

Introduzione a Unix e GNU/Linux

Introduzione a Unix e GNU/Linux

Autore: Michael Opdenacker

Free Electrons <http://free-electrons.com>

Traduzione: Gianluca Moro

<http://www.giammy.com/>



Ringraziamenti

- ▶ Al progetto OpenOffice.org, per i loro ottimi strumenti di presentazione e di word processing.
- ▶ Alla comunità Handhelds.org, per avermi dato il loro aiuto e l'opportunità di aiutare.
- ▶ A tutti i membri della comunità Free Software e Open Source, per aver condiviso il meglio di sé: il loro lavoro, le loro conoscenze, la loro amicizia.
- ▶ Alle persone che hanno inviato commenti e correzioni: Jeff Ghislain, Leif Thande, Frédéric Desmoulins, Przemysław Ciesielski



Diritti di riproduzione



COMMONS DEED

Attribuzione - Condividi allo stesso modo 2.0

Tu Sei libero:

- di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire o recitare l'opera
- di creare opere derivate
- di usare l'opera a fini commerciali

Alle seguenti condizioni



Attribuzione. Devi riconoscere il contributo dell'autore originario.



Condividi allo stesso modo. Se alteri, trasformi o sviluppi quest'opera, puoi distribuire l'opera risultante solo per mezzo di una licenza identica a questa.

- In ogni atto di riutilizzazione o distribuzione, devi chiarire agli altri i termini della licenza di quest'opera.
- Se ottieni il permesso dal titolare del diritto d'autore, è possibile rinunciare ad ognuna di queste condizioni.

Le tue utilizzazioni libere e gli altri diritti non sono in nessun modo limitati da quanto sopra

Licenza: <http://creativecommons.org/licenses/by-sa/2.0/legalcode>

© Copyright 2006-2004

Michael Opdenacker

michael@free-electrons.com

Sorgenti, aggiornamenti e traduzioni

http://free-electrons.com/training/intro_unix_linux

Correzioni, suggerimenti, contributi e traduzioni sono i benvenuti!



Storia del Documento

Se non diversamente specificato, i contributi sono di Michael Opdenacker

- ▶ 28 Lug. 2005. Traduzione di Gianluca Moro
- ▶ 14 Giu. 2005. Ultimo aggiornamento, correzioni e miglioramenti minori
- ▶ 6 Dic. 2004. Nuova sezione di amministrazione di sistema per principianti, ed alcune modifiche.
- ▶ 28 Sett. 2004. Prima edizione pubblica.



Cos'è il presente documento

- ▶ Scopo di questo documento è di essere usato come supporto visuale in una presentazione o in una lezione: è solo un riassunto o un complemento a ciò che è detto. Quindi le spiegazioni possono non essere esaustive.
- ▶ Comunque questo documento intende anche essere un riferimento per il pubblico. Inoltre è indicato per lettori autodidatti. Così spesso si va più in dettaglio, rendendo il documento esteticamente meno accattivante.



Contenuto delle lezioni (1)

Introduzione

- ▶ Storia dello Unix
- ▶ Unix: filosofia e caratteristiche
- ▶ I vari livelli in un sistema Unix
- ▶ Il progetto GNU, la licenza GPL
- ▶ Linux, Distribuzioni
- ▶ Altri sistemi Unix liberi



Contenuto delle lezioni (2)

Shell, filesystem e gestione file

- ▶ Interpreti della linea di comando
- ▶ Struttura del filesystem di Unix
- ▶ Gestione file e directory
- ▶ Vedere, gestire e ordinare file
- ▶ Link simbolici e hard
- ▶ Diritti di accesso ai file



Contenuto delle lezioni (3)

Standard I/O, ridirezione e pipe

- ▶ Standard input e output
- ▶ Ridirezionare lo standard input e output su file
- ▶ Pipe: ridirezionare lo standard output ad altri comandi
- ▶ Standard error



Contenuto delle lezioni (4)

Controllo dei processi

- ▶ Unix: multitask dall'inizio
- ▶ Esecuzione in background, sospensione,, ripresa e terminazione
- ▶ Lista dei processi attivi
- ▶ Terminazione di 1 o più processi
- ▶ Variabili d'ambiente
- ▶ Variabile d'ambiente PATH
- ▶ Alias nella shell, file `.bashrc`



Contenuto delle lezioni (5)

Varie

- ▶ Editor di testo
- ▶ Compressione e archiviazione
- ▶ Stampa di file
- ▶ Confronto tra file
- ▶ Ricerca di file
- ▶ Ottenere informazioni sugli utenti



Contenuto delle lezioni (6)

Elementi base di gestione del sistema

- ▶ Varie: proprietà dei file, spegnimento ...
- ▶ Configurazione della rete
- ▶ Filesystem: crearli e montarli

Un passo avanti

- ▶ Trovare aiuto: accedere alla pagine del manuale in linea
- ▶ Cercare risorse in internet
- ▶ Usare GNU/Linux a casa

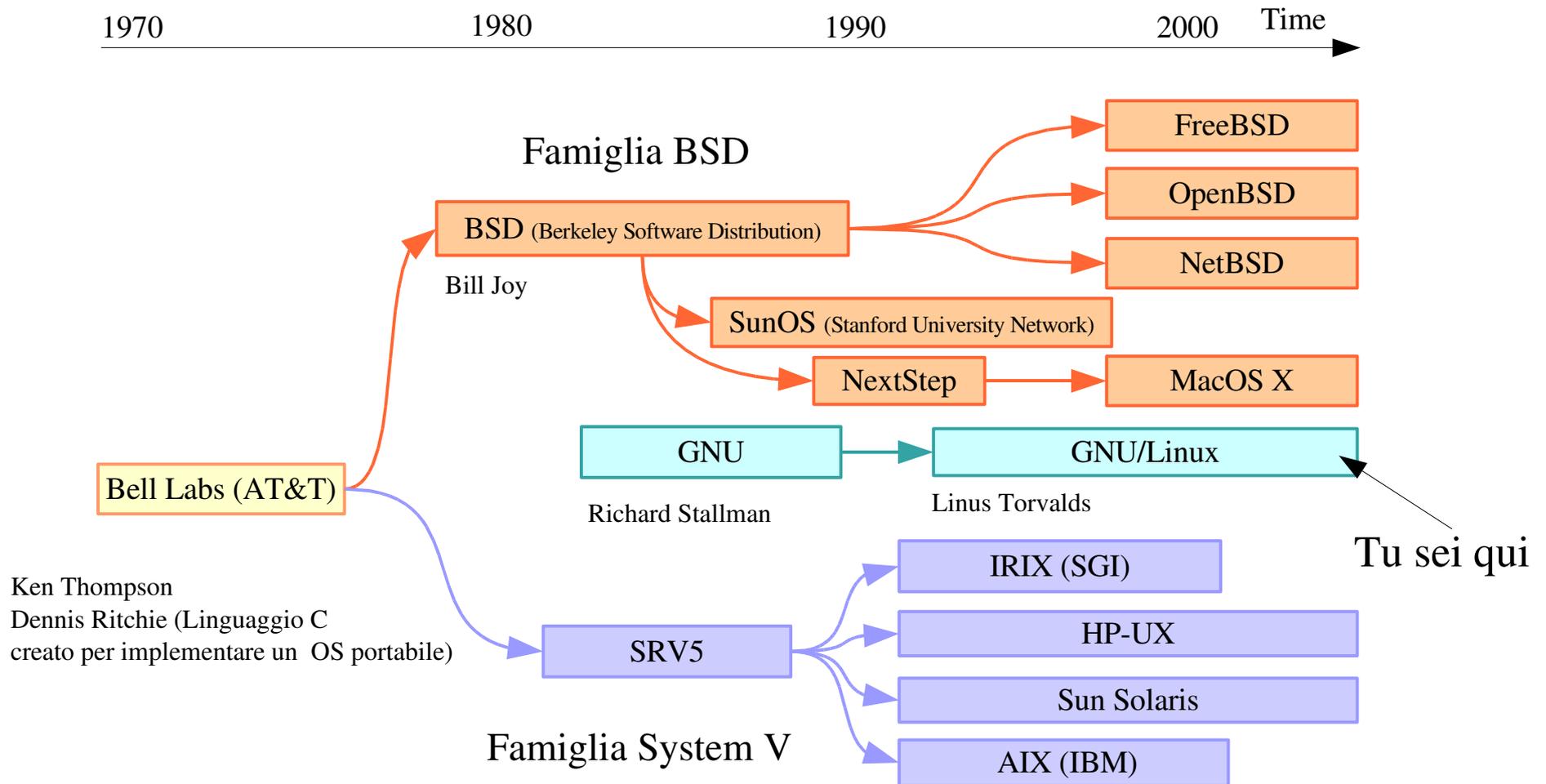


Introduzione a Unix e GNU/Linux

Introduzione



Albero genealogico dello Unix



Architettura del sistema Unix

Applicazioni Grafiche

Browser Web, ufficio, multimedia...



Applicazioni da linea di comando

ls, mkdir, wget, ssh, gcc, busybox...



Librerie condivise

libjpeg, libstdc++, libxml...

Libreria C

Libreria GNU C, uClibc...

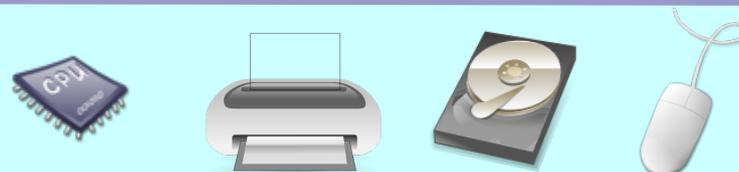


Kernel del Sistema operativo

Linux, Hurd...



Hardware e periferiche



Spazio Utente

Spazio Kernel

Hardware



La filosofia di Unix

I sistemi più potenti di oggi sono basati su progetti di 35 anni fa!

- ▶ Piccolo è bello
- ▶ Fa fare ad ogni programma una sola cosa, ma bene
- ▶ Preferire la portabilità all'efficienza
- ▶ Evitare interfacce utenti limitanti

Astrazioni di sistema

- ▶ Kernel: livello hardware
- ▶ Shell: livello in modalità testo
- ▶ X Windows: livello GUI



Principali caratteristiche di Unix

Unix è stato creato per grandi sistemi multiutente

- ▶ Multiutente e sicuro:
Utenti normali non possono modificare i file degli altri (di default)
In particolare, utenti normali non possono modificare impostazioni di sistema, né cancellare programmi, etc.
- ▶ **root**: utente amministratore con tutti i privilegi
- ▶ Preemptive multi-tasking
- ▶ Supporto per processori multipli
- ▶ Estremamente flessibile
- ▶ Supporto della rete
- ▶ Portabilità
- ▶ Scalabilità



Il Progetto GNU

GNU = GNU Non è Unix (un acronimo ricorsivo!)

- ▶ Progetto per implementare un sistema operativo come Unix completamente libero
- ▶ Iniziato da Richard Stallman nel 1984, un ricercatore del MIT, in un periodo in cui i sorgenti di Unix non erano più liberi
- ▶ Componenti iniziali: compilatore C (gcc), make (GNU make), Emacs, libreria C (glibc), coreutils (ls, cp ...)
- ▶ Comunque, nel 1991, il progetto GNU non aveva ancora un kernel e veniva eseguito su macchine Unix proprietarie.



Software Libero

Il Software Libero garantisce le seguenti 4 libertà all'utente:

- ▶ La libertà di eseguire un programma, per ogni scopo
- ▶ La libertà di studiare come il programma funziona, e adattarlo ai suoi bisogni
- ▶ La libertà di ridistribuire copie per aiutare gli altri
- ▶ La libertà di migliorare il programma, e rilasciare i propri miglioramenti al pubblico

Vedi <http://www.gnu.org/philosophy/free-sw.html>



Licenze di software libero tipo BSD

- ▶ Naturalmente garantisce le 4 libertà agli utenti
- ▶ Consente di scrivere software proprietario
- ▶ Licenze di esempio: BSD, Apache

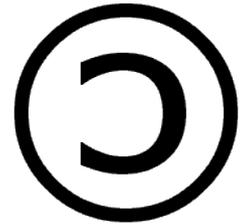


La Licenza GNU (GPL)

Il maggior contributo del progetto GNU!

- ▶ Le Licenze *Copyleft* usano le leggi del copyright per consentire all'autore di chiedere che versioni modificate siano anch'esse software libero.

