Master in Tecnologia del

Software Libero e Open Source

A.A. 2004-2005

Laboratorio di Sistemi Operativi

Stefano Zacchiroli

Mi presento

- *Stefano Zacchiroli*
  - email: zack@cs.unibo.it web: http://www.cs.unibo.it/~zacchiro
  - studente di dottorato
    - Università degli Studi di Bologna, Dipartimento di Scienze dell'Informazione
    - Mathematical Knowledge Management
  - sviluppatore Debian GNU/Linux
    - maintainer di pacchetti relativi al linguaggio OCaml
  - queste slide: http://www.cs.unibo.it/~zacchiro/courses/mfosset0405/

Riferimenti

- *Appunti di informatica libera*, Daniele Giacomini, http://a2.pluto.it/ (GPL) [“appunti”]
- *Unix Power Tools*, Shelly Powers et al., O'Reilly
- *Programming Perl*, Larry Wall et al., O'Reilly
- *Advanced Programming in the UNIX Environment*, W. Richard Stevens, Addison Wesley [“APUE”]
- *GaPiL (Guida alla Programmazione in Linux)*, Simone Piccardi, http://gapil.firenze.linux.it/ (GFDL)

Nozioni di base

- utilizzo di base di un sistema *nix
  - gestione del filesystem (ls, cp, du, cat, mkdir, ...)
  - utenti, gruppi e permessi (su, chmod, chown, ...)
  - utilizzo di un editor di testo (vi, emacs, pico, ...)
  - archiviazione e (ri)compilazione (tar & make)
- riferimenti:
  - appunti, cap. 5 (teoria) & 7 (esercizi pratici)

Kernel Linux

Kernel Linux – distribuzione

- un po' di numeri (per i kernel della famiglia 2.6.x)
  - mole: ~ 320 Mb di sorgenti per ~ 4'500'000 LOC
  - costo dello sviluppo: ~ 4'500 anni/uomo, ~ 600'000'000 $ (fonte:http://www.dwheeler.com/essays/linux-kernel-cost.html)
- distribuito su http://www.kernel.org
  - tarball da 30 Mb contenenti sorgenti C (portabili su almeno 16 architetture ad oggi) e Assembly (architecture specific)
  - interamente (up to firmware ...) rilasciato sotto licenza GPL
  - compilabile utilizzando utility GNU (GCC, Make, Binutils)

Kernel Linux – installazione

- è un kernel *monolitico* (i.e. non microkernel), ma *modulare*
- l'installazione consta di:
  1. parte monolitica
  2. moduli
  3. symbol table
- riferimenti:
  - appunti, capp. 48—50
  - The Linux Kernel HOWTO (vecchio)
  - Linux Loadable Kernel Module HOWTO
  - The System.map file http://www.dirac.org/linux/system.map/

Kernel Linux – parte monolitica

- non risiede necessariamente sul filesystem, ma deve essere nota al boot loader (e.g. lilo, grub, yaboot, ...)
  - il boot loader è l'applicazione che si preoccupa di “installare” il kernel in modo che la procedura di bootstrap della macchina (tipicamente implementata nel BIOS) sia in grado di accedervi
    - è fortemente dipendente dall'architettura
    - esempi: lilo & grub (i386), yaboot (powerpc)
- quando risiede sul filesystem è tipicamente un unico file /boot/vmlinu{x,z}-*<kernel-version>* (e.g. /boot/vmlinux-2.6.8)
- viene caricata in memoria al boot e mai scaricata

Kernel Linux – moduli

- parti di kernel caricati/scaricati on demand, manualmente dall'utente o da demoni preposti (kerneld/kmod)
- devono essere accessibili a runtime, pertanto risiedono all'interno del filesystem
  - un albero di file oggetto (.o/.ko) radicato in /lib/modules/*<kernel-version>*/ (e.g. /lib/modules/2.6.8/)
- flessibilità dei moduli
  - name aliasing ed opzioni (/etc/modules.conf)
  - dipendenze inter-modulo (/lib/modules/.../modules.dep)

Kernel Linux – gestione dei moduli

- caricamento on demand
  - insmod (full path, non gestisce alias e dipendenze)
  - modprobe (nome modulo, gestisce alias e dipendenze)
- rimozione: rmmod, modprobe (gestisce dipendenze)
- ispezione: lsmod
- calcolo delle dipendenze: depmod
- manutenzione opzioni ed alias: update-modules

