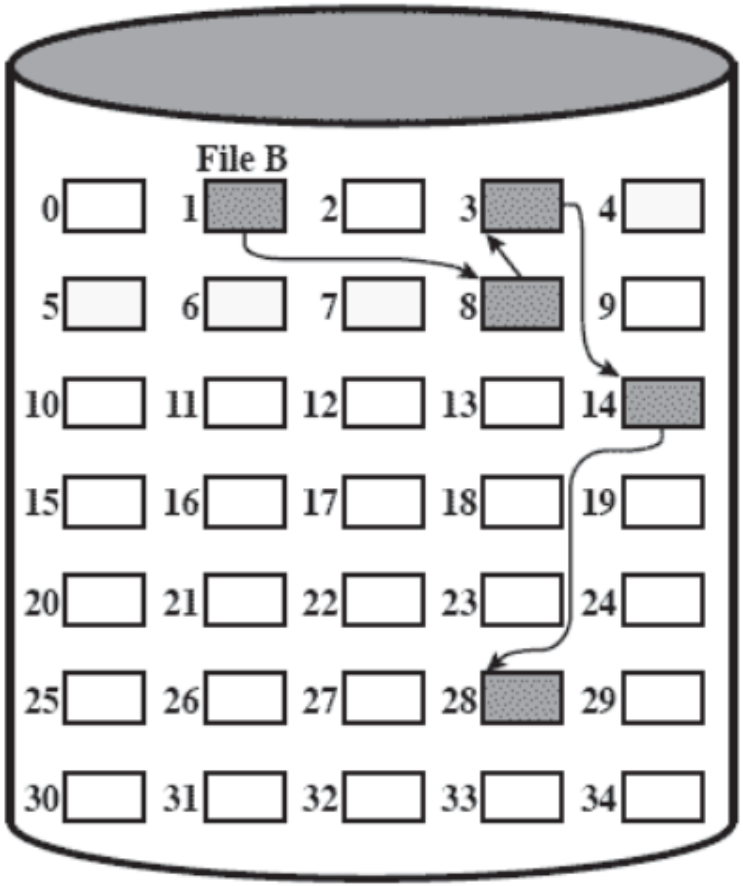
File Allocation Table

File Name	Start Block	Length
File A	0	3
File B	3	5
File C	8	8
File D	19	2
File E	16	3

Allocazione Concatenata

- Allocazione di un blocco alla volta
- Ogni blocco ha un puntatore al prossimo blocco
  - la prima parte del blocco sono dati del file, l'ultima (piccola) parte del blocco è il puntatore
- È necessaria una sola entry nella tabella di allocazione dei file
  - blocco di partenza e lunghezza del file
  - lunghezza del file anche calcolabile, ma è comodo avere già il valore calcolato...
- Niente frammentazione esterna
  - frammentazione interna trascurabile
- Ok per file da accedere sequenzialmente
  - ma se serve un certo blocco che si trova b blocchi dopo quello iniziale, occorre scorrere tutta la lista
- Consolidamento: analogo alla compattazione, per mettere i blocchi di un file contigui e migliorare l'accesso non sequenziale