<http://www.gnu.org/copyleft/copyleft.html>



- ▶ La GNU GPL chiede che modifiche e lavori derivati siano GPL:
 - ▶ Si applica solo a software **rilasciato**
 - ▶ Ogni programma che usa codice GPL (sia con link statici che dinamici) è considerato una estensione del codice

GPL FAQ: <http://www.gnu.org/licenses/gpl-faq.html>



GNU Lesser General Public License

<http://www.gnu.org/copyleft/lesser.html>

- ▶ La licenza Copyleft è simile alla GNU GPL:
Le modifiche devono essere condivise alle stesse condizioni
- ▶ Consente il linkaggio con moduli non liberi
- ▶ È usata da parecchie librerie di Software libero. Esempi:
glibc, GTK, Wine, SDL



Software Libero e Open Source

Il Movimento del Software libero

- ▶ Approccio centrato sull'obiettivo
- ▶ Orientato alla libertà individuale e all'utilità sociale della cooperazione.
Vedi: <http://www.gnu.org/philosophy/free-software-for-freedom.html>

Il Movimento Open Source

- ▶ Approccio pragmatico
- ▶ Evidenzia principalmente i vantaggi della condivisione dei sorgenti e fa scelte basate sulla superiorità tecnica.

Sebbene i motivi di base siano diversi, entrambi i movimenti lavorano assieme e cooperano molto bene!



Linux

- ▶ Kernel libero, tipo Unix creato nel 1991 da Linus Torvalds
- ▶ L'intero sistema usa i tool della GNU: libreria C, gcc, binutils, fileutils, make, emacs...
- ▶ Così l'intero sistema si chiama “GNU/Linux”
- ▶ Condiviso dall'inizio come software libero (licenza GPL), ha attirato sempre più contributi e utenti.
- ▶ Dal 1991, sta crescendo più velocemente di qualsiasi altro sistema operativo.



Distribuzioni GNU/Linux

- ▶ Si occupano di rilasciare versioni compatibili di kernel, librerie C, compilatori e utilità... Veramente un grosso lavoro!
- ▶ Le utilità sono disponibili in *pacchetti* che possono essere facilmente installati, rimossi o aggiornati. La dipendenza dalle varie versioni è gestita automaticamente.
- ▶ Distribuzioni Commerciali: includono l'assistenza. I sorgenti sono liberi, ma i binari di solito no.
- ▶ Distribuzioni Community: sia i sorgenti che i binari sono liberi. Nessuna assistenza di solito.
- ▶ Non confondete la versione della distribuzione con la versione del kernel di Linux!



Distribuzioni Commerciali

- Red Hat: <http://www.redhat.com/>

La più popolare. Affidabile, sicura, facile da usare e da installare, supportata da tutti i venditori di hardware e software.



- Suse (Novell): <http://www.suse.com/>

La principale alternativa. Facile da installare e da usare, stabile. supportata da tutti i venditori di hardware e software.



- Mandriva (ex Mandrake): <http://mandrivalinux.com/>

Facile da usare e da installare, più aggiornata ma meno stabile. Più orientata ad utenti individuali. Poco supporto da parte del venditore.



Distribuzioni Community

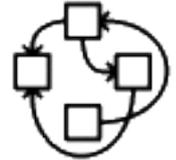
- Fedora Core: <http://fedora.redhat.com/>
Stabile, sicura, facile da usare e da installare. Frequenti aggiornamenti.
- Ubuntu Linux: <http://ubuntu-linux.org/>
Distribuzione in crescita. Basata su Debian ma rilascia una nuova versione ogni 6 mesi. Facile da usare. Ottima per chi inizia.
- Debian: <http://debian.org/>
Molto stabile e sicura, ma più difficile da configurare e installare. Buona per gli sviluppatori, ma non ancora per gli utenti. Le nuove versioni non sono abbastanza frequenti (ogni 2 o 3 anni). Ottima per i server, ma non per i principianti!
- Mandriva Community: <http://mandrivalinux.com/>
Facile da installare e da usare, sicura, rilascia frequenti, ma meno stabile (ci sono pochi test e manca la gestione del feedback dell'utente).



Altri sistemi Unix Liberi (1)

GNU / Hurd: <http://www.gnu.org/software/hurd/hurd.html>

- ▶ Utilità GNU con Hurd, il kernel GNU (microkernel)
- ▶ Sta maturando, ma non abbastanza per un uso generico.
Al momento (2005) usato per lo più da sviluppatori Hurd.



Famiglia BSD

- ▶ FreeBSD: <http://www.freebsd.org/>
Sistema BSD potente, multi piattaforma, sicuro e popolare.
- ▶ OpenBSD: <http://openbsd.org/>
Costruito mirando alla massima sicurezza e affidabilità.
Popolare nei server Internet.
- ▶ NetBSD: <http://netbsd.org/>
Distribuzione BSD progettata per essere portabile (disponibile su ARM e altri)



Altri sistemi Unix liberi (2)

▶ ECOS: <http://ecos.sourceware.org/>

Sistema in tempo reale, molto leggero, per dispositivi embedded sviluppato da Red Hat/Cygnus solutions.
È conforme alle API POSIX.



I file system di Unix



Tutto è un file

Quasi tutto in Unix è un file!

- ▶ **File regolari**
- ▶ **Directory**
Le directory sono semplicemente file che elencano un insieme di file
- ▶ **Link simbolici**
File che si riferiscono al nome di un altro file
- ▶ **Device e periferiche**
Leggi e scrivi da dispositivi come fossero file regolari
- ▶ **Pipe “|”**
Usati per mettere in sequenza programmi
`cat *.log | grep error`
- ▶ **Socket**
Comunicazione tra processi



Nomi di file

Caratteristiche dei nomi dei file, fin dagli inizi di Unix

- ▶ Distingue maiuscolo/minuscolo
- ▶ Nessun limite (stretto) di lunghezza
- ▶ può contenere qualsiasi carattere (incluso lo spazio, escluso “/”). I tipi dei file sono memorizzati nei file stessi (“magic numbers”). Le estensioni dei file non sono necessarie e non sono interpretate. Sono usate per comodità.

- ▶ Esempi di nomi di file:

README
index.htm

.bashrc
index.html

Windows Buglist
index.html.old



File path

Un *path* è una sequenza di directory con un file o una directory alla fine, separati dal carattere “/”

- ▶ Path relativo: `documents/fun/microsoft_jokes.html`
Relativo alla directory corrente
- ▶ Path assoluto: `/home/bill/bugs/crash9402031614568`
- ▶ “/” : *directory root*.
Inizio dei path assoluto per tutti i file presenti sul sistema (anche per i file su dispositivi rimovibili e condivisi in rete).



Struttura del filesystem di GNU/Linux (1)

Non è imposta dal sistema. Può variare da un sistema ad un altro, persino tra due installazioni GNU/Linux!

/	Directory root
/bin/	Comandi di sistema di base
/boot/	Immagini del kernel, initrd e configurazioni
/dev/	File che rappresentano dispositivi, ad esempio: <code>/dev/hda</code> : primo Hard Disk IDE
/etc/	File di configurazione del sistema
/home/	Directory degli utenti
/lib/	Librerie condivise di base del sistema



Struttura del filesystem di GNU/Linux (2)

<code>/lost+found</code>	File danneggiati che ha cercato di recuperare
<code>/mnt/</code>	Filesystem montati <code>/mnt/usbdisk/</code> , <code>/mnt/windows/</code> ...
<code>/opt/</code>	Programmi specifici installati da sysadmin Spesso si usa <code>/usr/local/</code> invece
<code>/proc/</code>	Accesso ad informazioni di sistema <code>/proc/cpuinfo</code> , <code>/proc/version</code> ...
<code>/root/</code>	home directory dell'utente root
<code>/sbin/</code>	Comandi riservati all'amministratore
<code>/sys/</code>	Controlli del sistema e dei dispositivi (frequenza cpu, alimentazione dispositivi, etc.)



Struttura del filesystem di GNU/Linux (3)

<code>/tmp/</code>	File temporanei
<code>/usr/</code>	Programmi dell'utente (non essenziali al sistema) <code>/usr/bin/</code> , <code>/usr/lib/</code> , <code>/usr/sbin...</code>
<code>/usr/local/</code>	Software specifico installato dall'amministratore (spesso preferito a <code>/opt/</code>)
<code>/var/</code>	Dati usati dal sistema o dai server di sistema <code>/var/log/</code> , <code>/var/spool/mail</code> (mail in arrivo), <code>/var/spool/lpd</code> (code di stampa)...



Shell e gestione file



Interpreti della linea di comando

- ▶ Shell: programma per eseguire comandi utente
- ▶ Chiamato “shell” perché nasconde i dettagli del sottostante sistema operativo come un guscio.
- ▶ I comandi sono inseriti testualmente in un terminale, in una finestra in un ambiente grafico, o in una console solo testo.
- ▶ Anche i risultati sono mostrati sul terminale. Non serve nessuna grafica.
- ▶ Si possono usare degli script: fornisce tutte le risorse per scrivere programmi complessi (variabili, if, iterazioni...)



Le shell più note

Le shell più famose e popolari

- ▶ **sh**: La Bourne shell (obsoleta)
Tradizionale, shell base presente nei sistemi Unix, di Steve Bourne.
- ▶ **csh**: la shell C (obsoleta)
Una shell popolare tempo fa, con sintassi simile al C
- ▶ **tcsh**: la TC shell (ancora molto popolare)
Una implementazione compatibile con la C shell con caratteristiche evolute (completamento dei comandi, editing della storia e altro...)
- ▶ **bash**: la Bourne Again shell (la più popolare)
Una versione migliorata di sh con molte caratteristiche in più.



Comando ls

Elenca i file nella directory corrente, in ordine alfanumerico, eccetto i file che iniziano con il carattere “.”.

- ▶ `ls -a` (all-tutti)
Elenco di tutti i file (inclusi i file `.*`)
- ▶ `ls -l` (lungo)
Elenco lungo (tipo, data, dimensione, permessi, proprietà)
- ▶ `ls -t` (tempo)
Elenco dei file più recenti
- ▶ `ls -S` (size-dimensione)
Elenco dai file più grandi
- ▶ `ls -r` (rovescio)
Inverti l'ordinamento
- ▶ `ls -ltr` (le opzioni possono essere combinate): elenco lungo, con i file più recenti alla fine



Sostituzione dei caratteri speciali

Meglio procedere per esempi!

▶ `ls *txt`

La shell sostituisce `*txt` con tutti i file e i nomi di directory che finiscono per `txt` (incluso `.txt`), esclusi quelli che iniziano per `.`, e poi esegue il comando `ls`.

▶ `ls -d .*`

Elenca tutti i file e directory che iniziano con `.`

`-d` dice a `ls` di non mostrare il contenuto delle directory `.*`

▶ `cat ?.log`

Mostra tutti i file che iniziano con un solo carattere e finiscono per `.log`



Directory speciali (1)

`./`

- ▶ La directory corrente. Utile per comandi che vogliono come argomento una directory. Utile anche per eseguire comandi nella directory corrente (vedi più avanti)
- ▶ dunque `./readme.txt` e `readme.txt` sono equivalenti

`../`

- ▶ La directory padre (superiore), a cui appartiene la directory `.` (vedi `ls -a`). Unico riferimento alla directory padre.
- ▶ Uso tipico:
`cd ..`



Directory speciali (2)

~/

- ▶ Non è una vera directory speciale. La shell la sostituisce con il nome della directory dell'utente attuale.
- ▶ Non può essere usata in molti programmi, non essendo una directory reale.

~sydney/

- ▶ Analogo, sostituito dalla shell con il nome della directory dell'utente `sydney`.



Il comando cd e pwd

- ▶ `cd <dir>`
cambia la directory corrente a `<dir>`
- ▶ `pwd`
Mostra la directory corrente ("directory di lavoro")



Il comando cp

- ▶ `cp <file_sorgente> <file_destinazione>`
Copia il file sorgente nella destinazione
- ▶ `cp file1 file2 file3 ... dir`
Copia i file nella directory destinazione (ultimo argomento)
- ▶ `cp -i` (interattivo)
Chiede conferma se il file destinazione esiste già
- ▶ `cp -r <dir_sorgente> <dir_destinazione>`
(ricorsivo)
Copia l'intera directory



Copia intelligente di directory con rsync

rsync (sync remoto) è pensato per mantenere in sincronizzazione directory su 2 macchine con un collegamento lento.

