Caratteristiche del *File System* – 1

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

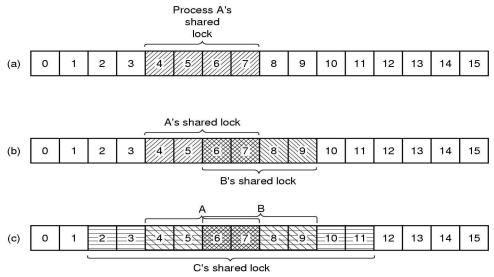
Caratteristiche del *File System* – 2

- *File* designato mediante cammino (*path*) assoluto o relativo
  - Il cammino relativo richiede la nozione di *directory* corrente (di lavoro)
    - **pwd** per visualizzarne la posizione assoluta
    - **cd** per cambiare posizione
  - Un intero FS **B** posto su una partizione visibile può essere ritenuto come parte di 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

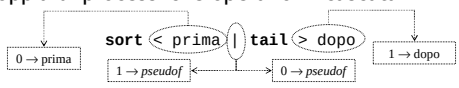
- **Controllo di accessi concorrenti** (*locking*)
  - A grana grossa (per *directory* o per *file*)
    - Mediante uso esplicito di normali semafori
  - A grana fine (per gruppi di *byte* in un *file*)
    - Mediante meccanismi dedicati
- 2 modalità d'uso
  - **Accesso simultaneo condiviso** (*shared 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* – 4



Caratteristiche del *File System* – 5

- ∀ *file* aperto vi è un descrittore (`int > 0`) che denota anche 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

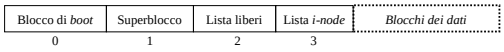


Esempi di chiamate di *File System*

- Disponibili all'utente solo indirettamente tramite incapsulazione in procedure di libreria
  - **`lseek`** : fissa l'indice di posizione all'interno di un *file* (come *offset* espresso in *byte* rispetto ad un riferimento dato) ← **accesso diretto**
  - **`stat`** : fornisce informazioni su *file* prelevandole dall'*i-node* corrispondente
    - Chiamata incapsulata dal comando **`stat`** di *shell*
      - Provare per esercizio dopo aver letto "**`man stat`**" ☺

Realizzazione del FS in UNIX – 1

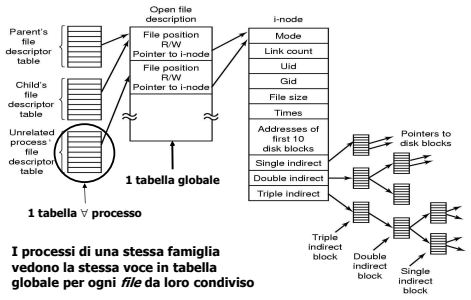
- **Struttura di partizione secondo UNIX v7**
- Il super-blocco (1) indica, tra l'altro, il # di *i-node* e di blocchi nel FS e fornisce il puntatore alla lista dei blocchi liberi (2)
- *i-node* (3) numerati 1..N, tutti di ampiezza $\geq$ 64 B
- *Directory* come insieme variabile e non ordinato di unità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



Realizzazione del FS in UNIX – 2

- 1 tabella di nucleo **per processo** contiene i descrittori dei *file* attualmente in uso al processo
  - A ogni descrittore deve corrispondere l'attuale posizione di R/W
  - Ogni processo deve avere il suo proprio indice di posizione sui propri *file* aperti ($\rightarrow$ più posizioni 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
- 1 tabella di nucleo **globale** mantiene la corrispondenza tra tutti i *file* aperti e i loro *i-node*
  - Ciascuna voce nella tabella di processo punta a 1 voce nella tabella globale che specifica diritti e posizione corrente nel *file*
    - La stessa voce $\forall$ *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