Kernel Linux – symbol table

- /boot/System.map-*<kernel-version>*
- contiene la lista dei simboli associata al kernel, nel formato di nm: <symbol value, symbol type, symbol name>
- viene utilizzata da alcuni programmi di sistema (e.g. klogd, ps, lsof) per la risoluzione da symbol value a symbol name
  - e.g. in caso di kernel oops (“segfault” in kernel space), il kernel dispone solamente del valore corrente dell'EIP (instruction pointer). klogd intercetta l'oops ed utilizza System.map per ottenere il symbol name corrispondente all'EIP e loggarlo via syslog
- comandi: nm

Kernel Linux – configurazione

- configurazione, compilazione ed installazione del kernel sono gestite dal Makefile distribuito assieme al kernel
- configurazione
  - target del Makefile: config, xconfig, menuconfig, oldconfig, gconfig (solo 2.6.x)
  - ogni componente del kernel può essere:
    - compilata ed inclusa nella parte monolitica
    - compilata come modulo
    - non compilata
  - vengono gestite dipendenze inter-componente
  - risiede nel file .config all'interno dell'albero dei sorgenti

Kernel Linux – compilazione & installazione

- compilazione
  - target del Makefile
    - pulizia: clean, mrproper
    - compilazione parte monolitica: bzImage
    - compilazione moduli: modules
  - risultati della compilazione
    - arch/<*architecture*>/boot/bzImage + file (.o/.ko) per i moduli
- installazione
  - su disco: fortemente dipendente dal bootloader
  - su floppy: cp ..../bzImage /dev/fd0
  - moduli: target modules_install del Makefile

UML – User Mode Linux

- UML è una macchina virtuale che implementa il kernel linux in user space su una macchina host linux
- motivazioni: sandboxing, kernel hacking, insegnamento
- attualmente supporta i kernel della famiglia 2.4.x su macchine host i386
- riferimenti:
  - http://user-mode-linux.sourceforge.net
    - User Mode Linux HOWTO
    - *A user-mode port of the Linux kernel*, Jeff Dike, http://user-mode-linux.sourceforge.net/als2000/index.html

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

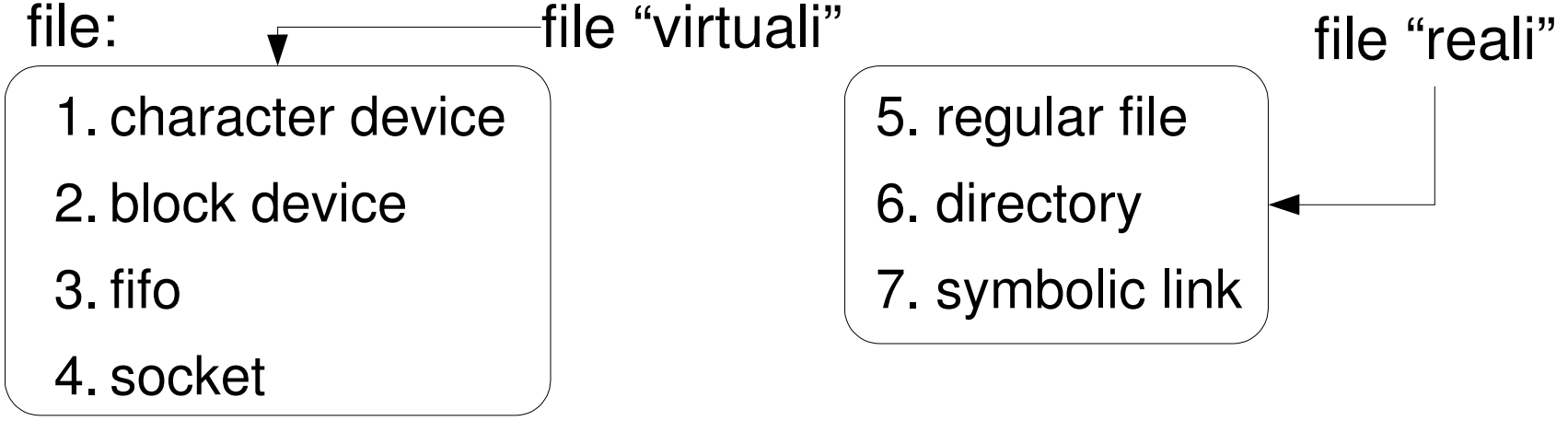
La struct “stat”