- ▶ Copia solo i file che sono stati cambiati. I files della stessa dimensione sono confrontati con un checksum.
- ▶ Trasferisce solo i blocchi di un file che sono diversi!
- ▶ Può comprimere i blocchi trasferiti
- ▶ Mantiene i link simbolici e i permessi dei file: utile anche per copie nella stessa macchina.
- ▶ Può lavorare con ssh (shell remota sicura). Molto utile per aggiornare i contenuti di un sito web, ad esempio.



Esempi di rsync (1)

▶ `rsync -a /home/arvin/sd6_agents/ /home/sydney/misc/`

-a: modo archivio. Equivalente a `-r1ptgoD...` un modo facile per dire che vuoi la ricorsione, preservando quasi tutto.

▶ `rsync -Pav --delete /home/steve/ideas/ /home/bill/my_ideas/`

-P: `--partial` (tieni i file parzialmente trasferiti) and `--progress` (mostra i progressi durante il trasferimento)

`--delete`: cancella i file che non esistono nei sorgenti.

Attenzione: i nome delle directory devono finire con `/` altrimenti si ottiene la directory `my_ideas/ideas/` nella destinazione.



Esempi di rsync (2)

- ▶ Copiare su una macchina remota

```
rsync -Pav /home/bill/legal/arguments/ \  
bill@www.sco.com:/home/legal/arguments/
```

L'utente `bill` deve inserire la password

- ▶ Copiare da una macchina remota con ssh

```
rsync -Pav -e ssh  
homer@tank.duff.com/prod/beer/ \  
fridge/homer/beer/
```

L'utente `homer` deve inserire la sua password



Comandi mv e rm

- ▶ `mv <vecchio_nome> <nuovo_nome>` (muovi)
Rinomina i file o le directory
- ▶ `mv -i` (interattivo)
Se il nuovo file esiste, chiedi conferma all'utente
- ▶ `rm file1 file2 file3 ...` (cancella)
Cancella i file dati
- ▶ `rm -i` (interattivo)
Chiedi conferma all'utente
- ▶ `rm -r dir1 dir2 dir3` (ricorsivo)
Rimuove le directory date e il loro contenuto



Creare e cancellare le directory

- ▶ `mkdir dir1 dir2 dir3 ...` (crea le directory)
Crea le directory con i nomi dati
- ▶ `rmdir dir1 dir2 dir3 ...` (cancella directory)
Cancella le directory date
È più sicuro: funziona solo con directory vuote
Alternativa: `rm -r`



Mostrare il contenuto dei file

Ci sono parecchi modi per mostrare il contenuto dei file

▶ `cat file1 file2 file3 ...` (concatena)

Concatena e mostra il contenuto dei file dati

▶ `more file1 file2 file3 ...`

Dopo ciascuna pagina chiede all'utente di premere un tasto.

Si può anche andare direttamente alla parola voluta (comando /)

▶ `less file1 file2 file3 ...`

Fai più di more, e con meno fatica

Non legge l'intero file prima di iniziare

Gestisce il movimento all'indietro nel file (comando ?)



I comandi head e tail

▶ `head [-<n>] <file>`

Mostra le prime <n> linee (o 10 di default) del file dato.

Non deve leggere l'intero file per fare questo!

▶ `tail [-<n>] <file>`

Mostra le ultime <n> linee (o 10 di default) del file dato.

Non deve leggere l'intero file in RAM! Utile per file grandi.

▶ `tail -f <file>` (inseguì)

Mostra le ultime 10 linee del file e continua a mostrare le nuove linee quando vengono aggiunte.

Molto utile per controllare i cambiamenti di un file di log.

▶ Esempi:

```
head windows_bugs.txt
```

```
tail -f outlook_vulnerabilities.txt
```



Il comando grep

▶ `grep <pattern> <files>`

Cerca nei file dati e mostra le linee che corrispondono allo schema dato.

▶ `grep error *.log`

Mostra tutte le linee che contengono la parola `error` nei file `*.log`

▶ `grep -i error *.log`

Lo stesso, senza distinguere tra maiuscolo e minuscolo

▶ `grep -ri error .`

Lo stesso, ma ricorsivamente in tutti i file in `.` e nelle sue sottodirectory

▶ `grep -v info *.log`

Mostra tutte le linee nei file `*.log` eccetto quelle contenenti `info`



Il comando sort

▶ `sort <file>`

Ordina alfanumericamente le linee del file dato e le mostra

▶ `sort -r <file>`

Lo stesso, ma in ordine inverso

▶ `sort -ru <file>`

u: unico. Lo stesso, ma mostra una sola volta linee identiche.

▶ Molte altre possibilità seguiranno a breve!



Link simbolici

Un link simbolico è un file speciale che contiene solo un riferimento al nome di un altro file o directory:

- ▶ Utile per ridurre l'occupazione del disco e il disordine quando 2 file hanno lo stesso contenuto
- ▶ Esempio:
`anakin_skywalker_biography -> darth_vador_biography`
- ▶ Come identificare un link simbolico:
 - ▶ `ls -l` mostra -> e il nome del file collegato
 - ▶ GNU `ls` mostra i link in un colore diverso



Creare link simbolici

- ▶ Per creare un link simbolico (stesso ordine usato in `cp`):

```
ln -s file_name link_name
```

- ▶ Per creare un link con un file in un'altra directory, con lo stesso nome:

```
ln -s ../README.txt
```

- ▶ Per creare link multipli con un comando solo in una directory data:

```
ln -s file1 file2 file3 ... dir
```

- ▶ Per cancellare un link (questo non cancella il file originale!):

```
rm link_name
```



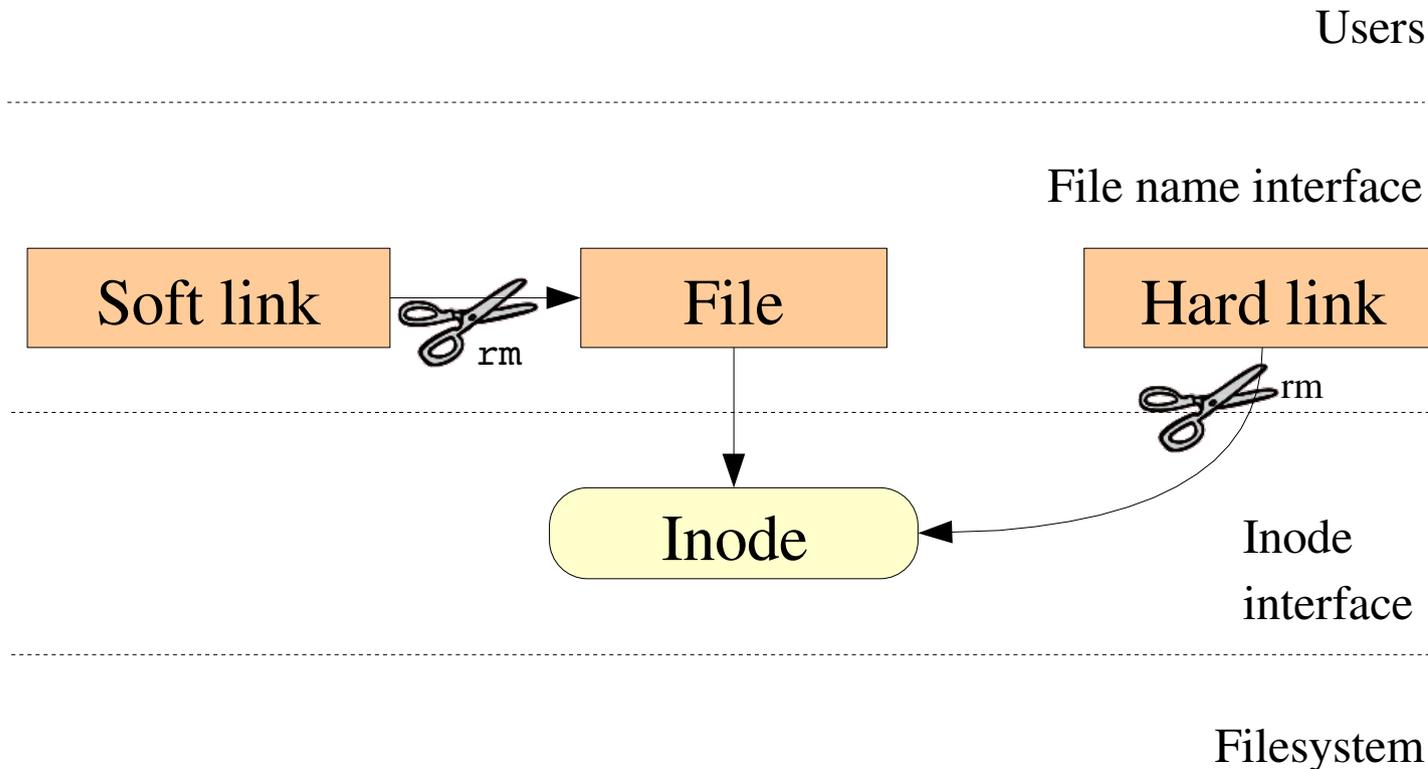
Hard link

- ▶ Il comportamento di default di `ln` è di creare *hard link*
- ▶ Un *hard link* ad un file è un file regolare con esattamente lo stesso contenuto fisico
- ▶ Si risparmia spazio, ma gli hard links non si possono distinguere dai file originali.
- ▶ Se si cancella il file originale, il contenuto dell'hard link non subisce cambiamenti.
- ▶ Il contenuto è rimosso quando non vi sono più file (hard links) che fanno riferimento ad esso.



Nomi di file e inodes

Come capire i link simbolici (soft) e hard !



Diritti di accesso ai file

Usa `ls -l` per verificare i diritti di accesso ai file

3 tipi di diritti di accesso

- ▶ Accesso in lettura (**r**)
- ▶ Accesso in scrittura (**w**)
- ▶ Diritto di esecuzione (**x**)

3 tipi di livello di accesso

- ▶ Utente (**u**): per il proprietario del file
- ▶ Gruppo (**g**): ciascun file ha anche un attributo di gruppo, corrispondente ad una lista di utenti
- ▶ Altri (**o**): per tutti gli altri utenti



Limiti nei diritti di accesso

- ▶ `x` senza `r` è legale, ma non serve a niente
Bisogna essere in grado di leggere un file per eseguirlo
- ▶ Per le directory serve sia `r` che `x`: `x` per entrare, `r` per vedere il contenuto.
- ▶ Non si può rinominare, cancellare, copiare file in una directory se non si ha il permesso di accesso in scrittura (`w`) alla directory.
- ▶ Se hai accesso `w` a una directory, PUOI cancellare un file anche se non hai il permesso di scrittura su quel file (ricorda che una directory è solo un file che contiene una lista di file). Questo consente di modificare (cancellare + ricreare) un file anche senza accesso ad esso.



Esempi di diritto d'accesso

▶ `-rw-r--r--`

Leggibile e scrivibile per il proprietario del file, solo leggibile per gli altri

▶ `-rw-r-----`

Leggibile e scrivibile per il proprietario del file, solo leggibile per gli utenti appartenenti allo stesso gruppo.

▶ `drwx-----`

Directory accessibile solo dal proprietario

▶ `-----r-x`

File eseguibile dagli altri ma non da te o dai tuoi amici. Bella protezione per una trappola...



chmod: cambiare i permessi

▶ `chmod <permessi> <file>`

2 formati per i permessi:

▶ Formato ottale(abc):

$a, b, c = r*4 + w*2 + x$ (r, w, x: booleani)

Esempio: `chmod 644 <file>`

(rw for u, r per g e o)

▶ O formato simbolico. Facile da capire con esempi:

`chmod go+r`: aggiungi permesso di lettura a gruppo e altri

`chmod u-w`: toglì permesso di scrittura all'utente

`chmod a-x`: (a: all) rimuovi permesso di esecuzione per tutti



Ancora chmod (1)

```
chmod -R a+rX linux/
```

Rende `linux` e tutto ciò che vi si trova disponibile a chiunque!

- ▶ R: esegue i cambiamenti ricorsivamente
- ▶ X: x, applica x solo alle directory e ai file già eseguibili e Molto utile per consentire in maniera ricorsiva l'accesso alle directory, senza aggiungere il diritto di esecuzione a tutti i file.