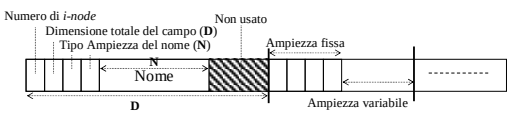
- L'*i-node* principale del *file* contiene (tra l'altro) l'indirizzo dei suoi primi 10 blocchi dati
  - 1 *i-node* ha la dimensione di 1 frazione di blocco
- Per *file* più grandi 1 campo dell'*i-node* principale punta a 1 *i-node* secondario che contiene puntatori ad altri blocchi dati
  - I-*node* 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*
- Può essere previsto anche un campo *triple-indirect*

Realizzazione del FS in UNIX – 5

- **Esempio 1** (campo *single-indirect*)
  - Blocco dati da 1 kB e *i-node* di uguale dimensione
  - Indirizzi di blocco su 4 B
  - Max dimensione di *file* = $(10 + 1\text{ kB} / 4\text{ B}) \times 1\text{ kB} = 266\text{ kB}$
- **Esempio 2** (campo *double-indirect*)
  - Stesse ipotesi di Esempio 1
    Max dimensione di *file* = $(10 + (1\text{ kB} / 4\text{ B})^2) \times 1\text{ kB} = 10\text{ kB} + 64\text{ MB}$
- **Esempio 3** (campo *triple-indirect*)
  - Per esercizio …

Realizzazione del FS in UNIX – 6

- 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



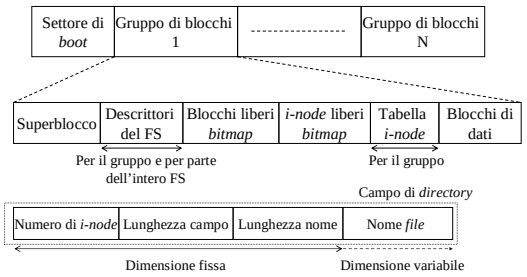
Realizzazione del FS in UNIX – 7

- 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)
    - Maggior complessità di gestione

Realizzazione del FS GNU/Linux – 1

- Inizialmente basato sul FS di MINIX però subito abbandonato per le eccessive limitazioni
  - MINIX: nomi ≤ 14 caratteri
  - MINIX: indirizzi di blocco = 2 B per blocchi ampi 1 KB
    - Ampiezza di *file* ≤ 64 MB (**perché?**)
- **`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
    - 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 → fino a (2^{32} = 4 G) blocchi
  - Blocchi di dimensione 1, 2, 4 kB scelta in fase di 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

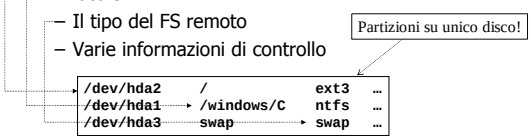
- Una *directory* **`/proc`** **virtuale** (non esistente su disco) contiene una *directory* ∀ processo presente nel sistema
  - Il nome della *directory* foglia è il PID del processo
  - Il contenuto della *directory* foglia è un insieme di *file* che descrivono il processo ed il suo ambiente
    - 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 ed **`ext2`**)
  - Modalità sostanzialmente analoga ad NFS (vedi seguito)

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* al quale i clienti possono fare accesso
  - 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
  - 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, disco di residenza, informazioni di controllo, numero di *i-node* della *directory* radice
    - Ogni accesso del cliente a *file* del FS importato userà tale 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 possibili inconsistenze
    - Lo stato di un *file* può cambiare tra due accessi remoti successivi
- Il controllo di accesso a *file* usa semplicemente diritti *owner*, *group*, *others*
  - Facilmente falsificabili se non autenticati
- Nessun supporto 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) od un ***r-node*** (per *file* remoti)
  - All'***r-node*** viene associato il descrittore di *file* remoto fornito dal *server* che ne esporta il FS
  - Traffico di rete 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 (*timer*)
      - 3 s. per blocchi dati e 30 s. per blocchi di *directory*
    - *Read-ahead* (sempre 8 kB in più sull'ultima lettura)
    - Scrittura differita al riempimento dell'unità di trasferimento

File System di rete (NFS) – 6