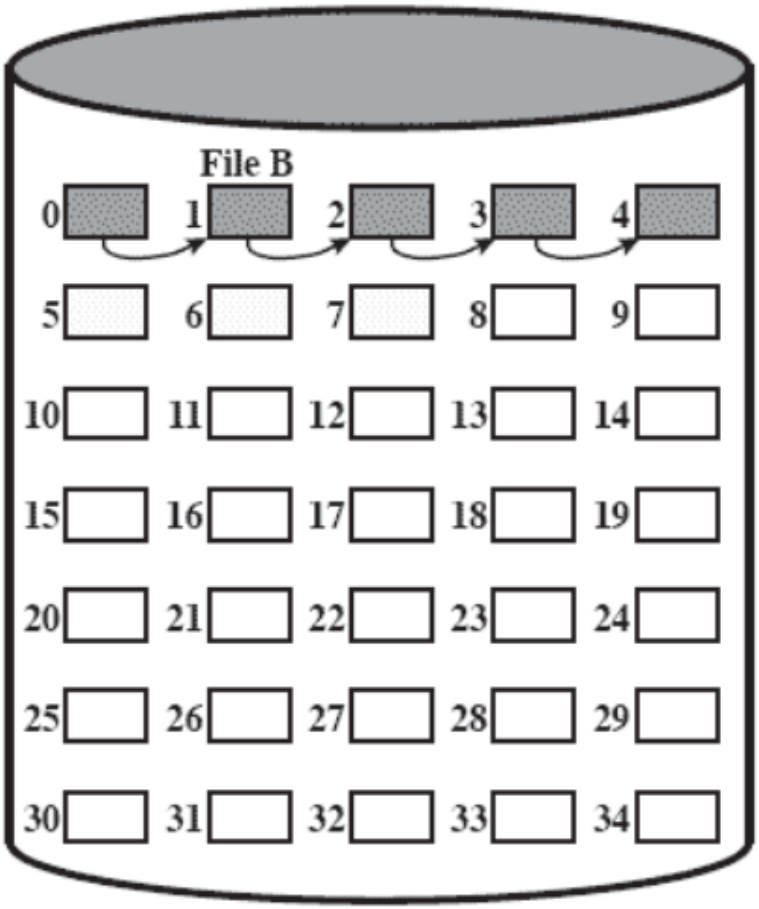
Allocazione Concatenata



File Allocation Table

File Name	Start Block	Length
...	...	...
File B	1	5
...	...	...

Allocazione Concatenata: Consolidamento



File Allocation Table

File Name	Start Block	Length
...	...	...
File B	0	5
...	...	...