Ancora chmod (2)

```
chmod a+t /tmp
```

- ▶ **t**: (sticky). Permesso speciale per directory, per consentire solo al proprietario della directory e del file di cancellare un file nella directory.
- ▶ Utile per directory con diritto di scrittura per tutti come /tmp.
- ▶ Mostrato da `ls -l` con il carattere **t**



Introduzione a Unix e GNU/Linux

Standard I/O, ridirezione e pipe



Standard output

Osservazioni sull'output dei comandi

- ▶ Tutti i comandi che mostrano output di testo sul terminale, lo fanno scrivendo sul loro *standard output*.
- ▶ Lo standard output può essere scritto su un file (ridirezione) usando il simbolo >
- ▶ Lo standard output può essere aggiunto in coda a un file esistente usando il simbolo >>



Esempi ridirezione dello standard output

- ▶ `ls ~saddam/* > ~gwb/weapons_mass_destruction.txt`
- ▶ `cat obiwan_kenobi.txt > starwars_biographies.txt`
`cat han_solo.txt >> starwars_biographies.txt`
- ▶ `echo "README: No such file or directory" > README`
Un modo per creare un file senza editor di testi.
In questo caso è anche uno scherzo di Unix.



Standard input

Alcune osservazioni sull'input dei comandi

▶ Molti comandi, quando non hanno argomenti di input, leggono il loro input dallo *standard input*.

▶ `sort`

`windows`

`linux`

`[Ctrl][D]`

`linux`

`windows`

`sort` prende il suo input dallo standard input: in questo caso, quello che inserisci dal terminale (terminato da `[Ctrl][D]`)

▶ `sort < participants.txt`

Lo standard input di `sort` è preso dal file dato.



Pipe

- ▶ I pipe di Unix sono molto utili per ridirezionare lo standard output di un comando allo standard input di un altro.
- ▶ Esempi
 - ▶ `cat *.log | grep -i error | sort`
 - ▶ `grep -ri error . | grep -v "ignored" | sort -u \> serious_errors.log`
 - ▶ `cat /home/*/homework.txt | grep mark | more`
- ▶ Questa è una delle caratteristiche più potenti delle shell di Unix!



Il comando tee

```
tee [-a] file
```

- ▶ Il `tee` comando può essere usato per inviare lo standard output contemporaneamente ad un file e sul video
- ▶ `make | tee build.log`
Esegue il comando `make` e memorizza l'output in `build.log`
- ▶ `make install | tee -a build.log`
Esegue il comando `make install` e aggiunge il suo output a `build.log`



Standard error

▶ I messaggi di errore sono di solito stampati (se il programma è scritto bene) su *standard error* invece che su standard output.

▶ Lo standard error può essere redirezionato con `2>` o `2>>`

▶ Esempio:

```
cat f1 f2 nofile > nuovofile 2> errfile
```

▶ 1 è il descrittore di standard output, cioè `1>` è equivalente a `>`

▶ Si può ridirezionare sia lo standard output che lo standard error allo stesso file usando `&>`

```
cat f1 f2 nofile &> wholefile
```



Il comando yes

Utile per riempire lo standard input con una stringa ripetuta.

▶ `yes <stringa> | <comando>`

Continua a riempire lo standard input di `<comando>` con `<stringa>` (y di default)

▶ Esempi

```
yes | rm -r dir/
```

```
bank> yes no | credit_applicant
```

```
yes "" | make oldconfig
```

(equivalente a premere `Enter` per accettare tutte le scelte di default)



Device speciali

Sembrano file reali, ma

▶ /dev/null

Il buco nero dei dati! Butta via tutti i dati scritti su questo file.

Utile per liberarsi di output non voluto, tipicamente informazioni di log:

```
mplayer black_adder_4th.avi &> /dev/null
```

▶ /dev/zero

Leggendo da questo file, si ottiene sempre il carattere \0

Utile per creare file riempiti con zeri:

```
dd if=/dev/zero of=disk.img bs=1k count=2048
```



Controllo dei processi



Controllo dei processi

- ▶ Unix, fin dalle origini, gestisce un vero multitasking preemptivo.
- ▶ Capacità di eseguire molti processi in parallelo, e terminarli anche se corrompono il loro stato e i loro dati.
- ▶ Possibilità di scegliere quali programmi eseguire.
- ▶ Capacità di scegliere quale input dare ai tuoi programmi e dove mandare l'output.



Processi

“Tutto in Unix è un file,
Tutto ciò che non è un file, è un processo”

Processi

- ▶ Istanze di un programma in esecuzione
- ▶ Diverse istanze dello stesso programma possono essere in esecuzione allo stesso tempo
- ▶ Dati associati ai processi:
File aperti, memoria allocata, stack, numero di processo, padre, priorità, stato...



Eseguire processi in background

L'uso è lo stesso per tutte le shells

▶ Utile

- ▶ Per compiti da linea di comando il cui output può essere esaminato più tardi, in particolare per programmi che richiedono un lungo tempo di esecuzione.
- ▶ Per iniziare applicazioni grafiche dalla linea di comando e poi continuare a controllarle con il mouse.
- ▶ Per far partire un processo in background: aggiungi & alla fine del comando:

```
find_prince_charming --cute --clever --rich &
```



Controllo dei processi in background

▶ jobs

Restituisce la lista dei processi in background nella shell usata

```
[1]-  Running ~/bin/find_meaning_of_life --without-god &  
[2]+  Running make mistakes &
```

▶ fg

fg %<n>

Porta in foreground l'ultimo/ennesimo processo in background

▶ Mette il processo corrente in modo background:

[Ctrl] Z

bg

▶ kill %<n>

Uccide l'ennesimo processo.



Esempio di controllo processi

```
> jobs
[1]-  Running ~/bin/find_meaning_of_life --without-god &
[2]+  Running make mistakes &

> fg
make mistakes

> [Ctrl] Z
[2]+  Stopped make mistakes

> bg
[2]+  make mistakes &

> kill %1
[1]+  Terminated ~/bin/find_meaning_of_life --without-god
```



Elenco di tutti i processi

... in qualsiasi modo siano stati fatti partire

▶ `ps -ux`

Elenco di tutti i processi che appartengono all'utente corrente

▶ `ps -aux` (Note: `ps -edf` on System V systems)

Elenco di tutti i processi in esecuzione nel sistema

▶ `ps -aux | grep bart | grep bash`

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
bart	3039	0.0	0.2	5916	1380	pts/2	S	14:35	0:00	/bin/bash
bart	3134	0.0	0.2	5388	1380	pts/3	S	14:36	0:00	/bin/bash
bart	3190	0.0	0.2	6368	1360	pts/4	S	14:37	0:00	/bin/bash
bart	3416	0.0	0.0	0	0	pts/2	RW	15:07	0:00	[bash]

▶ PID: Identificatore del processo
VSZ: Dimensione Virtuale del processo (codice + dati + stack)
RSS: Dimensione residente del Processo: numero di KB attualmente in RAM
TTY: Terminale
STAT: Stato: R (In esecuzione), S (Addormentato), W (In attesa), Z (Zombie)...



Terminare i processi (1)

▶ `kill <pids>`

Manda un segnale di abort al processo dato. Consente al processo di salvare i dati e di uscire di sua volontà. Dovrebbe essere usato per primo. Esempio:

```
kill 3039 3134 3190 3416
```

▶ `kill -9 <pids>`

Manda un segnale di terminazione immediata. Il sistema stesso uccide il processo. Utile quando un processo è veramente bloccato (non risponde a `kill -1`).

▶ `kill -9 -1`

Termina tutti i processi (opzione `-1`) dell'utente corrente.



Terminare i processi (2)

▶ `killall [-<signal>] <command>`

Termina tutti i processi lanciati con `<command>`.

Esempio: `killall bash`

▶ `xkill`

Ti lascia terminare una applicazione grafica selezionandola con il mouse!

Molto veloce! Utile quando non conosci il comando che ha lanciato l'applicazione.



Attività dei processi

- ▶ **top** – Mostra i processi più importanti, in ordine di occupazione di tempo macchina.

```
top - 15:44:33 up 1:11, 5 users, load average: 0.98, 0.61, 0.59
Tasks: 81 total, 5 running, 76 sleeping, 0 stopped, 0 zombie
Cpu(s): 92.7% us, 5.3% sy, 0.0% ni, 0.0% id, 1.7% wa, 0.3% hi, 0.0% si
Mem: 515344k total, 512384k used, 2960k free, 20464k buffers
Swap: 1044184k total, 0k used, 1044184k free, 277660k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3809	jdoe	25	0	6256	3932	1312	R	93.8	0.8	0:21.49	bunzip2
2769	root	16	0	157m	80m	90m	R	2.7	16.0	5:21.01	X
3006	jdoe	15	0	30928	15m	27m	S	0.3	3.0	0:22.40	kdeinit
3008	jdoe	16	0	5624	892	4468	S	0.3	0.2	0:06.59	autorun
3034	jdoe	15	0	26764	12m	24m	S	0.3	2.5	0:12.68	kscd
3810	jdoe	16	0	2892	916	1620	R	0.3	0.2	0:00.06	top

- ▶ Puoi cambiare l'ordine premendo **M**: Memoria, **P**: %CPU, **T**: Tempo.
- ▶ Puoi terminare un task premendo **k** e il numero del processo.



Recuperare una sessione grafica bloccata

- ▶ Se la tua sessione grafica è bloccata e non puoi più usare i tuoi terminali, non fare un reboot!
- ▶ Probabilmente il tuo sistema è ancora funzionante. Prova ad accedere ad una console premendo la sequenza [Ctrl][Alt][F1] (o [F2],[F3] per altre console di testo)
- ▶ Nella console di testo, puoi provare ad uccidere l'applicazione bloccata.
- ▶ Una volta fatto, puoi tornare alla sessione grafica premendo [Ctrl][Alt][F5] o [Ctrl][Alt][F7] (dipende dalla distribuzione)
- ▶ Se non puoi identificare il programma bloccato, puoi uccidere tutti i tuoi processi: `kill -9 -1`
Poi compare la schermata di login.



Sequenze di comandi

- ▶ Puoi scrivere il prossimo comando sul terminale anche se il comando corrente non è terminato.
- ▶ Puoi separare i comandi con il simbolo `;` :
`echo "I love thee"; sleep 10; echo " not"`
- ▶ Condizionali: usa `||` (o) oppure `&&` (e):
`more God || echo "Sorry, God doesn't exist"`
Esegue `echo` solo se il primo comando fallisce

```
ls ~sd6 && cat ~sd6/* > ~sydney/recipes.txt
```

Esegue il `cat` del contenuto della directory se il comando `ls` ha successo (cioè ha diritti di accesso in lettura).



Uso delle virgolette (1)

Doppi apici, o virgolette, (") possono essere usati per evitare che la shell interpreti gli spazi come separatori di argomenti, e per evitare l'espansione dei nomi dei file con caratteri speciali.

