Filesystem

Filesystem-centrismo

- i sistemi UNIX sono “filesystem-centrici”: tutte le risorse hw/sw di sistema sono mappate all'interno del filesystem ed indirizzate come tali
- esempi:
  - disco ide /dev/hda
  - console /dev/tty1
  - bus PCI /proc/pci
  - power management /sys/power/state (>= 2.6.x)
  - prng /dev/random
- vantaggio: uniformità di accesso

File – metainformazioni

- ad ogni file mappato all'interno del filesystem corrisponde un insieme di metainformazioni
- sono accedibili mediante la system call stat (man 2 stat) e l'utility omonima del pacchetto GNU core utilities (man stat)
- comandi: stat, ls

- analizzeremo le caratteristiche dell'interfaccia offerta da sistemi operativi *nix al filesystem seguendo la traccia della system call stat

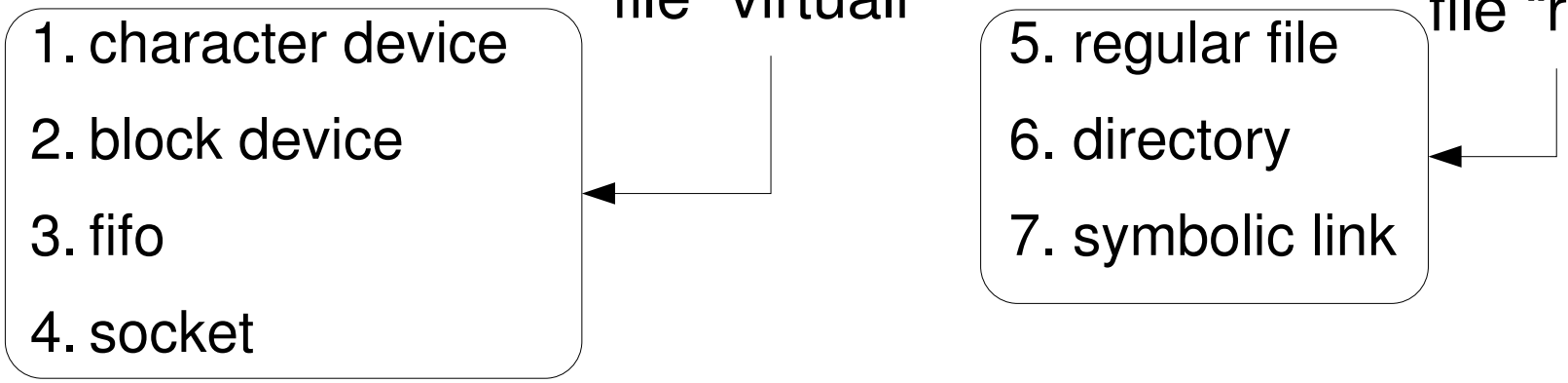
La struct “stat”

- da /usr/include/bits/stat.h:

```
struct stat {
    dev_t         st_dev;      /* device */
    ino_t         st_ino;      /* inode */
    mode_t        st_mode;     /* protection */
    nlink_t       st_nlink;    /* number of hard links */
    uid_t         st_uid;      /* user ID of owner */
    gid_t         st_gid;      /* group ID of owner */
    dev_t         st_rdev;     /* device type (if inode device) */
    off_t         st_size;     /* total size, in bytes */
    blksize_t     st_blksize;  /* blocksize for filesystem I/O */
    blkcnt_t      st_blocks;   /* number of blocks allocated */
    time_t        st_atime;    /* time of last access */
    time_t        st_mtime;    /* time of last modification */
    time_t        st_ctime;    /* time of last status change */
};
```

Tipologia di file (st_mode)

- per permettere di utilizzare l'astrazione di file sulle risorse di sistema, i filesystem UNIX supportano molteplici tipologie di file:



- per ottenere il tipo di file da st_mode sono disponibili le macro S_ISREG, S_ISDIR, S_ISCHR, ... (man 2 stat)

Proprietari (st_uid, st_gid)