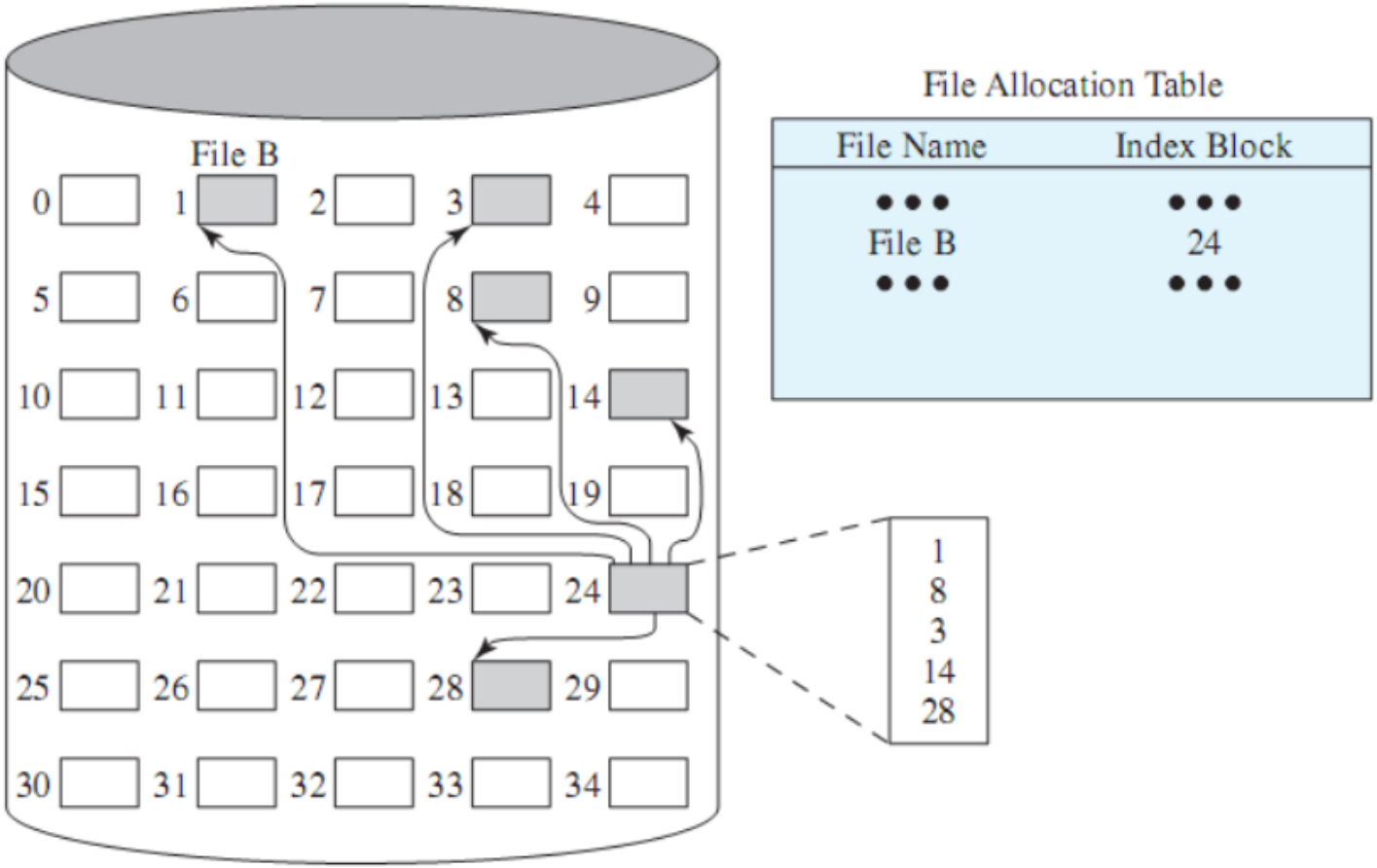
Allocazione Indicizzata

- Via di mezzo tra i due precedenti, ne risolve quasi tutti i problemi
- La tabella di allocazione dei file contiene, apparentemente, una sola entry, con l'indirizzo di un blocco
- Questo blocco, in realtà, ha una entry per ogni porzione allocata al file
  - quindi fa parte della tabella a tutti gli effetti
  - pur trovandosi in un blocco apparentemente indistinguibile da quelli usati per i dati del file
- E se il file è troppo grande? si fanno più livelli
  - esempio tipico: i-node di Unix-Linux
- Ovviamente ci dev'essere un bit che dica se un blocco è composto da dati o è un indice