```
> echo "Hello World"  
Hello World
```

```
> echo "You are logged as $USER"  
You are logged as bgates
```

```
> echo *.log  
find_prince_charming.log cosmetic_buys.log
```

```
> echo "*.log"  
*.log
```



Uso delle virgolette (2)

Un singolo apice è simile, ma non viene eseguita nessuna sostituzione di ciò che si trova tra apici

```
> echo 'You are logged as $USER'  
You are logged as $USER
```

L'apice rovescio (`) può essere usato per chiamare un comando da dentro un altro

```
> cd /lib/modules/`uname -r`; pwd  
/lib/modules/2.6.9-1.6_FC2
```

L'apice rovescio può essere usato dentro i doppi apici

```
> echo "You are using Linux `uname -r`"  
You are using Linux 2.6.9-1.6_FC2
```



Misurare il tempo trascorso

```
▶ time find_expensive_housing --near  
<...command output...>  
real      0m2.304s (tempo trascorso effettivamente)  
user      0m0.449s (tempo in cui la CPU ha eseguito il  
             codice del programma)  
sys       0m0.106s (tempo in cui la CPU ha eseguito  
             chiamate di sistema)
```

reale = utente + sistema + *attesa*

attesa = tempo di attesa di I/O + tempo idle (altri programmi in esecuzione)



Variabili d'ambiente

- ▶ Le shell consentono di definire delle *variabili*.
Possono essere usate in comandi shell.
Convenzione: caratteri minuscoli
- ▶ Si possono anche definire *variabili d'ambiente*: variabili che sono visibili anche all'interno di script o programmi eseguibili chiamati dalla shell.
Convenzione: caratteri maiuscoli
- ▶ `env`
Elenco di tutte le variabili d'ambiente e il loro valore



Esempi di variabili shell

variabili shell (bash)

- ▶ `projdir=/home/marshall/coolstuff`
`ls -la $projdir; cd $projdir`

Variabili ambiente (bash)

- ▶ `cd $HOME`

- ▶ `export DEBUG=1`
`./trova_vita_extraterreste`
(mostra informazioni di debug se DEBUG è impostato)



Principali variabili d'ambiente standard

Usate da moltissime programmi!

- ▶ **LD_LIBRARY_PATH**
Dove sono le librerie condivise
- ▶ **DISPLAY**
Identificativo del monitor su cui mostrare applicazioni X.
- ▶ **EDITOR**
Editor di default (vi, emacs...)
- ▶ **HOME**
Directory Home dell'utente
- ▶ **HOSTNAME**
Nome della macchina locale

- ▶ **MANPATH**
Dove sono le pagine manuale
- ▶ **PATH**
Dove sono i comandi
- ▶ **PRINTER**
Stampante di default
- ▶ **SHELL**
Nome della shell corrente
- ▶ **TERM**
Nome/modo del terminale
- ▶ **USER**
Nome dell'utente attuale



Variabile d'ambiente PATH

▶ PATH

Indica alla shell l'ordine in cui cercare i programmi

```
/home/abox/bin:/usr/local/bin:/usr/kerberos/bin  
:/usr/bin:/bin:/usr/X11R6/bin:/bin:/usr/bin
```

▶ LD_LIBRARY_PATH

Indica l'ordine in cui cercare le librerie condivise (librerie di codice condiviso dalle applicazioni, come la libreria C) per ld

```
/usr/local/lib:/usr/lib:/lib:/usr/X11R6/lib
```

▶ MANPATH

Indica l'ordine in cui cercare le pagine del manuale

```
/usr/local/man:/usr/share/man
```



Accorgimenti nell'uso di PATH

Si raccomanda di non mettere la directory “.” nella variabile d'ambiente PATH, in particolare mai all'inizio:

- ▶ Un cracker potrebbe mettere un file `ls` modificato in una directory. Verrebbe eseguito quando si lancia `ls` in questa directory e potrebbe causare danni ai dati.
- ▶ Se hai un file eseguibile chiamato `test` in una directory, questo viene eseguito al posto del programma di sistema `test` e alcuni script potrebbero non funzionare.
- ▶ Ogni volta che esegui un `cd` in una nuova directory, la shell perde tempo ad aggiornare la lista dei comandi disponibili.

Invoca i comandi locali con la seguente sintassi: `./test`



Alias

La shell consente di definire comandi *alias*: abbreviazioni di comandi usati molto spesso

Esempi

- ▶ `alias ls='ls -la'`
Un modo utile per eseguire comandi con argomenti di default
- ▶ `alias rm='rm -i'`
Utile per far chiedere sempre conferma a `rm`
- ▶ `alias frd='find_rambaldi_device --asap --risky'`
Utile per sostituire una linea di comando lunga e frequente.
- ▶ `alias cia='. /home/sydney/env/cia.sh'`
Utile per impostare una variabile d'ambiente in maniera veloce
(`.` è un comando di shell per eseguire il contenuto di uno shell script)



Il comando which

which ti dice dove si trova un comando eseguibile

- ▶ `bash> which ls`
`alias ls='ls --color=tty'`
`/bin/ls`
- ▶ `tcsh> which ls`
`ls: aliased to ls --color=tty`
- ▶ `bash> which alias`
`/usr/bin/which: no alias in`
`(/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin)`
- ▶ `tcsh> which alias`
`alias: shell built-in command.`



Il file ~/.bashrc

- ▶ ~/.bashrc

Uno shell script letto ogni volta che la shell bash è eseguita

- ▶ Puoi usare questo file per definire

- ▶ Le tue variabili d'ambiente (PATH, EDITOR...)

- ▶ I tuoi alias

- ▶ Il tuo prompt (vedi il manuale bash per dettagli)

- ▶ Un messaggio di saluto



Utilità varie



Edit dei comandi

- ▶ Puoi usare i tasti freccia destra e sinistra per muovere il cursore nella linea di comando.
- ▶ Puoi usare [Ctrl][a] per andare all'inizio della linea, e [Ctrl][e] per andare alla fine.
- ▶ Puoi usare i tasti freccia su e giù per selezionare comandi precedenti.



Storia dei comandi (1)

- ▶ `history`

Mostra gli ultimi comandi che hai eseguito e il loro numero.
Puoi fare copia e incolla sulle righe dei comandi.

- ▶ Puoi richiamare l'ultimo comando:

`!!`

- ▶ Puoi richiamare un comando con il suo numero

`!1003`

- ▶ Puoi richiamare l'ultimo comando corrispondente ad una stringa data:

`!cat`



Storia dei comandi (2)

- ▶ Puoi fare sostituzioni sull'ultimo comando:
`^more^less`
- ▶ Puoi eseguire un altro comando con gli stessi argomenti:
`more !*`



Editor di testo

Editor di testo grafici

Buoni per la maggior parte degli utilizzi

- ▶ nedit
- ▶ Emacs, Xemacs

Editor di testo solo-testo

Spesso usati dai gestori e ottimi per utenti esperti

- ▶ vi
- ▶ nano



nedit(1)

```
Makefile - /data/mike/handhelds/stock_kernel/linux-2.6.8.1/arch/arm/
File Edit Search Preferences Shell Macro Windows Help
#
# arch/arm/Makefile
#
# This file is subject to the terms and conditions of the GNU General Public
# License. See the file "COPYING" in the main directory of this archive
# for more details.
#
# Copyright (C) 1995-2001 by Russell King

LDFLAGS_vmlinux :=-p --no-undefined -X
LDFLAGS_BLOB :=--format binary
AFLAGS_vmlinux.lds.o = -DTEXTADDR=$(TEXTADDR) -DDATAADDR=$(DATAADDR)
OBJCOPYFLAGS :=-O binary -R .note -R .comment -S
GZFLAGS :=-9
#CFLAGS +=-pipe

ifeq ($(CONFIG_FRAME_POINTER),y)
CFLAGS +=-fno-omit-frame-pointer -mapcs -mno-sched-prolog
endif

ifeq ($(CONFIG_CPU_BIG_ENDIAN),y)
CFLAGS += -mbig-endian
AS += -EB
LD += -EB
AFLAGS += -mbig-endian
else
CFLAGS += -mlittle-endian
AS += -EL
LD += -EL
AFLAGS += -mlittle-endian
endif

comma = ,

# This selects which instruction set is used.
# Note that GCC does not numerically define an architecture version
# macro, but instead defines a whole series of macros which makes
# testing for a specific architecture or later rather impossible.
```

<http://www.nedit.org/>

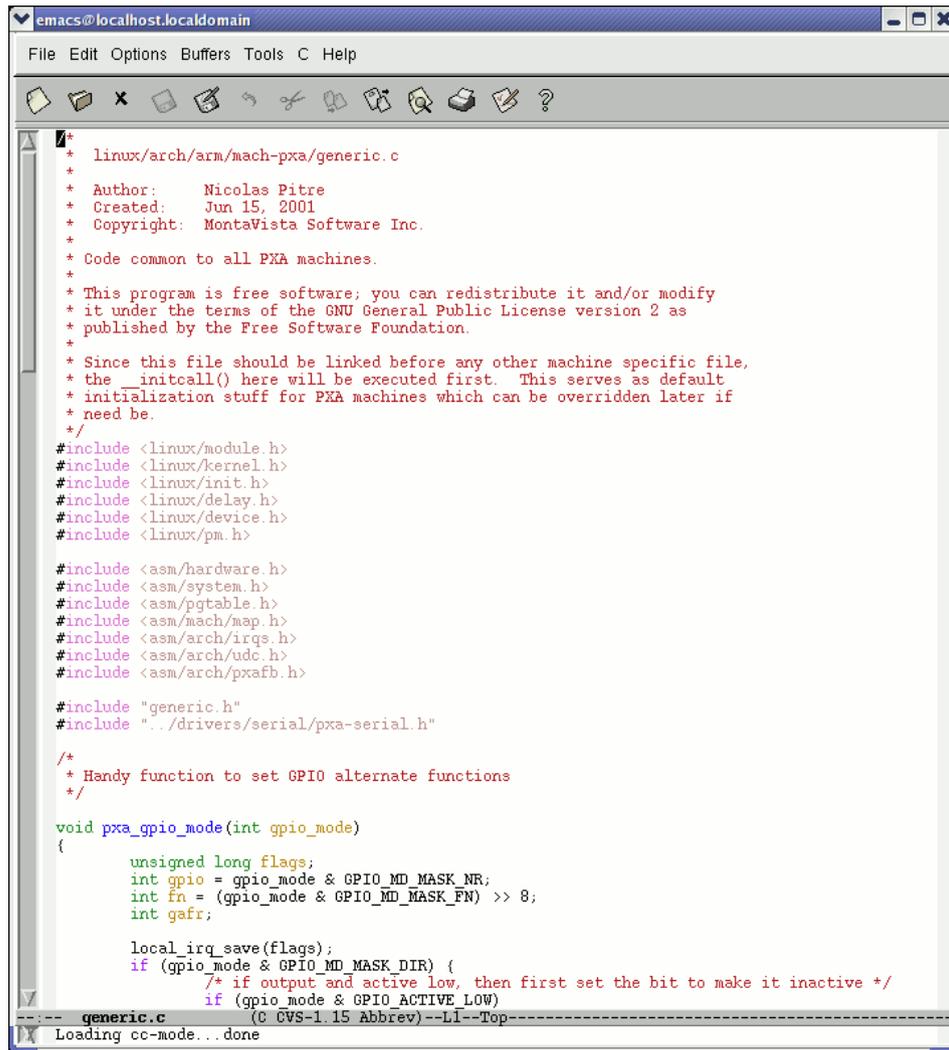


nedit (2)

- Il migliore editor di testo per chi non usa `vi` o `emacs`
- ▶ Caratteristiche principali:
 - Selezione e spostamento del testo molto facile
 - Evidenziazione del testo per molti linguaggi e formati. Può essere personalizzato per i propri file di log, per evidenziare particolari errori e avvertimenti.
 - Facile da personalizzare usando i menu
- ▶ Non è installato di default da tutte le distribuzioni



Emacs / Xemacs



```
emacs@localhost.localdomain
File Edit Options Buffers Tools C Help
[*]
* linux/arch/arm/mach-pxa/generic.c
*
* Author:   Nicolas Pitre
* Created:  Jun 15, 2001
* Copyright: MontaVista Software Inc.
*
* Code common to all PXA machines.
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation.
*
* Since this file should be linked before any other machine specific file,
* the __initcall() here will be executed first. This serves as default
* initialization stuff for PXA machines which can be overridden later if
* need be.
*/
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/delay.h>
#include <linux/device.h>
#include <linux/pm.h>

#include <asm/hardware.h>
#include <asm/system.h>
#include <asm/pgtable.h>
#include <asm/mach/map.h>
#include <asm/arch/irqs.h>
#include <asm/arch/udc.h>
#include <asm/arch/pxafb.h>

#include "generic.h"
#include "../drivers/serial/pxa-serial.h"