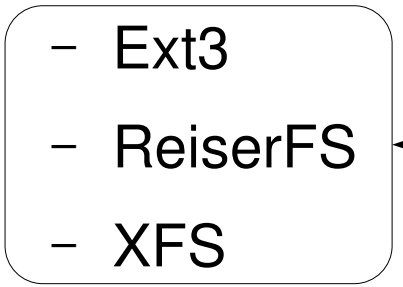
- ogni file presente sul filesystem è associato a:
  - un unico proprietario (campo st_uid)
  - un unico gruppo di appartenenza (campo st_gid)
- quando un file viene creato
  - l'owner corrisponde all'uid del processo in esecuzione
  - il gruppo corrisponde al gid del processo in esecuzione
    - è modificabile
    - dipende dal bit setgid della directory di appartenenza
- owner e gruppo possono essere modificati
- comandi: chgrp, chown, newgrp, sg

Permessi (st_mode)

- il campo st_mode contiene altre informazioni oltre al tipo di file, in particolare contiene i permessi associati al file
- i permessi sono divisi in 3 realm: proprietario del file (owner), membri del gruppo associato al file (group), altri utenti (other)
- ad ogni realm è associato un sottoinsieme delle seguenti capability: {read, write, execute}
- st_mode contiene inoltre i 3 bit: setuid, setgid, sticky
- comandi: chmod, umask

Organizzazione fisica

- i file delle tipologie che abbiamo denominato “reali” risiedono fisicamente su disco, divisi in blocchi
- la loro organizzazione fisica in blocchi dipende dal filesystem in uso
- alcuni dei filesystem più diffusi per sistemi GNU/Linux:
  - Ext2 file system
  - Ext3 journaled
  - ReiserFS
  - XFS

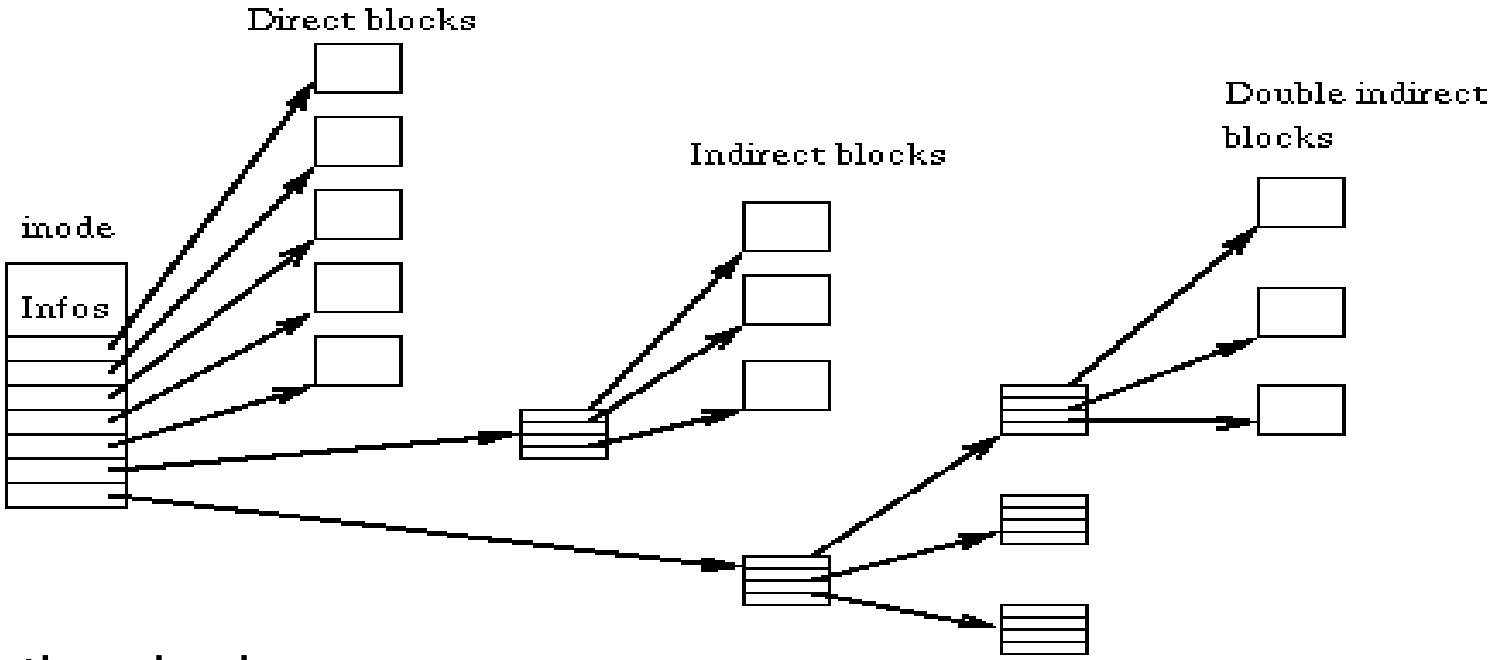


Extended Filesystem

- l'organizzazione fisica piu' diffusa per sistemi GNU/Linux e' quella del second extended filesystem (ext2), condivisa anche da ext3 (con l'aggiunta di un file esterno per il supporto del journaling)
- riferimenti:
  - APUE, cap. 4.14
  - *Design and Implementation of the Second Extended Filesystem*, Rémy Card et al., http://e2fsprogs.sourceforge.net/ext2intro.html