- 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)

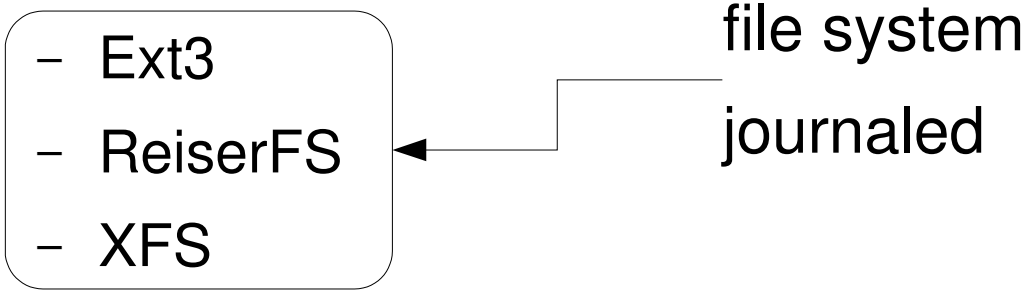
- ogni file presente sul filesystem è associato a:
  - un unico proprietario (campo st_uid)
  - un unico gruppo di appartenenza (campo st_gid)
- quando un file viene creato
  - l'owner corrisponde all'uid del processo in esecuzione
  - il gruppo corrisponde al gid del processo in esecuzione
    - è modificabile
    - 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
  - Ext3
  - ReiserFS
  - XFS