/*
 * Handy function to set GPIO alternate functions
 */

void pxa_gpio_mode(int gpio_mode)
{
    unsigned long flags;
    int gpio = gpio_mode & GPIO_MD_MASK_NR;
    int fn = (gpio_mode & GPIO_MD_MASK_FN) >> 8;
    int gafr;

    local_irq_save(flags);
    if (gpio_mode & GPIO_MD_MASK_DIR) {
        /* if output and active low, then first set the bit to make it inactive */
        if (gpio_mode & GPIO_ACTIVE_LOW)
            generic.c
            (C CVS-1.15 Abbrev)--LI--Top
Loading cc-mode... done
```

- Emacs e Xemacs sono simili (dipende dai propri gusti)
- Estremamente potente come editor
- Fantastico per utenti avanzati
- Meno ergonomico di nedit
- Abbreviazioni non standard
- Molto più di un editor di testi (giochi, posta elettronica, shell, navigatore)
- Alcuni potenti comandi richiedono un po' di studio



vi

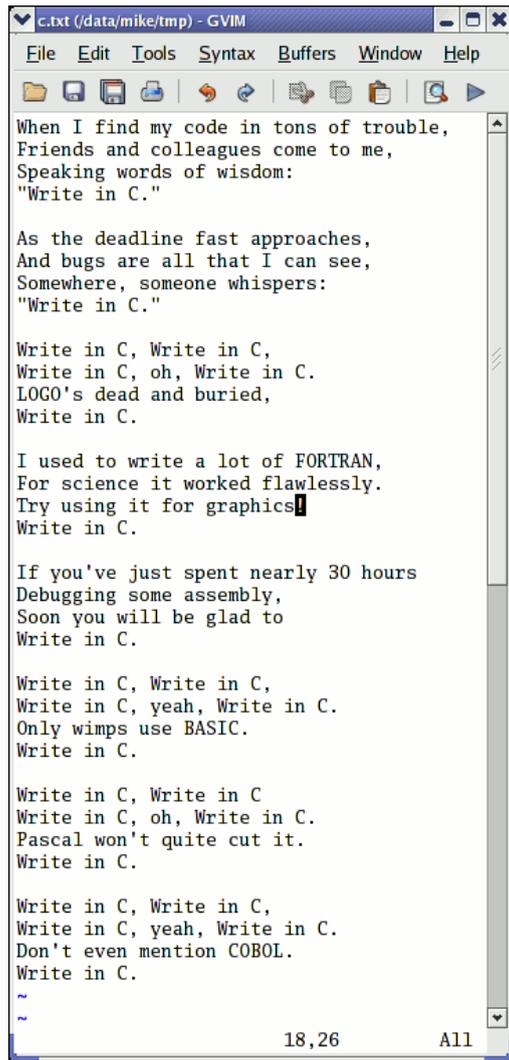
Editor in modalità testo disponibile in tutti i sistemi Unix.

Creato prima che vi fossero i computer con i mouse.

- Difficile da imparare ad usare se si è abituati ad un editor grafico.
- Molto produttivo per utenti esperti.
- Spesso non si può evitare di usarlo per editare file di configurazione nell'amministrazione di un sistema o in ambienti embedded, quando è disponibile solo la console in modalità testo.



vim - vi improved (migliorato)



```
c.txt (/data/mike/tmp) - GVIM
File Edit Tools Syntax Buffers Window Help
When I find my code in tons of trouble,
Friends and colleagues come to me,
Speaking words of wisdom:
"Write in C."

As the deadline fast approaches,
And bugs are all that I can see,
Somewhere, someone whispers:
"Write in C."

Write in C, Write in C,
Write in C, oh, Write in C.
LOGO's dead and buried,
Write in C.

I used to write a lot of FORTRAN,
For science it worked flawlessly.
Try using it for graphics
Write in C.

If you've just spent nearly 30 hours
Debugging some assembly,
Soon you will be glad to
Write in C.

Write in C, Write in C,
Write in C, yeah, Write in C.
Only wimps use BASIC.
Write in C.

Write in C, Write in C
Write in C, oh, Write in C.
Pascal won't quite cut it.
Write in C.

Write in C, Write in C,
Write in C, yeah, Write in C.
Don't even mention COBOL.
Write in C.
~
~
18,26 All
```

- ▶ l'implementazione di `vi` che si trova nella maggior parte dei sistemi GNU/Linux
- ▶ Implementa molte caratteristiche dei moderni editor: evidenziazione del testo, storia dei comandi, aiuto in linea, undo senza limiti e molto altro.
- ▶ Esempio di una interessante caratteristica: si possono aprire direttamente i file di testo compressi.
- ▶ Ha una interfaccia grafica in GTK (`gvim`)
- ▶ Sfortunatamente non è software libero (per una piccola restrizione sulla libertà di fare cambiamenti)



Comandi base di vi

[Esc]	Switch to command mode (when in edit mode)
i	Insert text
a	Identical to insert command, except starts inserting at character after cursor
r	Replace character under cursor
R	Replace multiple characters
J	Join lines (default 2)
ndd	Delete n lines. Deleted lines are copied to buffer
nyy	Yank n lines. n lines starting at cursor line are copied to buffer
p	Paste contents of buffer after the line that contains the cursor
D	Delete from cursor position to end of line
nG	Goto line n. If n is not specified, goes to the last line
H	Goto first line of file
/string	Find next occurrence of string
:1,\$s/str1/str2/g	Replaces every occurrence of str1 with str2, starting from line 1 to the end of text
n	Find next occurrence of the last search string
?string	Find previous occurrence of string
^f	Go forward a page
^b	Go backward a page
h	Move cursor left
l	Move cursor right
j	Move cursor down
k	Move cursor up
^L	Redraw screen
u	Undo the latest change
U	Undo all changes on a line, while not having moved off of it
:wq	Save file and exit vi
:w name	Save to file "name"
:x,y w name	Writes lines x through y to file "name"
:q!	Exit vi without saving changes
:f	Displays file information on bottom on screen
ZZ	Write contents of buffer to current file and quit vi

Lancia il comando
vimtutor per imparare!
Servono solo 30 minuti per
completare il tutorial!



GNU nano

<http://www.nano-editor.org/>

- ▶ Un altro piccolo editor solo modalità testo, senza mouse.
- ▶ Una imitazione migliorata di Pico (editor non libero di Pine)
- ▶ Facile da usare e da conoscere per i principianti grazie all'elenco dei comandi presenti a video.
- ▶ Eseguibili disponibili per parecchie piattaforme.
- ▶ Un'alternativa a vi in sistemi embedded. Comunque non è disponibile all'interno di busybox.



Schermata di GNU nano

```
GNU nano 1.2.3           File: fortune.txt

The herd instinct among economists makes sheep look like independent thinkers.

Klingon phaser attack from front!!!!
100% Damage to life support!!!

Spock: The odds of surviving another attack are 13562190123 to 1, Captain.

Quantum Mechanics is God's version of "Trust me."

I'm a soldier, not a diplomat.  I can only tell the truth.
-- Kirk, "Errand of Mercy", stardate 3198.9

Did you hear that there's a group of South American Indians that worship
the number zero?

Is nothing sacred?

They are called computers simply because computation is the only significant
job that has so far been given to them.

As far as the laws of mathematics refer to reality, they are not
certain, and as far as they are certain, they do not refer to reality.
-- Albert Einstein

Tact, n.:
The unsaid part of what you're thinking.

Support bacteria -- it's the only culture some people have!

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Txt  ^T To Spell
```



Introduzione a Unix e GNU/Linux

Varie
Comprimere e archiviare



Determinare la dimensione di un file

▶ `du -h <file>` (uso del disco)

`-h`: restituisce la dimensione del file dato, in un formato leggibile: K (kilobytes), M (megabytes) o G (gigabytes)

Altrimenti, `du` restituisce il numero di blocchi di disco usati dal file (difficile da leggere).

Nota che l'opzione `-h` esiste solo nel `du` di GNU. Non è disponibile sul `du` di Sun Solaris, per esempio.

▶ `du -sh <dir>`

`-s`: restituisce la somma delle dimensioni di tutti i file della directory data.



Misurare lo spazio su disco

▶ `df -h <dir>`

Restituisce lo spazio del disco usato e libero per il filesystem contenente la directory data.

Analogamente, l'opzione `-h` esiste solo in GNU `df`.

▶ Esempio:

```
> df -h .
```

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/hda5	9.2G	7.1G	1.8G	81%	/

▶ `df -h`

Restituisce le informazioni per tutti i filesystems disponibili nel sistema. In caso d' errore è utile per trovare filesystem pieni.



Compressione

Utilissima per ridurre la dimensione di grossi file e risparmiare spazio

▶ `[un]compress <file>`

Utilità tradizionale di compressione di Unix. Crea file `.z` .
Tenuta per compatibilità. Medie prestazioni.

▶ `g[un]zip <file>`

Utilità di compressione di GNU. Crea file `.gz` .
Buone prestazioni (simili a Zip).

▶ `b[un]zip2 <file>`

La più recente e migliore utilità di compressione. Crea file `.bz2` .
Di solito comprime il 20-25% più di `gzip`.

Usate questa! Ora è disponibile in tutti i sistemi Unix.



Archiviazione (1)

Utile per fare copie o rilasciare un insieme di file come file unico

▶ `tar`: in origine: “archivio a nastro”

▶ Creare un archivio:

```
tar cvf <archive> <files or directories>
```

`c`: crea

`v`: con molti messaggi. Utile per verificare i progressi

`f`: file. L'archivio è creato in un file (altrimenti usa il nastro)

▶ Esempio:

```
tar cvf /backup/home.tar /home
```

```
bzip2 /backup/home.tar
```



Archiviazione (2)

- ▶ Vedere il contenuto di un archivio o verificarne l'integrità:

```
tar tvf <archive>
```

```
t: test
```

- ▶ Estrarre tutti i file da un archivio:

```
tar xvf <archive>
```

- ▶ Estrarre solo alcuni file da un archivio:

```
tar xvf <archivio> <file o directory>
```

I file o le directory sono indicati con un path relativo alla directory base dell'archivio.



Altre opzioni di GNU tar

tar = gtar = GNU tar su GNU/Linux

Può comprimere e decomprimere archivi al volo. Utile per evitare di creare enormi file intermedi.

Molto più semplice che usare tar e bzip2!

▶ opzione j: [s]comprime al volo con bzip2

▶ opzione z: [s]comprime al volo con gzip

▶ Esempi (quale ricordi meglio?)

▶ `gtar jcvf bills_bugs.tar.bz2 bills_bugs`



▶ `tar cvf - bills_bugs | bzip2 > bills_bugs.tar.bz2`



Il comando wget

Invece di scaricare i file dal tuo browser, fa il copia e incolla dell'URL e scaricali con `wget`!

Principali caratteristiche di `wget`

- ▶ supporto di `http` e `ftp`
- ▶ Può continuare download interrotti
- ▶ Può scaricare interi siti o almeno verificare collegamenti corrotti
- ▶ Molto utile negli scripts o quando non c'è disponibile la grafica (gestione di sistema, sistemi embedded)
- ▶ Supporto proxy (variabili d'ambiente `http_proxy` e `ftp_proxy`)



Esempi di wget

- ▶ `wget -c \`
`http://microsoft.com/customers/dogs/winxp4dogs.zip`
Continua un download interrotto
- ▶ `wget -m http://lwn.net/`
Esegue la copia di un sito
- ▶ `wget -r -np http://www.xml.com/ldd/chapter/book/`
Scarica ricorsivamente un libro on-line per un accesso off-line.
-np: "no-parent". Segue solo i link nella directory corrente.



Verifica dell'integrità dei file

Una soluzione molto veloce per la verifica dell'integrità dei file

▶ `md5sum FC3-i386-disk*.iso > MD5SUM`

Calcola un MD5 (Message Digest Algorithm 5) 128 bit checksum del file dato. Di solito è ridiretto su un file.

▶ Esempio di output:

```
db8c7254beeb4f6b891d1ed3f689b412 FC3-i386-disc1.iso
2c11674cf429fe570445afd9d5ff564e FC3-i386-disc2.iso
f88f6ab5947ca41f3cf31db04487279b FC3-i386-disc3.iso
6331c00aa3e8c088cc365eeb7ef230ea FC3-i386-disc4.iso
```

▶ `md5sum -c MD5SUM`

Verifica l'integrità dei file in MD5SUM verificando il loro effettivo MD5 con quello originale.



Introduzione a Unix e GNU/Linux

Varie
Stampa



La stampa in Unix

- ▶ È multi-utente, multi-processo, multi-cliente, multi-stampante
In Unix/Linux, i comandi di stampa non stampano realmente. Mandano la stampa ad una coda di stampa, che può essere sulla macchina locale, su server di stampa in rete o su una stampante di rete.
- ▶ Sistema di stampa indipendente:
I server di stampa accettano solo stampe in PostScript o testo. I driver di stampa sul server si occupano di convertire i dati in funzione del tipo di stampante.
- ▶ Sistema robusto:
Se il sistema viene riavviato, stamperà i lavori sospesi.



Comandi di stampa

- ▶ Una comoda variabile d'ambiente: `PRINTER`
Imposta la stampante di default del sistema. Esempio:
`export PRINTER=lp`
- ▶ `lpr [-P<coda>] <file>`
Manda i file dati alla coda di stampa specificata
I file devono essere in formato testo o in Postscript. Altrimenti stampa solo simboli senza senso.
- ▶ `a2ps [-P<coda>] <file>`
“Any to PostScript” converte molti formati a PostScript e invia l'output alla coda specificata. Caratteristiche utili: parecchie pagine per foglio, numerazione delle pagine, informazioni extra...



Controllo delle code di stampa

▶ `lpq [-P<code>]`

Elenco di tutti i lavori di stampa sulla coda data o sulla coda di default.