inode (st_ino)

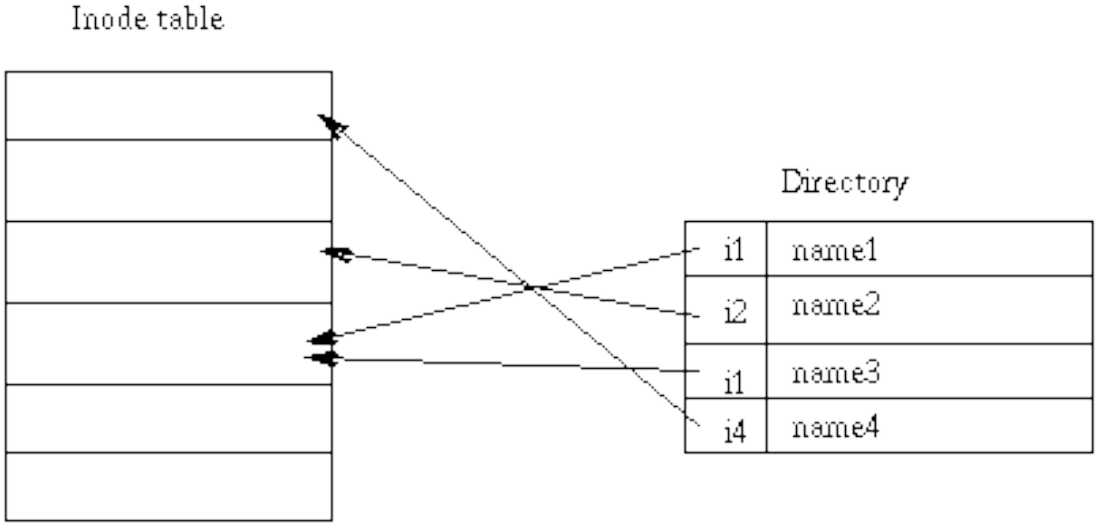
- ad ogni file è associata una struttura chiamata “inode”
  - il campo st_ino di stat è l'identificativo numerico di un inode



- motivazioni:
  - file piccoli sono acceduti random, file grandi sequenzialmente

Directory

- le directory sono semplici file, contenenti liste di coppie <nome, inode>
  - gli inode *non* contengono il nome del file
  - l'associazione <nome, inode> non è iniettiva