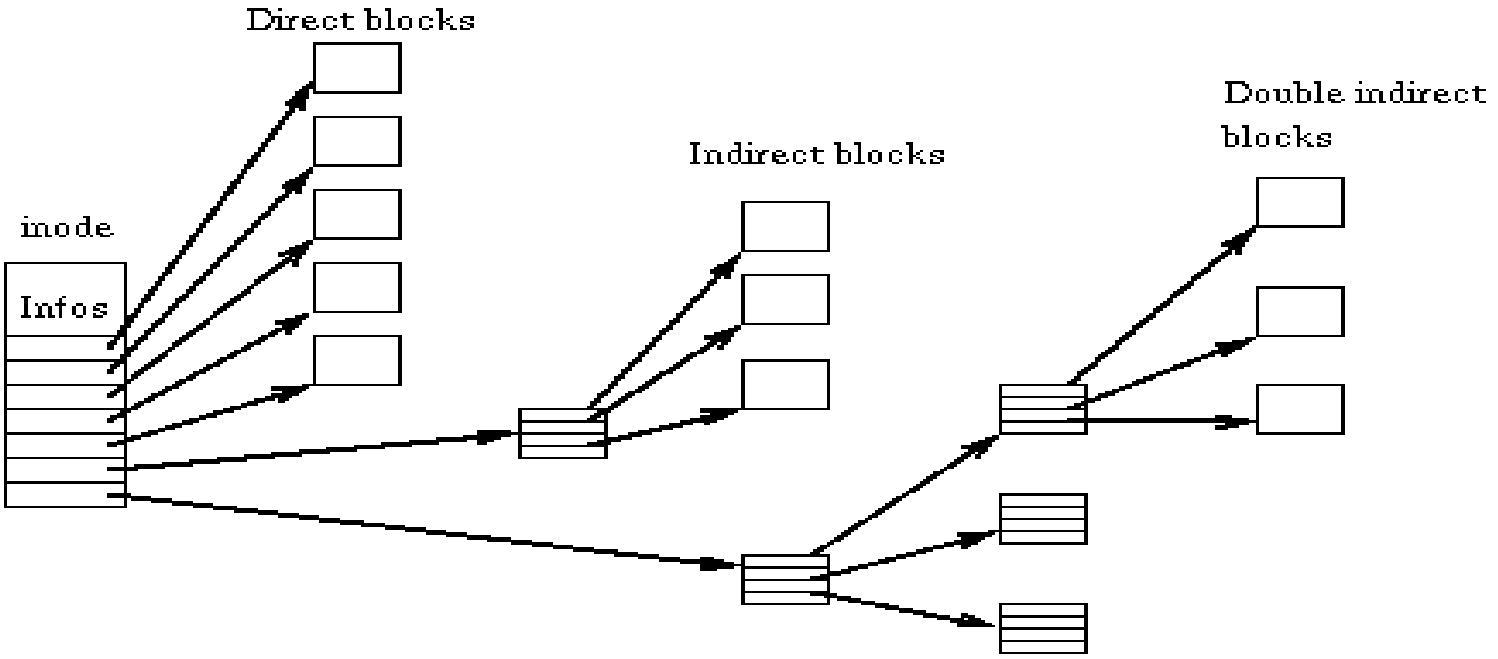


Filesystem Unix

- 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)

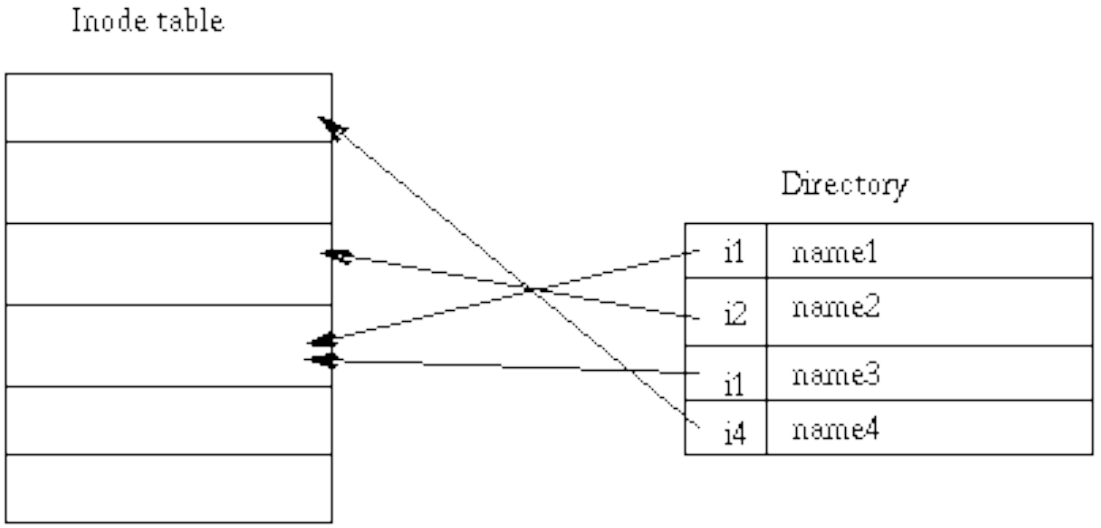
- ad ogni file è associata una struttura chiamata “inode”
  - il campo st_ino di stat è l'identificativo numerico di un inode



- 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



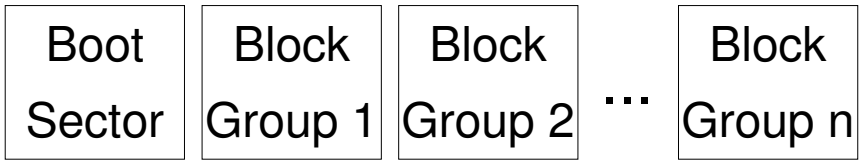
Devices

- 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

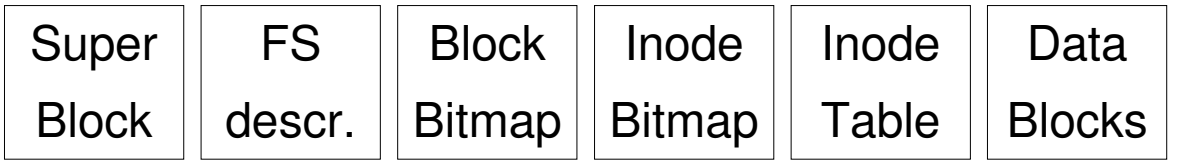
Ext2

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

filesystem:



block group:



- 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 (system call: read, execve)
- change time: aggiornato ogni qual volta le informazioni contenute nell'inode vengono cambiate
- modify time: aggiornato ogni qual volta il contenuto del file cambia (system call: 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 la 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 “/”

/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;
/mnt/	punti di innesto temporanei;
/opt/	applicativi aggiuntivi;
/root/	directory personale dell'utente root;
/sbin/	binari essenziali (competenza di root);
/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/)
- /mnt/
  - mount point
- /opt/
  - sotto-gerarchie per-applicazione:
    - /opt/*<appname>*/bin/
    - /opt/*<appname>*/etc/
    - /opt/*<appname>*/...