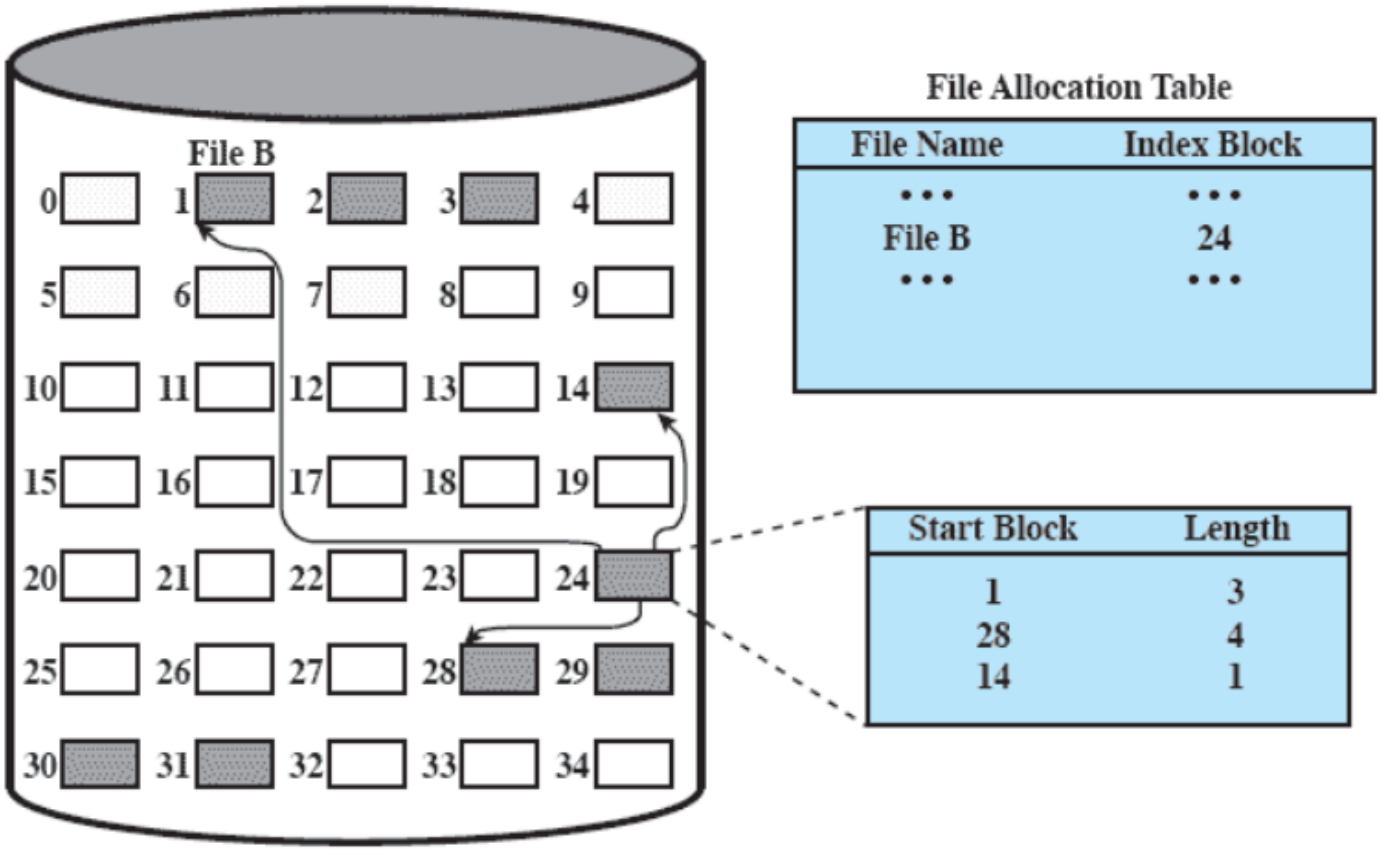
Allocazione Indicizzata

- L'allocazione può essere con:
  - blocchi di lunghezza fissa: niente frammentazione esterna
  - blocchi di lunghezza variabile: migliora la località
- A volte occorre il consolidamento
  - blocchi di lunghezza fissa: migliora la località
  - blocchi di lunghezza variabile: riduce la dimensione dell'indice

Allocazione Indicizzata (Porzioni di Lunghezza Fissa)



Allocazione Indicizzata (Porzioni di Lunghezza Variabile)



Gestione dello Spazio Libero

- La gestione dello spazio libero è altrettanto importante di quello occupato
- Per allocare spazio per i file, occorre sapere dov'è lo spazio libero
  - non è realistico guardare la tabella di allocazione di tutti i file per determinare quali blocchi/porzioni sono liberi!
- Serve una tabella di allocazione di disco, oltre che a quella di allocazione per i file
- Ogni volta che si alloca o si cancella un file, lo spazio libero va aggiornato

Tabelle di Bit

- Vettore con un bit per ogni blocco su disco
  - 0: libero; 1: occupato
- Va bene per tutti gli schemi visti finora
- Minimizza lo spazio richiesto alla tabella di allocazione del disco
- Se il disco è quasi pieno, la ricerca di uno spazio libero può richiedere molto tempo
  - risolvibile con delle tabelle riassuntive di porzioni della tabella di bit
  - es.: numero di blocchi liberi in totale e numero di blocchi liberi contigui

Porzioni Libere Concatenate

- Le porzioni libere possono essere concatenate le une alle altre usando, per ogni blocco libero, un puntatore ed un intero per la dimensione
- Praticamente senza overhead di spazio
- Va bene per tutti gli schemi visti finora
- Problemi:
  - se c'è frammentazione, le porzioni sono tutte da un blocco e la lista diventa lunga
  - occorre leggere un blocco libero per sapere qual è il prossimo: se occorre allocarne molti, diventa time consuming
  - è lungo anche cancellare file molto frammentati

Indicizzazione

- Tratta lo spazio libero come un file, e quindi usa un indice come si farebbe per un file
- Per efficienza, l'indice gestisce le porzioni come se fossero di lunghezza variabile
  - quindi c'è una entry per ogni porzione libera nel disco
- È un approccio che fornisce un supporto efficiente a tutti i metodi di allocazione visti finora

Lista dei Blocchi Liberi

- Ad ogni blocco viene assegnato un numero sequenziale
- La lista di questi numeri viene memorizzata in una parte dedicata del disco
- Se per ogni blocco servono 4 bytes e i blocchi sono da 512 bytes, richiede meno dell'1% di spazio su disco
- Per avere parti della lista in memoria principale, si può:
  - organizzare la lista come pila, e tenere solo la parte alta
  - pop per allocare spazio libero, push per deallocare spazio occupato
  - quando la parte in memoria principale finisce, si prende una nuova parte da disco
  - anche come coda