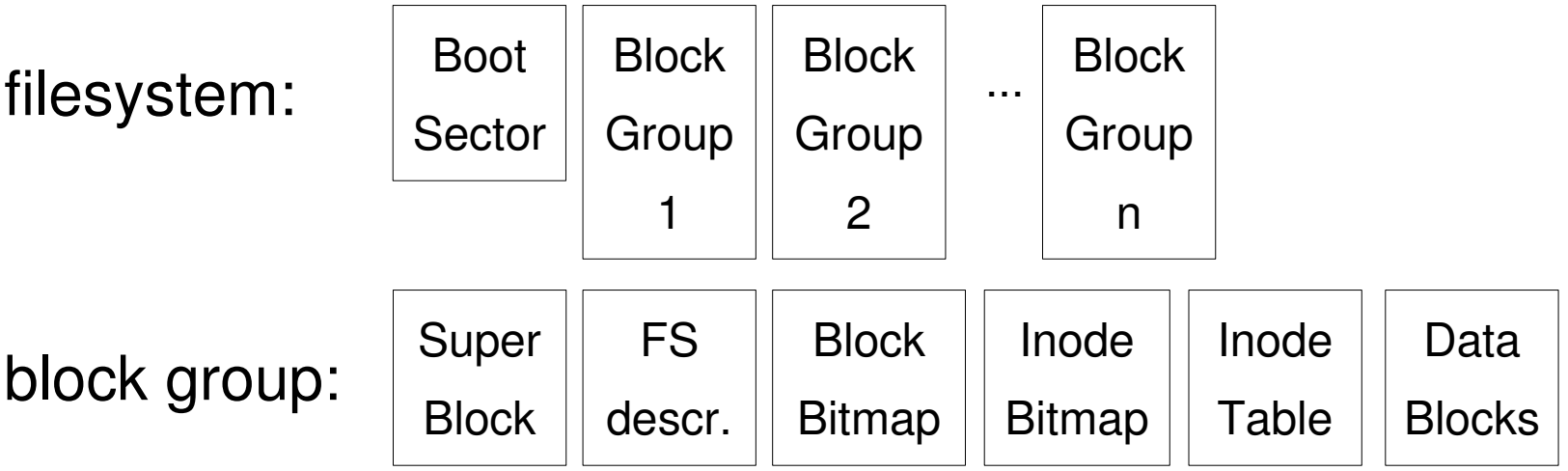


Devices

- sono file “virtuali”, utilizzati come punti di accesso a device driver del kernel
- sono identificati da una coppia
  - major number: identifica il tipo di device
    - e.g. disco IDE, scheda audio
  - minor number: identifica l'unità
    - e.g. n-esima partizione di un disco, mixer
- non occupano spazio sul filesystem (ma occupano inode)
- comandi: mknod, MAKEDEV

Ext2

- attualmente il più diffuso file system per sistemi GNU/Linux



- motivazioni:
  - minimizzare la distanza inode table – data blocks
- comandi: dumpe2fs, mke2fs, tune2fs

Tempi (st_{a,c,m}time)

- access time: aggiornato ogni qual volta il file viene acceduto (e.g., syscall read, execve)
- change time: aggiornato ogni qual volta le informazioni contenute nell'inode vengono cambiate (e.g., syscall chmod)
- modify time: aggiornato ogni qual volta il contenuto del file cambia (e.g., syscall write, truncate)
- è possibile cambiare manualmente questi tempi mediante la system call utime (man 2 utime)
- comandi: touch

Organizzazione logica

- i filesystem UNIX sono organizzati logicamente come DAG (Direct Acyclic Graph)
- la vista dell'utente su questo grafo corrisponde ad un unico albero generalizzato con la possibilità di creare link non circolari

Link

- un link è composto da una coppia (orientata) di nomi all'interno del file system <sorgente, destinazione>
- esistono due tipologie di link: hard link e symbolic link
- hard link
  - due o più directory (di uno stesso filesystem) contengono nomi che puntano ad un inode comune
  - ad ogni inode è associato il numero di hard link (st_nlink)
    - impostato a 1 quando il file viene creato
    - incrementato quando il file viene linkato
    - decrementato quando un link viene rimosso
    - l'inode viene liberato quando st_nlink = 0

Link

- symbolic link
  - alla sorgente del link corrisponde un file realmente esistente su disco il cui contenuto è il path destinazione del link
  - PRO
    - inter file system
    - late binding
  - CONS
    - un livello di indirezione
    - possibilità di dangling reference
- comandi: ln

Mount

- altri sistemi operativi (e.g. Windows) organizzano i filesystem in foreste di alberi generalizzati: un albero per ogni unità logica (partizioni, floppy, cd-rom, pendrive, ...)
- nei filesystem UNIX tutte le unità logiche vengono mappate in sottoalberi
- l'azione di mappare il filesystem presente su di una unità logica ad un sottoalbero prende il nome di “mounting” (“unmounting” è l'azione inversa)
- “mount point” è il punto del DAG ove il filesystem viene mappato

Mount

- al boot il kernel mappa un device come radice del DAG
- gli altri sottoalberi da mappare durante il boot sono specificati nella tabella dei filesystem (/etc/fstab)
- mounting ed unmounting di altri device possono essere effettuati a runtime
- comandi: rdev, mount, umount, sync, df

File Hierarchy Standard

- per favorire l'interoperabilità tra sistemi UNIX-like, è nato un gruppo di standardizzazione della gerarchia del filesystem
- FHS è lo standard risultante: http://www.pathname.com/fhs/
- caratteristiche considerate
  - file statici vs file variabili
  - file condivisibili vs file non condivisibili
  - file necessari all'avvio vs file non necessari all'avvio
- riferimenti:
  - Filesystem Hierarchy Standard 2.3, 29/01/2004
  - appunti, cap. 99
  - Debian Policy Manual, cap 9.1, http://www.debian.org/doc/debian-policy/

