Caratteristiche del *File System* – 1

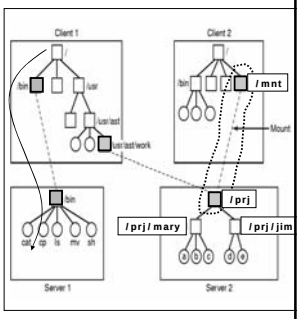
- Paradigma minimalista di tipo "*small is beautiful*"
- *File* visto da FS come **sequenza di *byte*** di contenuto arbitrario
  - Significato fissato dal programma applicativo
- *File* regolari, *file* cartelle (*directory*) e *file* speciali che mappano dispositivi di I/O
- Nome inizialmente limitato a 14 caratteri (UNIX v7)
- Poi esteso fino a 255 (UNIX BSD → GNU/Linux)
  - Estensione **non** obbligatoria
  - Convenzione di estensione a scelta del programma applicativo e/o dell'utente
    - Esempio: `makefile` assume estensione, `emacs` no

Caratteristiche del *File System* – 2

- *File* designato mediante cammino (*path*) assoluto o relativo
  - Il cammino relativo richiede la nozione di "*directory* (di lavoro) corrente"
    - **`pwd`** per visualizzarne la posizione assoluta
      - *Print working directory*
    - **`cd`** per cambiare posizione
      - *Change (to) directory*
  - Un intero FS **`B`** posto su una partizione visibile può essere inglobato in un FS **`A`** mediante **`mount`**
    - La radice di **`B`** viene designata con un nome (cammino) specifico in **`A`** detto *mount point*

Caratteristiche del *File System* – 3

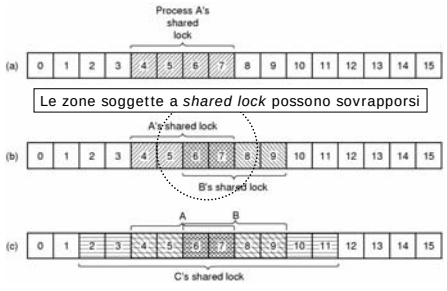
- Tramite "**mount**" la partizione "Client 1" ingloba dalla partizione "Server 1" il FS radicato in `/bin`
  - Il cammino `/bin/cat` ora porta al *file* `cat` come se fosse pienamente nella partizione di "Client 1"
- Lo stesso avviene per il FS radicato in `/prj` della partizione "Server 2"
  - Il cammino `/usr/ast/work/mary/a` porta al *file* `/prj/mary/a`



Caratteristiche del *File System* – 4

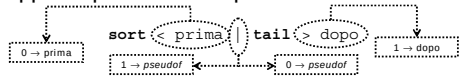
- **Controllo di accessi concorrenti** (*locking*)
  - A grana grossa (per *directory* o per *file*)
    - Mediante uso esplicito di semafori convenzionali
  - A grana fine (per gruppi di *byte* in un *file*)
    - Mediante meccanismi dedicati
- Due distinte modalità d'uso
  - **Accesso simultaneo condiviso** (*shared lock*)
    - Più accessi R alla stessa zona ma anche a zone solo parzialmente sovrapposte
  - **Accesso esclusivo** (*exclusive lock*)
    - Consente un solo accesso per zona selezionata

Caratteristiche del *File System* – 5



Caratteristiche del *File System* – 6

- ∀ *file* aperto vi è un descrittore (`int > 0`) che designa la posizione corrente di R/W
- 3 descrittori sono pre-assegnati dalla *shell* per altrettanti *file* aperti per definizione
  - **0** per **`stdin`**, **1** per **`stdout`**, **2** per **`stderr`**
  - La redirezione (> , <) modifica tali assegnamenti
- La *pipe* | crea uno pseudo-*file* (con descrittore proprio) che rileva gli **`stdout`** e **`stdin`** di una coppia di processi che operano in cascata



Realizzazione del FS in UNIX – 1

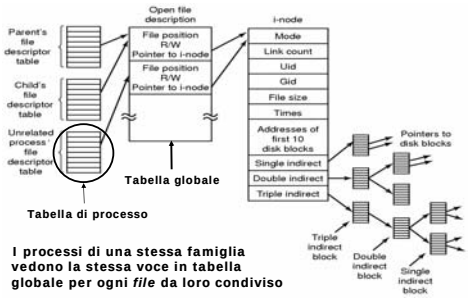
- **Struttura di partizione secondo UNIX v7**
- Il super-blocco (**1**) indica tra l'altro il # di *i-node* e di blocchi nel FS e fornisce il puntatore alla lista dei blocchi liberi (**2**)
- Gli *i-node* (**3**) sono numerati 1..N
  - E tutti sono di ampiezza ≥ 64 B
- *Directory* come insieme **variabile** e **non ordinato** di unità informative (*entry*)
  - Ampie 16 B
    - 14 B (codifica ASCII) per nome di *file*
    - 2 B per numero di *i-node*