```
lp is not ready
Rank      Owner    Job      File(s)                Total Size
1st       asloane  84       nsa_windows_backdoors.ps 60416 bytes
2nd       amoore   85       gw_bush_iraq_mistakes.ps 65024000 bytes
```

▶ `cancel <job#> [<coda>]`

Rimuove il lavoro identificato dal numero dato dalla coda di default.



Usare file PostScript e PDF

Vedere un file PostScript

- ▶ Esistono visualizzatori di PostScript, ma hanno una scarsa qualità.
- ▶ Meglio convertirli a PDF con `ps2pdf`:
`ps2pdf decss_algorithm.ps`
`xpdf decss_algorithm.pdf &`

Stampare un file PDF

- ▶ Non serve un visualizzatore di file PDF!
- ▶ Meglio convertirli a PostScript con `pdf2ps`:
`pdf2ps rambaldi_artifacts_for_dummies.pdf`
`lpr rambaldi_artifacts_for_dummies.ps`



Introduzione a Unix e GNU/Linux

Varie
Confrontare file e directory



Confrontare file e directory

▶ `diff file1 file2`

Elenca le differenze tra 2 file, o non mostra niente se sono uguali.

▶ `diff -r dir1/ dir2/`

Elenca tutte le differenze tra i file con lo stesso nome nelle 2 directory.

▶ Per vedere le differenze in dettaglio, è meglio usare uno strumento grafico!



tkdiff

<http://tkdiff.sourceforge.net/>

Strumento grafico utile per confrontare file e fonderli

```
75 machine-$(CONFIG_ARCH_C0285) := footbridge
76 incdir-$(CONFIG_ARCH_C0285) := ebsa285
77 - machine-$(CONFIG_ARCH_FTVPXI) := ftvpci
78 - incdir-$(CONFIG_ARCH_FTVPXI) := nexuspki
79 - machine-$(CONFIG_ARCH_TBOX) := tbox
80 machine-$(CONFIG_ARCH_SHARK) := shark
81 machine-$(CONFIG_ARCH_SAI100) := sai100
82 ifeq ($(CONFIG_ARCH_SAI100),y)
83 # SAI111 DMA bug: we don't want the kernel to live in p
84 textaddr-$(CONFIG_SAI111) := 0xc0208000
85 endif
86 machine-$(CONFIG_ARCH_PXA) := pxa
87 machine-$(CONFIG_ARCH_L7200) := l7200
88 machine-$(CONFIG_ARCH_INTEGRATOR) := integrator
89 machine-$(CONFIG_ARCH_CAMELOT) := epxa10db
90 textaddr-$(CONFIG_ARCH_CLPS711X) := 0xc0028000
91 machine-$(CONFIG_ARCH_CLPS711X) := clps711x
92 textaddr-$(CONFIG_ARCH_FORTUNET) := 0xc0008000
93 machine-$(CONFIG_ARCH_IOP3XX) := iop3xx
94 ! machine-$(CONFIG_ARCH_ADI_FCC) := adi_fcc
95 machine-$(CONFIG_ARCH_OMAP) := omap
96 machine-$(CONFIG_ARCH_S3C2410) := s3c2410
97 machine-$(CONFIG_ARCH_LH7A40X) := lh7a40x
98 machine-$(CONFIG_ARCH_VERSATILE_PB) := versatile
99
100 TEXTADDR := $(textaddr-y)

76 machine-$(CONFIG_ARCH_C0285) := footbridge
77 incdir-$(CONFIG_ARCH_C0285) := ebsa285
78
79 machine-$(CONFIG_ARCH_SHARK) := shark
80 machine-$(CONFIG_ARCH_SAI100) := sai100
81 ifeq ($(CONFIG_ARCH_SAI100),y)
82 # SAI111 DMA bug: we don't want the kernel to live in p
83 textaddr-$(CONFIG_SAI111) := 0xc0208000
84 endif
85 machine-$(CONFIG_ARCH_PXA) := pxa
86 machine-$(CONFIG_ARCH_L7200) := l7200
87 machine-$(CONFIG_ARCH_INTEGRATOR) := integrator
88 machine-$(CONFIG_ARCH_CAMELOT) := epxa10db
89 textaddr-$(CONFIG_ARCH_CLPS711X) := 0xc0028000
90 machine-$(CONFIG_ARCH_CLPS711X) := clps711x
91 textaddr-$(CONFIG_ARCH_FORTUNET) := 0xc0008000
92 machine-$(CONFIG_ARCH_IOP3XX) := iop3xx
93 ! machine-$(CONFIG_ARCH_IXP4XX) := ixp4xx
94 machine-$(CONFIG_ARCH_OMAP) := omap
95 machine-$(CONFIG_ARCH_S3C2410) := s3c2410
96 machine-$(CONFIG_ARCH_LH7A40X) := lh7a40x
97 machine-$(CONFIG_ARCH_VERSATILE_PB) := versatile
98 +ifeq ($(CONFIG_ARCH_EBSA110),y)
99 +# This is what happens if you forget the IOCS16 line.
100 +# PCMCIA cards stop working.
101 +CFLAGS_3c589_cs.o := -DISA_SIXTEEN_BIT_PERIPHERAL
102 +export CFLAGS_3c589_cs.o
103 +endif
104 +
105 TEXTADDR := $(textaddr-y)
```

Introduzione a Unix e GNU/Linux

© Copyright 2006-2004, Michael Opdenacker

Creative Commons Attribution-ShareAlike 2.0 license

<http://free-electrons.com>

9 ago 2006



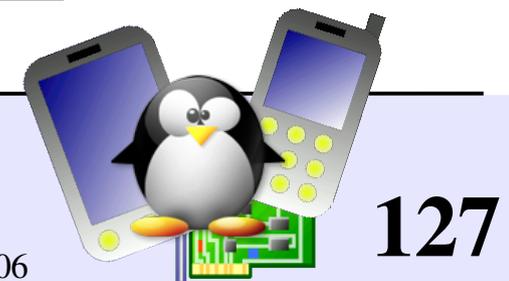
kompare

Un altro strumento ben fatto per confrontare file e fonderli
Parte del pacchetto kdesdk (Fedora Core)

```
File Difference Settings Help
Makefile
76 incdir-$(CONFIG_ARCH_C0285) := ebsa285
77 machine-$(CONFIG_ARCH_FTVPCCI) := ftvpcci
78 incdir-$(CONFIG_ARCH_FTVPCCI) := nexuspcci
79 machine-$(CONFIG_ARCH_TBOX) := tbox
80 machine-$(CONFIG_ARCH_SHARK) := shark
81 machine-$(CONFIG_ARCH_SA1100) := sa1100
82 ifeq ($(CONFIG_ARCH_SA1100),y)
83 # SA1111 DMA bug: we don't want the kernel to live in p
84 textaddr-$(CONFIG_SA1111) := 0xc0208000
85 endif
86 machine-$(CONFIG_ARCH_PXA) := pxa
87 machine-$(CONFIG_ARCH_L7200) := l7200
88 machine-$(CONFIG_ARCH_INTEGRATOR) := integrator
89 machine-$(CONFIG_ARCH_CAMELOT) := epxa10db
90 textaddr-$(CONFIG_ARCH_CLPS711X) := 0xc0028000
91 machine-$(CONFIG_ARCH_CLPS711X) := clps711x
92 textaddr-$(CONFIG_ARCH_FORTUNET) := 0xc0008000
93 machine-$(CONFIG_ARCH_IOP3XX) := iop3xx
94 machine-$(CONFIG_ARCH_ADIFCC) := adifcc
95 machine-$(CONFIG_ARCH_OMAP) := omap
96 machine-$(CONFIG_ARCH_S3C2410) := s3c2410
97 machine-$(CONFIG_ARCH_LH7A40X) := lh7a40x
98 machine-$(CONFIG_ARCH_VERSATILE_PB) := versatile
99
100 TEXTADDR := $(textaddr-y)
101 ifeq ($(incdir-y),)
102 incdir-y := $(machine-y)
103 endif
104 INCDIR := arch-$(incdir-y)
105
106 export TEXTADDR GZFLAGS
107

Makefile
75 incdir-$(CONFIG_FOOTBRIDGE) := ebsa285
75 textaddr-$(CONFIG_ARCH_C0285) := 0x60008000
76 machine-$(CONFIG_ARCH_C0285) := footbridge
77 incdir-$(CONFIG_ARCH_C0285) := ebsa285
78 machine-$(CONFIG_ARCH_SHARK) := shark
79 machine-$(CONFIG_ARCH_SA1100) := sa1100
80 ifeq ($(CONFIG_ARCH_SA1100),y)
82 # SA1111 DMA bug: we don't want the kernel to live in p
83 textaddr-$(CONFIG_SA1111) := 0xc0208000
84 endif
85 machine-$(CONFIG_ARCH_PXA) := pxa
86 machine-$(CONFIG_ARCH_L7200) := l7200
87 machine-$(CONFIG_ARCH_INTEGRATOR) := integrator
88 machine-$(CONFIG_ARCH_CAMELOT) := epxa10db
89 textaddr-$(CONFIG_ARCH_CLPS711X) := 0xc0028000
89 machine-$(CONFIG_ARCH_CLPS711X) := clps711x
90 textaddr-$(CONFIG_ARCH_FORTUNET) := 0xc0008000
91 machine-$(CONFIG_ARCH_IOP3XX) := iop3xx
92 machine-$(CONFIG_ARCH_IXP4XX) := ixp4xx
93 machine-$(CONFIG_ARCH_OMAP) := omap
94 machine-$(CONFIG_ARCH_S3C2410) := s3c2410
95 machine-$(CONFIG_ARCH_LH7A40X) := lh7a40x
96 machine-$(CONFIG_ARCH_VERSATILE_PB) := versatile
97
98 ifeq ($(CONFIG_ARCH_EBSA110),y)
99 # This is what happens if you forget the IOCS16 line.
100 # PCMCIA cards stop working.
101 CFLAGS_3c589_cs.o := -DISA_SIXTEEN_BIT_PERIPHERAL
102 export CFLAGS_3c589_cs.o
103 endif
104
105 TEXTADDR := $(textaddr-y)
```

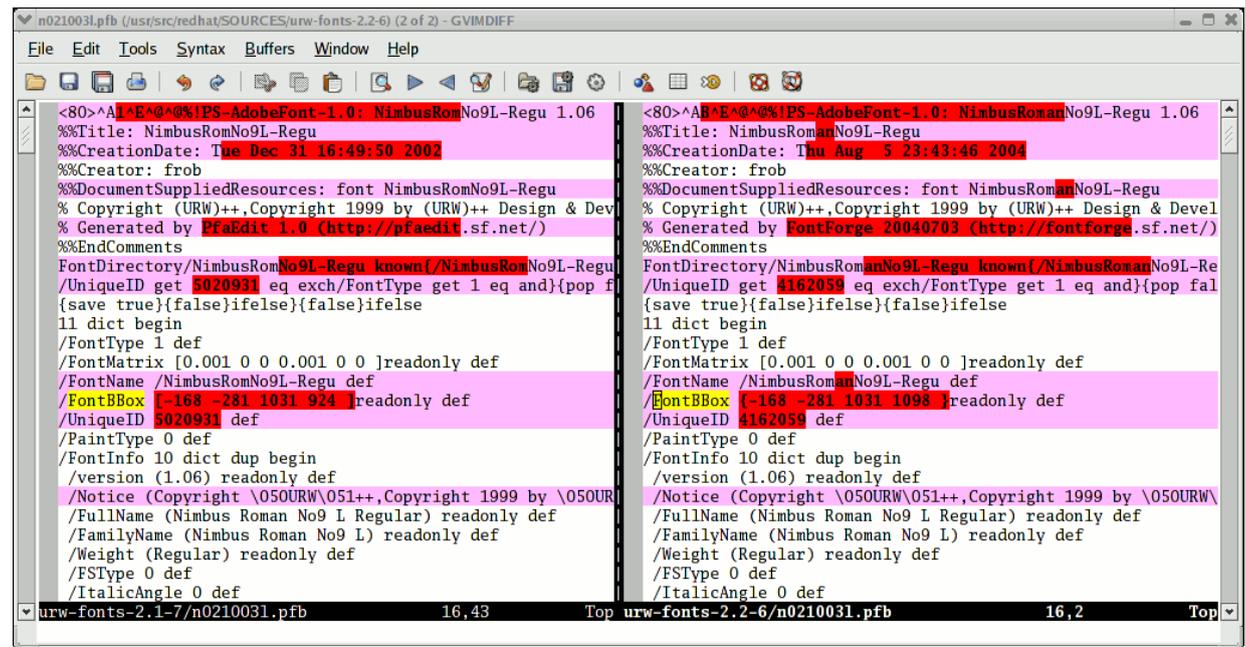
Comparing file file:/data/mike/handhelds/stock_kernel/linux-2.6....data/mike/handhelds/stock_kernel/linux-2.6.8.1/arch/arm/Makefile 1 of 11 differences, 0 applied 1 of 1 file



gvimdiff

Un'altro programma per vedere le differenze

Disponibile in molte
distribuzioni con gvim
Non usa diff.
Non funziona con file
con blocchi binari!