Volumi

- Essenzialmente, è un disco "logico"
  - partizione di un disco
  - più dischi messi insieme e visti come un disco solo (LVM)
- Un insieme di settori in memoria secondaria, che possono essere usati dal SO o dalle applicazioni
- I settori di un volume non devono necessariamente essere contigui, ma appariranno come tali al SO e alle applicazioni
- Un volume potrebbe essere il risultato dell'unione di volumi più piccoli

Dati e Metadati: Consistenza

- Dati: contenuto dei file
- Metadati: lista blocchi liberi, lista blocchi all'interno dei file, data di ultima modifica, ...
- I metadati devono essere su disco, perché devono essere persistenti
- Per efficienza, vengono anche tenuti in memoria principale
- Mantenere sempre consistenti metadati in memoria principale e su disco è inefficiente
- Quindi, si fa solo di tanto in tanto, quando il disco è poco usato, e con più aggiornamenti insieme
- *Journaling*: anziché scrivere le informazioni nelle opportune porzioni di disco, le si scrive in una zona di disco dedicata (*log*)
  - in caso di reboot dopo un crash, basta leggere il log

Dati e Metadati: Consistenza

- E se c'è un evento imprevisto?
  - il computer viene spento all'improvviso, senza una procedura di chiusura (ad es.: per mancanza di corrente)
  - il disco viene rimosso senza dare un appropriato comando (unmount; ad es.: chiavetta USB)
- Basta scrivere un bit all'inizio del disco, che dice se il sistema è stato spento correttamente
- Al reboot, se il bit è 0, occorre eseguire un programma di ripristino del disco
  - blocco in uso ma non appartenente a nessun file? lo si dichiara libero
  - blocco libero ma appartenente ad un file? lo si dichiara appartenente a quel file
  - con il journaling è più facile, basta consultare il log

Roadmap

- Visione d'insieme
- Le directory
- Gestione della memoria secondaria
- Gestione dei file in UNIX
- Gestione dei file su Windows

Gestione dei File in UNIX

- Sei tipi di file:
  - normale
  - directory
  - speciale (mappano su nomi di file i dispositivi di I/O)
  - named pipe (per far comunicare processi tra loro)
  - hard link (collegamenti, nome di file alternativo)
  - link simbolico (il suo contenuto è il nome del file cui si riferisce)