Blocco di *boot*	Superblocco	Lista liberi	Lista *i-node*	*Blocchi dati*
0	**1**	**2**	**3**	

Realizzazione del FS in UNIX – 2

- In nucleo usa **due** strutture di controllo
  - Un insieme di **tabelle di processo** contiene "descrittori utente" dei *file* attualmente in uso a ciascun processo
    - A ogni descrittore utente deve corrispondere l'attuale posizione di R/W
    - Però ogni processo deve avere il suo proprio indice di posizione sui propri *file* aperti
      - Possono esistere **più** posizioni di R/W su uno stesso *file* condiviso
      - L'indice **non può** essere ritenuto nell'*i-node* che è **unico** per *file* !
    - Sequenze ordinate di processi figli di uno stesso padre devono poter scrivere su uno stesso *file* consecutivamente
      - Lo **stesso indicatore** di posizione per processi di una **stessa famiglia**
  - Una **tabella globale** mantiene la corrispondenza tra tutti i *file* aperti e i loro *i-node*
    - Ciascuna voce nella **tabella di processo** punta a una voce nella **tabella globale** che specifica diritti e posizione di R/W corrente nel *file*
      - La stessa voce ∀ *file* condiviso da processi di una stessa famiglia
      - Voce diversa per stesso *file* per processi non apparentati

Realizzazione del FS in UNIX – 3



I processi di una stessa famiglia vedono la stessa voce in tabella globale per ogni *file* da loro condiviso

Realizzazione del FS in UNIX – 4

- L'*i-node* principale del *file* contiene (tra l'altro) l'indirizzo dei suoi primi 12 blocchi dati
  - L' *i-node* ha la dimensione di 1 frazione di blocco (64 B)
- Per *file* più grandi 1 campo dell'*i-node* principale punta a 1 *i-node* secondario che contiene puntatori ad altri blocchi dati
  - I-*node* principale con campo *single-indirect*
- Per *file* ancora più grandi l'*i-node* secondario contiene puntatori a nodi *single-indirect*
  - I-*node* principale con campo *double-indirect*
- È previsto anche un campo *triple-indirect*

Realizzazione del FS in UNIX – 5

Field	Bytes	Description
Mode	2	File type, protection bits, setuid, setgid bits
Nlinks	2	Number of directory entries pointing to this i-node (via *hard link*)
Uid	2	UID of the file owner
Gid	2	GID of the file owner
Size	4	File size in bytes
Addr	39	Address of first 10 disk blocks, then 3 indirect blocks (3 B / blocco)
Gen	1	Generation number (incremented every time i-node is reused)
Atime	4	Time the file was last accessed
Mtime	4	Time the file was last modified
Ctime	4	Time the i-node was last changed (except the other times)
	64	Struttura interna di un *i-node*

Esempio d'uso di *i-node* (UNIX v7)

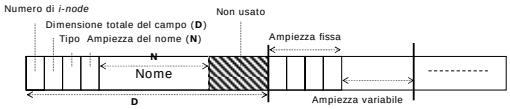
- **Ipotesi A** (le strutture "indirette" sono *i-node*)
  - Blocco dati di capienza 4 KB
  - *i-node* ampio 64 B
  - Indici di blocco espressi su 4 B
- **Esempio 1** (con uso di campo *single-indirect*)
  - Max dimensione di *file* rappresentabile
    - $(10 + 64\text{ B} / 4\text{ B}) \times 4\text{ KB} = (10 + 16) \times 4\text{ KB} = 104\text{ KB}$
- **Esempio 2** (con uso di campo *double-indirect*)
  - Max dimensione di *file* rappresentabile
    - $104\text{ KB} + 16^2 \times 4\text{ KB} = 1\text{ MB} + 104\text{ KB}$
- **Esempio 3** (con uso di campo *triple-indirect*)
  - Max dimensione di *file* rappresentabile
    - $1\text{ MB} + 104\text{ KB} + 16^3 \times 4\text{ KB} = 17\text{ MB} + 104\text{ KB}$