FHS

- la radice “/” contiene

/bin/	binari essenziali;
/boot/	file statici per l'avvio del sistema;
/dev/	file di dispositivo;
/etc/	configurazione particolare del sistema;
/home/	directory personali degli utenti;
/lib/	librerie essenziali e moduli del kernel;
/media/	mount point (plurale) per media rimuovibili;
/mnt/	punti di innesto temporanei;
/opt/	applicativi aggiuntivi;
/root/	directory personale dell'utente root;
/sbin/	binari essenziali (competenza di root);
/srv/	data per servizi dalla macchina corrente
/tmp/	file e directory temporanei;
/usr/	gerarchia secondaria;
/var/	dati variabili.

FHS

- /bin/ ed /sbin/
  - eseguibili essenziali
  - distinzione tra le necessità degli utenti e dell'amministratore
- /boot/
  - file necessari al boot, di competenza del boot loader
  - storicamente montata sotto al 1024-esimo cilindro
- /dev/
  - file di dispositivo e MAKEDEV
  - nei kernel recenti è un filesystem virtuale (devfs/udev)
- /etc/
  - configurazione di sistema
  - configurazioni delle applicazioni di sistema

FHS

- /lib/
  - moduli del kernel
  - librerie dinamiche essenziali
    - necessarie agli eseguibili in /bin/ e /sbin/
- /media/
  - mount point per media rimuovibili (/media/cdrom, ...)
- /mnt/
  - mount point (N.B. e' **un** mount point)
- /opt/
  - sotto-gerarchie per-applicazione:
    - /opt/<*appname*>/bin/
    - /opt/<*appname*>/etc/

FHS

- /proc/
  - mount point del filesystem virtuale proc
    - informazioni sul kernel e sui processi in esecuzione
- /root/
  - home directory dell'utente root
- /srv/
  - dati per servizi locali
- /tmp/
  - file temporanei
  - spesso mount point di un filesystem residente in memoria (e.g. tmpfs)
  - permessi “rwxrwxrwt” (leggibile/scrivibile al mondo, sticky bit)

FHS

- /usr/
  - gerarchia delle applicazioni di sistema
  - a differenza di /opt/ tutti gli applicativi installati in /usr/ condividono la stessa gerarchia
    - ad eccezione di /usr/X11R6/
  - tipicamente gestita dal sistema di pacchettizzazione delle distribuzioni GNU/Linux (e.g. dpkg, rpm)
  - contiene dati statici e condivisibili
- /usr/bin/, /usr/sbin/, /usr/lib/
  - eseguibili e librerie dinamiche non essenziali

FHS

- /usr/local/
  - programmi locali *non* gestiti dal sistema di pacchettizzazione
  - sotto-gerarchia simile a /usr/
    - /usr/local/bin/
    - /usr/local/etc/
    - /usr/local/...
- /usr/share/
  - file statici, condivisibili ed *indipendenti* dall'architettura
  - e.g. /usr/share/man/
- /usr/src/
  - sorgenti
  - e.g. /usr/src/linux/

FHS

- /var/
  - dati variabili (i.e. tutto ciò che non può stare in /usr/)
- /var/cache/ dati transitori
- /var/log/ log di sistema
- /var/mail/ mailbox degli utenti
- /var/spool/ code (e.g. mail, stampa)
- /var/run/ dati dei programmi in esecuzione (e.g. pid)

Partizionamento

- partizionamento *minimale*
  - /
  - partizione di swap
- esempio di /etc/fstab:

```
# fs      mount point   type       options           dump   pass
/dev/hda1    /          ext3       errors=remount-ro   0      1
/dev/hda2    none       swap       sw                  0      0
proc         /proc      proc       defaults            0      0
```

Partizionamento

- partizionamento *consigliato*
  - /
  - partizione di swap
- esempio di /etc/fstab:

```
# fs      mount point   type       options           dump   pass
/dev/hda1       /          ext3       errors=remount-ro       0
   1
/dev/hda2       none       swap       sw              0       0
/dev/hda3       /usr       ext3       rw              0       2
/dev/hda5       /home      ext3       rw              0       2
/dev/hda6       /var       ext3       rw              0       2
tmpfs           /tmp       tmpfs      size=30%        0       0
proc            /proc      proc       defaults        0       0
```