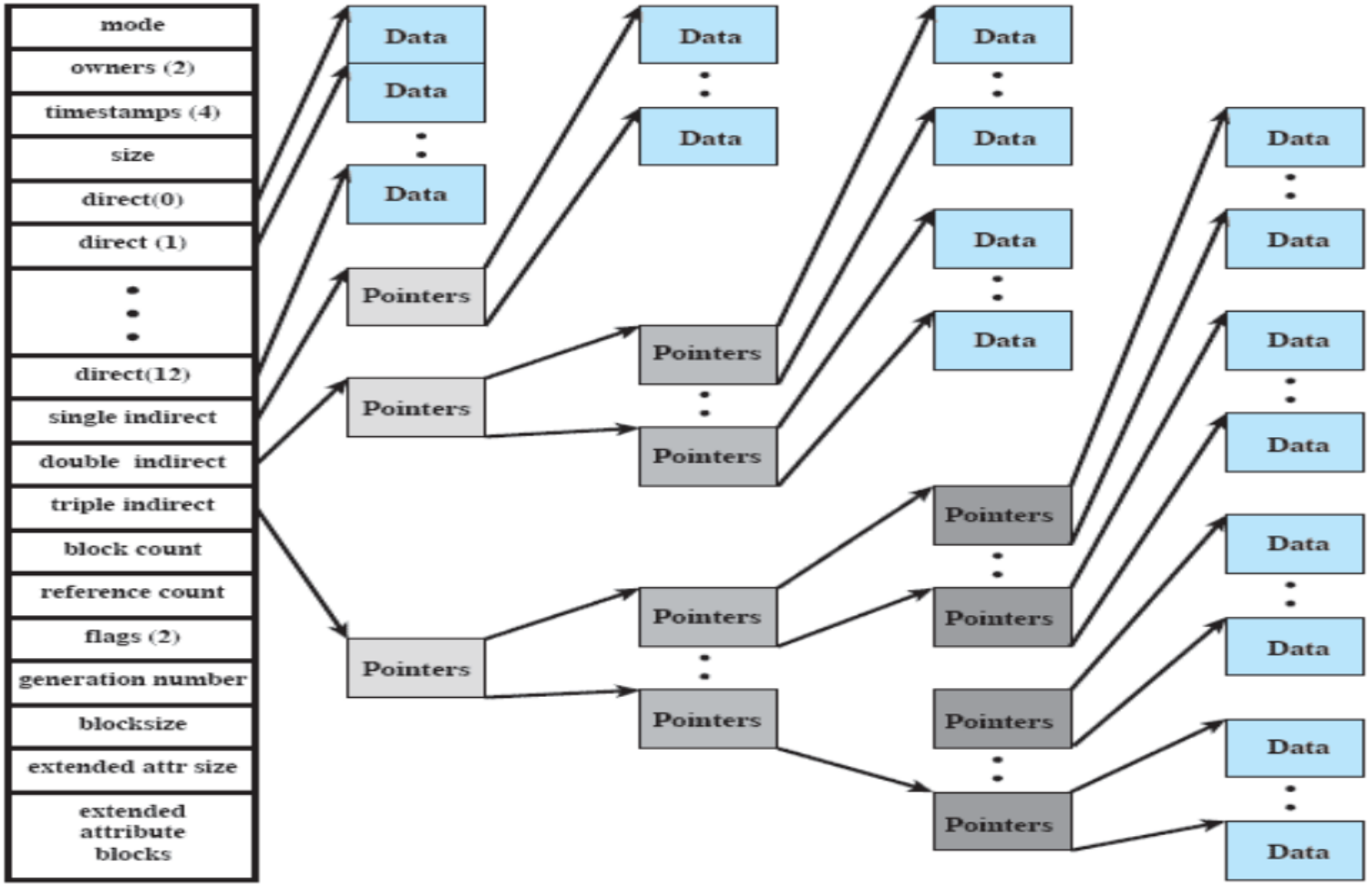
Inode

- Sta per "index node"
  - ispirato al metodo di allocazione indicizzato, con dimensione fissa dei blocchi
- Struttura dati che contiene le informazioni essenziali per un dato file
- Un dato inode potrebbe essere associato a più nomi di file
  - hard link
  - ma un inode attivo è associato ad un solo file
  - ogni file è controllato da un solo inode
- Viene mantenuta dal SO una tabella di tutti gli inode corrispondenti a file *aperti* (in memoria principale)
- Tutti gli altri i-node sono in una zona di disco dedicata (*i-list*)

Inode in Free BSD

- Tipo e modo di accesso del file
- Identificatore dell'utente proprietario e del gruppo cui tale utente appartiene
- Tempo di creazione e di ultimo accesso (lettura o scrittura)
- Flag utente e flag per il kernel
- Numero sequenziale di generazione del file
- Dimensione delle informazioni aggiuntive
- Altri attributi (controllo di accesso e altro)
- Dimensione
- Numero di blocchi, o numero di file (per le directory)
- Dimensione dei blocchi
- Sequenze di puntatori a blocchi