Esempio d'uso di *i-node* (UNIX v7)

- **Ipotesi B** (le strutture "indirette" sono blocchi)
  - Blocco dati di capienza 4 KB
  - *i-node* ampio 64 B
  - Indici di blocco espressi su 4 B
- **Esempio 1** (con uso di campo *single-indirect*)
  - Max dimensione di *file* rappresentabile
    - $(10 + 4\text{ KB} / 4\text{ B}) \times 4\text{ KB} = (10 + 1\text{ K}) \times 4\text{ KB} = 4\text{ MB} + 40\text{ KB}$

Realizzazione del FS in UNIX – 7

- La versione **BSD** introduce alcune migliorie importanti
  - Estensione del nome di *file* fino a 255 caratteri
  - *Directory* di dimensione **multipla** di blocco
    - Facilita e velocizza la scrittura su disco
    - Comporta frammentazione interna



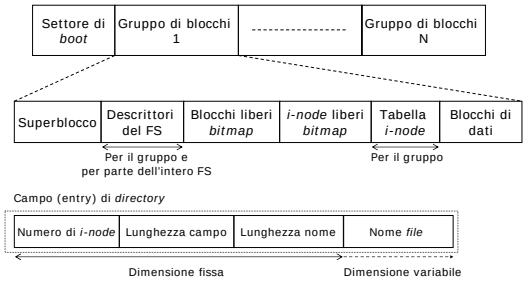
Realizzazione del FS in UNIX – 8

- Altre migliorie **BSD**
  - *Cache* dei nomi di *file* per evitare costosa ricerca lineare su *directory*
  - Disco suddiviso in **gruppi di cilindri**
    - Equivalenti a sotto-partizioni
    - Blocchi dati negli stessi gruppi dei propri *i-node*
    - Oggi di scarso interesse perché i dischi moderni tendono a nascondere al S/O la loro geometria interna
  - 2 ampiezze di blocco
    - Blocchi grandi per *file* molto grandi
    - Blocchi piccoli per *file* piccoli e medi (la norma)
    - Al costo di maggior complessità di gestione

Realizzazione del FS GNU/Linux – 1

- Inizialmente basato sul FS di MINIX
- Subito però abbandonato per le troppe limitazioni
  - Limitazioni MINIX
    - Nomi ≤ 14 caratteri
    - Indirizzi di blocco su 2 B per blocchi ampi 1 KB
    - Ampiezza massima di partizione ≤ 64 MB
      - 2^{16} blocchi × 1 KB = 64 K × 1 KB = 64 MB
- `ext2` diviene presto la versione di riferimento
  - Basata sulle scelte BSD con **diversa struttura fisica**
  - La maggiore innovazione è stata la suddivisione della partizione in **gruppi di blocchi**
    - *i-node* e relativi blocchi dati sono tenuti **vicini** sul disco
    - Maggior robustezza ottenuta replicando su ciascun gruppo le informazioni di controllo del superblocco

Realizzazione del FS GNU/Linux – 2



Realizzazione del FS GNU/Linux – 3

- Dimensione di *i-node* estesa a 128 B
  - Indirizzi di blocco ampi 4 B
    - Per denotare fino a 2^{32} = 4 G blocchi
  - Blocchi di dimensione 1, 2, 4 KB scelta al momento della configurazione del FS
    - Partizione di dimensione ≥ 4 TB
    - 12 indirizzi diretti + 3 indiretti (*single*, *double*, *triple*)
    - Informazioni di controllo
    - Una parte riservata per uso futuro
- Ogni aggiunta a *file* viene realizzata quanto più **localmente** possibile entro lo stesso gruppo
  - Località **tra** *file* correlati tramite gruppi
  - Località **entro** *file* mediante preallocazione di N ≤ 8 blocchi contigui

Realizzazione del FS GNU/Linux – 4

- Una *directory* **`/proc` virtuale** (non esistente su disco) contiene una *directory* ∀ processo presente nel sistema
  - Il nome della *directory* foglia è il PID del processo
  - Il contenuto della *directory* foglia è un insieme di *file* che descrivono il processo e il suo ambiente
    - L'informazione originale resta nel nucleo da dove essa viene estratta alla lettura del *file* virtuale corrispondente
- L'accesso di utente al FS viene filtrato da un **FS virtuale** che consente la coesistenza di più FS di tipo diverso (p. es.: FAT-32 insieme con **`ext2`**)
  - Modalità sostanzialmente analoga a NFS (vedi seguito)