FHS

- /proc/
  - mount point del filesystem virtuale proc
    - informazioni sul kernel e sui processi in esecuzione
- /root/
  - home directory dell'utente root
- /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 *minimo*
  - /
  - 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
```

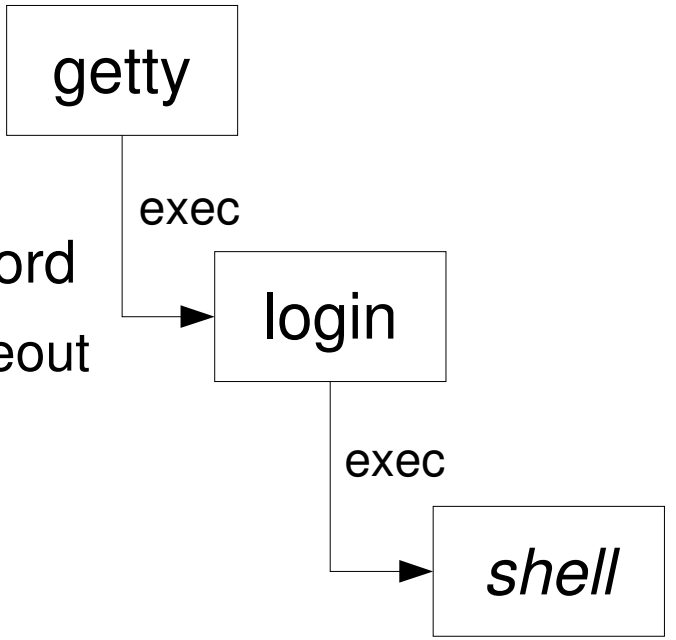
Utenza

Utenza

- Riferimenti:
  - appunti, capp. 79 – 82
  - APUE, cap. 9.2
  - Michael H. Jackson, Linux Shadow Password HOWTO
  - The Linux-PAM System Administrators' Guide, Andrew G. Morgan, http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam.html

Login

- procedura di login
  1. getty in attesa su tty di uno username
  2. getty esegue login che verifica la password
     - se la password è errata ritenta dopo timeout
     - se l'utente è root
       - verifica che tty sia in /etc/securetty
     - se l'utente non è root
       - verifica che non esista /etc/nologin
  3. login stampa le informazioni di login (lastlog, mail, /etc/motd)
  4. login registra il successo del login nel registro di sistema (log)
  5. login passa alla home directory dell'utente (oppure /) ed esegue la sua shell (oppure /bin/sh)



Il registro degli account

- risiede nel file /etc/passwd
- è un database testuale di record a 7 campi, separati da ":"
  1. nome utente
  2. password (crittata)
  3. uid
  4. gid
  5. nome completo
  6. home directory
  7. shell
- esempio di record (1 riga di /etc/passwd)