Inode in Free BSD



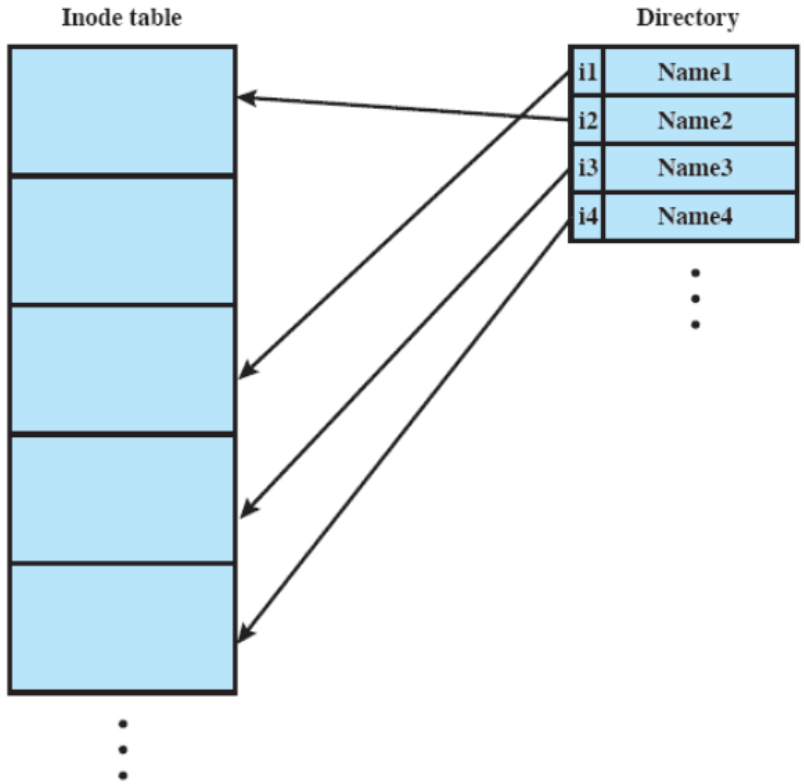
Allocazione di File

- Fatta a blocchi
- Allocazione dinamica
  - quindi, blocchi potenzialmente non contigui
- L'indicizzazione tiene traccia dei blocchi dei file
  - parte dell'indice è memorizzata nell'inode
- L'inode ha anche alcuni puntatori diretti
  - e 3 puntatori indiretti

Level	Number of Blocks	Number of Bytes
Direct	12	48K
Single Indirect	512	2M
Double Indirect	$512 \times 512 = 256K$	1G
Triple Indirect	$512 \times 256K = 128M$	512G

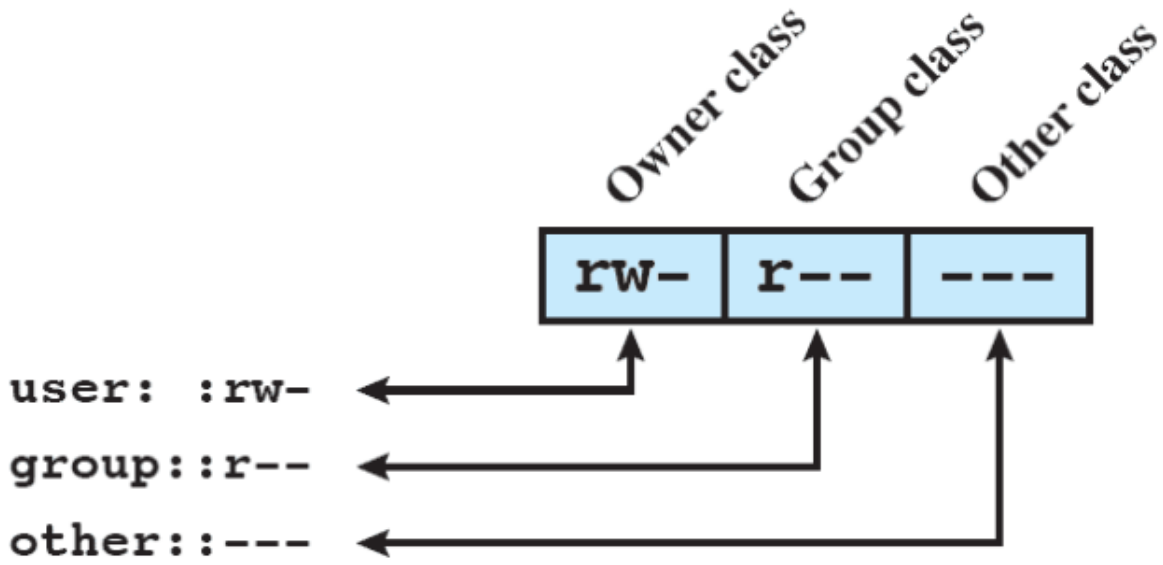
Inode e Directory

- Le directory sono file che contengono:
  - una lista di coppie (nome di file, puntatore ad inode)
  - alcuni di questi file potrebbero essere a loro volta directory, quindi è una struttura gerarchica
  - una directory può essere modificata solo dal sistema operativo, ma letta da ogni utente



Accesso ai File

- Per ogni file, ci sono 3 terne di permessi
  - lettura, scrittura, esecuzione
  - per il proprietario, per il suo gruppo e per tutti gli altri



(a) Traditional UNIX approach (minimal access control list)

Roadmap

- Visione d'insieme
- Le directory
- Gestione della memoria secondaria
- Gestione dei file in UNIX
- Gestione dei file su Windows