Evoluzioni del FS in GNU/Linux

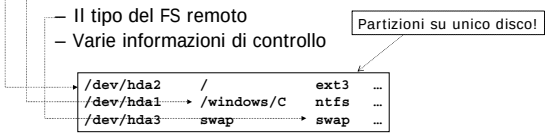
- **Ext3** (2001)
  - Maggior garanzia di consistenza del FS grazie all'uso del *journaling*
    - Archivio delle modifiche al FS salvato su disco prima della loro effettuazione
      - Persistenza anche in caso di interruzioni di alimentazione
- **ReiserFS** (2001)
  - *Directory* strutturata a *B+tree* per maggior efficienza d'uso
  - *Journaling* semplificato
    - Ricorda solo "metadati"
      - Le azioni di preparazione aggiornamento ma non i dati
- **Reiser4** (2007)
  - Lo spazio lasciato libero da frammentazione interna può essere usato per memorizzare *file* piccoli
    - Maggior capienza effettiva quando si usano blocchi medio-grandi

File System di rete (NFS) – 1

- NFS consente a un insieme arbitrario di utenti remoti di condividere uno stesso FS_r
- FS_r viene ospitato su un *server* che i clienti possono contattare
  - Clienti remoti
    - Posti su rete locale o geografica ed eterogenea
  - Clienti locali
    - Posti sullo stesso nodo del *server*
- Il *server* esporta FS_r come sottoalbero del proprio FS_p locale indicandone la "radice"
  - La lista delle "radici" esportate dal nodo viene posta nel *file* di configurazione **`/etc/exports`**
- Il cliente importa (**`mount`**) FS_r posizionandolo come un sottoalbero del proprio FS_c
  - La posizione della "radice" di FS_r è detta *mount point*

File System di rete (NFS) – 2

- Il file **`/etc/fstab`** di ciascun cliente fornisce la lista dei FS importabili mediante **`mount`**
- Per ciascun FS remoto si indicano
  - Il dispositivo di residenza (area su disco)
  - La posizione da assumere nella gerarchia del FS locale
  - Il tipo del FS remoto
  - Varie informazioni di controllo



File System di rete (NFS) – 3

- NFS definisce i protocolli che regolano il dialogo tra il *server* e i suoi clienti
  - **Protocollo di importazione di FS remoto** (**`mount`**)
    - Il cliente invia al *server* il nome della "radice" del FS importato
    - Se la richiesta ha successo il *server* invia al cliente un descrittore unicamente associato al FS esportato
      - Tipo di FS_r disco di residenza, informazioni di controllo, numero di *i-node* della *directory* radice
    - Ogni accesso del cliente a *file* del FS importato userà quel descrittore
    - 2 modalità di importazione
      - **Esplicita**
        - Per inizializzazione eseguita dallo *script* **`/etc/rc`**
      - **Automatica**
        - Al riferimento a *file* residenti in FS importato

File System di rete (NFS) – 4

- **Protocollo di accesso a *file* remoti**
  - Il *server* non mantiene informazioni di stato (*stateless*)
  - Le richieste del cliente sono messaggi contenenti il descrittore del *file* remoto e i parametri dell'operazione richiesta
  - Il *server* ha compito semplice ma a rischio di inconsistenze
    - Lo stato di un *file* può cambiare tra due accessi remoti successivi
- Il controllo di accesso a *file* usa semplicemente diritti di tipo *owner*, *group*, *others*
  - Facilmente falsificabili se non autenticati
- **Nessun** supporto previsto per *lock*

File System di rete (NFS) – 5

- Un livello di FS virtuale (**VFS**) filtra ogni richiesta di accesso ai FS localmente visibili a ogni cliente
- VFS mantiene un ***v-node*** per ogni *file* aperto al quale associa un *i-node* (per *file* locali) oppure un ***r-node*** (per *file* remoti)
  - All'***r-node*** viene associato il descrittore di *file* remoto fornito dal *server* che ne esporta il FS
  - Il traffico di rete viene ridotto trasferendo dati tra cliente e *server* in unità R/W da 8 KB e istituendo 2 *cache* presso il cliente per dati di *file* e riferimenti (*i-node*)
    - Le informazioni in *cache* hanno validità limitata
      - Regolata da un *timer* fissato a 3 s. per blocchi dati e 30 s. per blocchi di *directory*
    - Modalità *read-ahead*
      - Sempre 8 KB in più sull'ultima lettura
    - Scrittura differita al riempimento dell'unità di trasferimento

File System di rete (NFS) – 6