```
zack:bGqudB41a4a:3148:3148:Stefano Zacchiroli,,,:
/home/zack:/bin/bash
```

- riferimenti
  - man 5 passwd
- comandi: passwd, su, id

Il registro degli account

1. molte applicazioni hanno necessità di accedervi
   - e.g. gli inode contengono l'uid, ma non lo user name, ls necessita di /etc/passwd per mostrarlo!
2. in caso di reti che condividono utenti, tutte le macchine necessitano di acceso a /etc/passwd
3. le password sono crittate con crypt (man 3 crypt), è un algoritmo basato su DES, suscettibile a brute force attack

- l'azione combinata di 1, 2 e 3 comporta problematiche di sicurezza
- soluzione: shadow password

Shadow password

- osservazione: il campo password di /etc/passwd è riservato e necessario durante il login, gli altri campi non sono riservati e sono necessari anche dopo il login
- utilizzando le shadow password il campo password viene spostato da /etc/passwd in /etc/shadow
- /etc/shadow non è accedibile agli utenti di sistema
- quando le shadow password sono in uso, il campo password di /etc/passwd viene settato ad “x”
- altri vantaggi delle shadow password
  - gestione degli expire time

Il registro dei gruppi

- risiede nel file /etc/group
- il formato è analogo a quello di /etc/passwd, con i campi seguenti
  1. group name
  2. password
  3. group id
  4. utenti membri
- le password dei gruppi sono utilizzate raramente, nel caso siano utilizzate il file /etc/gshadow fornisce funzionalità analoghe a quelle di /etc/shadow
- comandi: newgrp

Creazione utenti

- procedura di creazione di nuovi utenti
  1. aggiunta utente a /etc/passwd
  2. aggiunta gruppo a /etc/group
  3. aggiunta password a /etc/shadow, /etc/gshadow
  4. creazione home directory
- adduser
  - automatizza la procedera di creazione nuovi utenti
  - gestisce scheletri di home directory (/etc/skel/)
  - gestisce la creazione di utenti di sistema
  - /etc/adduser.conf
- comandi: adduser, addgroup

Eliminazione utenti

- procedura di eliminazione di utenti
  - è inversa alla procedura di creazione
  - in aggiunta può essere necessario cancellare tutti i file dell'utente (che non risiedono necessariamente nella sua home)
    - find / -user *<username>*
- deluser
  - automatizza la procedura di eliminazione di utenti
  - supporta la rimozione di tutti i file dell'utente
  - /etc/deluser.conf
- comandi: deluser, delgroup

Pluggable Authentication Modules

- inizialmente, login era l'unico programma di autenticazione e la password era l'unico meccanismo disponibile
- nei moderni sistemi GNU/Linux altri programmi hanno necessità di autenticare gli utenti ...
  - e.g.: sshd, pppd, gdm/xdm/kdm, xlock, ...
- ... ed meccanismi diversi dalla password sono disponibili
  - e.g.: autenticazione RSA/DSA di ssh
- PAM permette di utilizzare molteplici *servizi di autenticazione*
  - configurabili ed associabili alle applicazioni senza dovere ricompilare queste ultime
  - la scelta e la configurazione dei servizi di autenticazione sono effettuate dall'amministratore, non dallo sviluppatore

PAM – installazione

- PAM è modulare, diviso in librerie dinamiche
- installazione tipica:
  - /lib/security/ moduli PAM (e.g. /lib/security/pam_unix.so)
  - /etc/pam.d/ configurazione applicazioni che utilizzano PAM (e.g. /etc/pam.d/login)
- i file di configurazioni sono file testuali composti da record a 4 campi: <module-type, control-flag, module-path, arguments>
- ogni file di configurazione lista i moduli PAM utilizzati da una data applicazione
- riferimenti:
  - man 7 pam

PAM – module type

- module-type
  1. *auth* verifica l'identità dell'utente (e.g. password)
  2. *account* verifica lo stato dell'account dell'utente (e.g. è scaduto?)
  3. *password* controlli aggiuntivi sulla password (se prevista)
  4. *session* login/logout hook

PAM – control flag

- intuitivamente, ogni modulo PAM corrisponde ad una funzione, il cui valore di ritorno è ternario:
  - SUCCESS, IGNORE, FAILURE
- i moduli listati in un file di configurazione vengono eseguiti sequenzialmente
- il control flag stabilisce la politica a riguardo del valore di ritorno richiesto per un dato modulo
  - *requisite* richiede SUCCESS/IGNORE, termina se FAILURE
  - *required* richiede SUCCESS/IGNORE, ma non termina
  - *sufficient* in caso di SUCCESS, termina la procedura
  - *optional* tratta FAILURE come IGNORE
- in caso di soli valori IGNORE non viene consentito l'accesso

PAM – esempio

- /etc/pam.d/login

```
auth        requisite   pam_securetty.so
auth        required    pam_nologin.so
auth        required    pam_env.so
auth        required    pam_unix.so nullok
account     required    pam_unix.so
session     required    pam_unix.so
session     optional    pam_lastlog.so
session     optional    pam_motd.so
session     optional    pam_mail.so standard noenv
password    required    pam_unix.so nullok obscure min=4 max=8
```