Gestione dei File su Windows

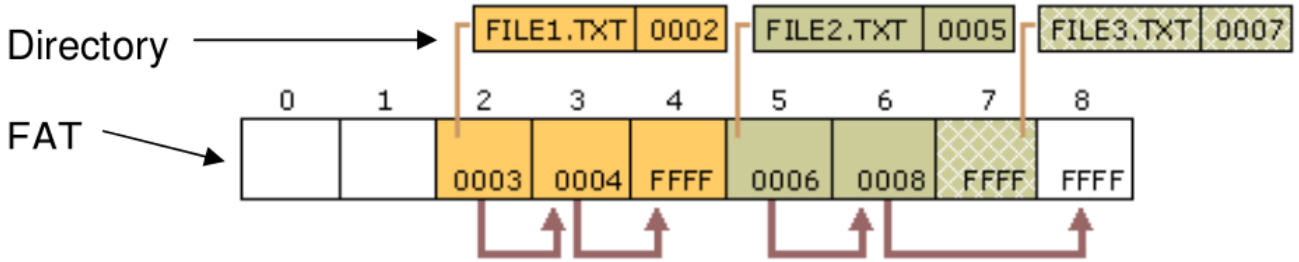
- File system vecchio (da MS-DOS): FAT
  - allocazione concatenata, con blocchi (*cluster*) di dimensione fissa
- File system nuovo: NTFS
  - allocazione con bitmap (!), con blocchi (*cluster*) di dimensione fissa

Caratteristiche Principali di FAT

- Molto limitato, andava bene per i vecchi dischi (soprattutto per i floppy)
- Ancora usato per le chiavette USB
- File Allocation Table (FAT), memorizzata all'inizio della partizione su disco
  - `https://it.wikipedia.org/wiki/File_Allocation_Table`
  - una sola colonna: valore intero a 12, 16 o 32 bit (FAT-12, FAT-16 o FAT-32)
  - tante righe quanti sono i *cluster* del disco
    - ogni cluster ha dimensione variabile tra 2 e 32 KB, ed è costituito da settori di disco contigui
    - variabile nel senso che può cambiare da partizione a partizione, ma resta fisso all'interno di una partizione
- Approccio non scalabile: la FAT stessa può occupare molto spazio

File Allocation Table

- Se la entry *i*-esima è zero, il blocco *i*-esimo è libero
- Se la entry *i*-esima non è zero e non è un valore speciale, allora è il prossimo blocco del file
- Valori speciali: tutti 1 vuol dire ultimo blocco del file
- È poi presente la struttura delle directory
  - all'inizio in FAT-12 e FAT-16, in cui la dimensione delle directory è limitata
  - assieme ai file in FAT-32 (ma la root comincia all'inizio)



FAT32: Esempio su Terminale Linux Ubuntu

- `dd if=/dev/zero of=fs-virtuale-vfat bs=1M count=100`
- `mkfs.fat -F32 fs-virtuale-vfat`
  - crea un file (chiamato `fs-virtuale-vfat`) da 100MB, e lo formatta con FAT32, come se fosse un disco
- `od -Ax -t x1 -t c fs-virtuale-vfat | less`
  - per visualizzare il contenuto del file system, byte per byte
  - prima riga in esadecimale, seconda in caratteri se ci sono caratteri stampabili ASCII, altrimenti ottale
- `mkdir -p dir; sudo mount -o uid=1000,gid=1000 -t vfat fs-virtuale-vfat dir`
  - monta il filesystem appena creato su una nuova directory; serve la password dell'utente attuale, che dev'essere superuser
- Creare file e/o directory dentro dir
- `sudo umount dir`
- di nuovo `od -Ax -t x1 -t c fs-virtuale-vfat | less`

E Linux?

- Nativamente, supporta ext2/ext3/ext4
  - ext2: direttamente dai file system Unix originari
  - ext3: ext2+journaling
  - ext4: ext3 in grado di memorizzare singoli file più grandi di 2TB e filesystem più grandi di 16TB
- Pieno supporto per gli i-node, memorizzati nella parte iniziale del file system
- Linux permette anche di leggere e scrivere altri file system, come per esempio quelli di Windows
  - con FAT, non è possibile memorizzare gli i-node sul dispositivo
  - vengono creati on-the-fly su una cache quando vengono aperti i file