```
<80>^A!^E^Q^@%!PS-AdobeFont-1.0: NimbusRomNo9L-Regu 1.06
%%Title: NimbusRomNo9L-Regu
%%CreationDate: Tue Dec 31 16:49:50 2002
%%Creator: frob
%%DocumentSuppliedResources: font NimbusRomNo9L-Regu
% Copyright (URW)++,Copyright 1999 by (URW)++ Design & Dev
% Generated by PfaEdit 1.0 (http://pfaedit.sf.net/)
%%EndComments
FontDirectory/NimbusRomNo9L-Regu known{/NimbusRomNo9L-Regu
/UniqueID get 5020931 eq exch/FontType get 1 eq and}{pop f
(save true){false}ifelse}{false}ifelse
11 dict begin
/FontType 1 def
/FontMatrix [0.001 0 0 0.001 0 0 ]readonly def
/FontName /NimbusRomNo9L-Regu def
/FontBBox [-168 -281 1031 924 ]readonly def
/UniqueID 5020931 def
/PaintType 0 def
/FontInfo 10 dict dup begin
/version (1.06) readonly def
/Notice (Copyright \05OURW\051++,Copyright 1999 by \05OURW
/FullName (Nimbus Roman No9 L Regular) readonly def
/FamilyName (Nimbus Roman No9 L) readonly def
/Weight (Regular) readonly def
/FSType 0 def
/ItalicAngle 0 def
<80>^A!^E^Q^@%!PS-AdobeFont-1.0: NimbusRomanNo9L-Regu 1.06
%%Title: NimbusRomanNo9L-Regu
%%CreationDate: Thu Aug 5 23:43:46 2004
%%Creator: frob
%%DocumentSuppliedResources: font NimbusRomanNo9L-Regu
% Copyright (URW)++,Copyright 1999 by (URW)++ Design & Devel
% Generated by FontForge 20040703 (http://fontforge.sf.net/)
%%EndComments
FontDirectory/NimbusRomanNo9L-Regu known{/NimbusRomanNo9L-Re
/UniqueID get 4162059 eq exch/FontType get 1 eq and}{pop fal
(save true){false}ifelse}{false}ifelse
11 dict begin
/FontType 1 def
/FontMatrix [0.001 0 0 0.001 0 0 ]readonly def
/FontName /NimbusRomanNo9L-Regu def
/FontBBox [-168 -281 1031 1098 ]readonly def
/UniqueID 4162059 def
/PaintType 0 def
/FontInfo 10 dict dup begin
/version (1.06) readonly def
/Notice (Copyright \05OURW\051++,Copyright 1999 by \05OURW\
/FullName (Nimbus Roman No9 L Regular) readonly def
/FamilyName (Nimbus Roman No9 L) readonly def
/Weight (Regular) readonly def
/FSType 0 def
/ItalicAngle 0 def
```



Introduzione a Unix e GNU/Linux

Misc
Cercare file



Il comando find

Meglio mostrare alcuni esempi!

▶ `find . -name "*.pdf"`

Elenco di tutti i file `*.pdf` nella directory corrente (`.`) o in sottodirectory. Servono i doppi apici per evitare che la shell espanda il carattere `*`.

▶ `find docs -name "*.pdf" -exec xpdf {} ';'`

Trova tutti i file `*.pdf` nella directory `docs` e li mostra in sequenza.

▶ Vi sono molte altre possibilità! Comunque i 2 esempi mostrati risolvono molti problemi.



Il comando locate

Un'alternativa per ricerche con espressioni regolari molto più veloce di `find`

▶ `locate keys`

Elenco di tutti i file nel sistema con `keys` nel nome.

▶ `locate "*.pdf"`

Elenco di tutti i file `*.pdf` disponibili nell'intera macchina

▶ `locate "/home/fridge/*beer*"`

Elenco di tutti i file `*beer*` nella data directory (path assoluto)

▶ `locate` è molto più veloce perché ha un indice di tutti i file in un database dedicato, che è regolarmente aggiornato.

▶ `find` è migliore per cercare su file recenti.



Introduzione a Unix e GNU/Linux

Varie Comandi vari



Informazioni sugli utenti

- ▶ `who`
Elenco di tutti gli utenti presenti sul sistema
- ▶ `whoami`
Mi dice con che nome sono entrato nel sistema
- ▶ `groups`
Mi dice a quale gruppi appartengo
- ▶ `groups <user>`
Mi dice a quali gruppi appartiene l'utente `<user>`
- ▶ `finger <user>`
Mi fornisce dettagli (nome, ecc.) su `<user>`
Disabilitato in alcuni sistemi (per ragioni di sicurezza)



Cambiare nome utente

Non serve eseguire un logout e un login per cambiare utente!

▶ `su hyde`

(Raro) Diventa l'utente `hyde`, ma mantiene le variabili di ambiente dell'utente originale.

▶ `su - jekyll`

(Più frequente) Diventa l'utente `jekyll`, con esattamente le stesse impostazioni del nuovo utente.

▶ `su -`

Se non viene dato un argomento, significa utente `root`.



Comandi vari (1)

▶ `sleep 60`

Aspetta per 60 secondi (non usa risorse di sistema)

▶ `wc report.txt` (conteggio parole)

```
438  2115 18302 report.txt
```

Conta il numero delle linee, parole e caratteri nel file dato o dello standard input.



Comandi vari (2)

▶ `bc` ("calcolatrice?")

`bc` è una piccola ma potente calcolatrice. Include anche un linguaggio di programmazione! Usa l'opzione `-l` per avere il supporto della libreria matematica standard.

▶ `date`

Restituisce la data attuale. Utile negli script per registrare il tempo di inizio e fine esecuzione dei comandi.



Introduzione a Unix e GNU/Linux

Cenni di gestione di sistema



Proprietà dei file

- ▶ `chown -R sco /home/linux/src` (-R: ricorsivo)
Rendi l'utente `sco` il nuovo proprietario di tutti i file in `/home/linux/src`
- ▶ `chgrp -R empire /home/askywalker`
Rendi `empire` il nuovo gruppo di tutto ciò che è in `/home/askywalker`
- ▶ `chown -R borg:aliens usss_entreprise/`
`chown` può essere usato per cambiare il proprietario e il gruppo allo stesso tempo.



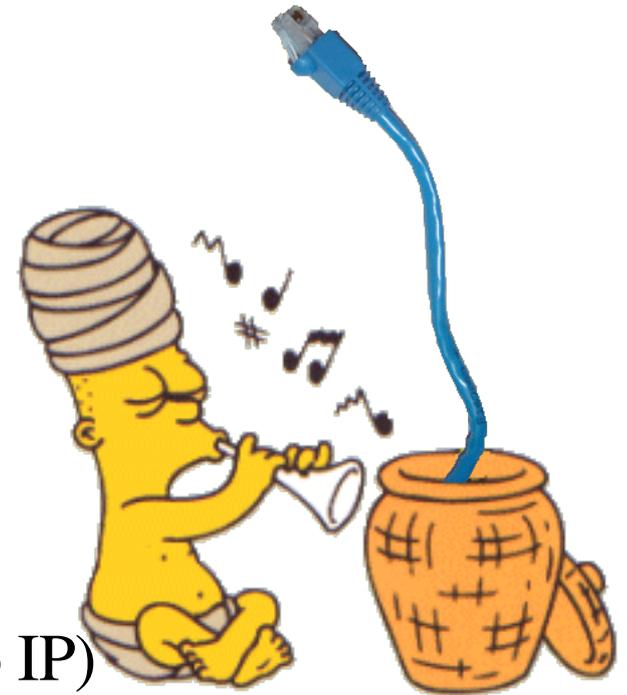
Spegnimento

- ▶ `shutdown -h +5 (-h: halt)`
Spegne il sistema dopo 5 minuti. Gli utenti vedono un avviso sulle loro console.
- ▶ `shutdown -r now (-r: reboot)`
- ▶ `init 0`
Un'altro modo di spegnere (usato da `shutdown`)
- ▶ `init 6`
Un'altro modo per eseguire il reboot (usato da `shutdown`)
- ▶ `[Ctrl][Alt][Del]`
Funziona anche questo su GNU/Linux (almeno sui PC!)



Installazione della rete (1)

- ▶ `ifconfig -a`
Mostra dettagli su tutte le interfacce di rete disponibili nel sistema.
- ▶ `ifconfig eth0`
Mostra i dettagli relativi all'interfaccia `eth0`
- ▶ `ifconfig eth0 192.168.0.100`
Assegna l'indirizzo `192.168.0.100` IP a `eth0` (1 indirizzo IP per interfaccia)
- ▶ `ifconfig eth0 down`
Spegne l'interfaccia `eth0` (libera il suo indirizzo IP)



Installazione della rete (2)

▶ `route add default gw 192.168.0.1`

Imposta l'instradamento di default per pacchetti diretti fuori dalla rete locale. Il gateway (qui 192.168.0.1) è responsabile per inviarli al prossimo gateway, ecc., fino alla destinazione finale.

▶ `route`

Mostra la tabella di instradamento attuale

▶ `route del default`

`route del <IP>`

Cancella la regola data

Utile per definire una nuova regola di instradamento.



Verifica della rete

▶ ping freshmeat.net
ping 192.168.1.1

Prova ad inviare pacchetti alla macchina indicata e aspetta un pacchetto di risposta.

```
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.  
64 bytes from 192.168.1.1: icmp_seq=0 ttl=150 time=2.51 ms  
64 bytes from 192.168.1.1: icmp_seq=1 ttl=150 time=3.16 ms  
64 bytes from 192.168.1.1: icmp_seq=2 ttl=150 time=2.71 ms  
64 bytes from 192.168.1.1: icmp_seq=3 ttl=150 time=2.67 ms
```

- ▶ Quando il ping riesce sul proprio gateway, l'interfaccia di rete è a posto.
- ▶ Quando si riesce a fare ping ad un indirizzo esterno, la configurazione della rete è corretta!



Riassunto sull'installazione della rete

Solo per casi semplici, senza server dhcp ...

- ▶ Connetti alla rete (cavo, scheda wireless ...)
- ▶ Identifica l'interfaccia di rete:
`ifconfig -a`
- ▶ Assegna un indirizzo IP all'interfaccia (esempio eth0)
`ifconfig eth0 192.168.0.100` (esempio)
- ▶ Aggiungi una tabella di routing al gateway (assumiamo 192.168.0.1) per pacchetti diretti alla rete esterna:
`route add default gw 192.168.0.1`



Risoluzione dei nomi

- ▶ I programmi devono conoscere a quale IP corrisponde un dato nome di macchina (come `kernel.org`)
- ▶ Domain Name Servers (DNS) si occupa di questo.
- ▶ Devi solo specificare l'indirizzo IP di 1 o più server DNS nel file `/etc/resolv.conf`:

```
nameserver 217.19.192.132  
nameserver 212.27.32.177
```
- ▶ Le modifiche hanno effetto immediato!



Creare un filesystem

Esempi

▶ `mkfs.ext2 /dev/sda1`

Formatta la chiave USB (`/dev/sda1`: 1^{ma} partizione) in formato ext2

▶ `mkfs.ext2 -F disk.img`

Formatta una immagine di disco in formato ext2

▶ `mkfs.vfat -v -F 32 /dev/sda1` (-v: verbose)

Formatta la chiave USB in formato FAT32

▶ `mkfs.vfat -v -F 32 disk.img`

Formatta una immagine di disco in formato FAT32

Una immagine vuota può essere creata come mostrato nell'esempio:

```
dd if=/dev/zero of=disk.img bs=1024 count=65536
```



Montare i dispositivi (1)

- ▶ Per rendere visibile un filesystem presente su un dispositivo qualsiasi (interno o esterno), bisogna montarlo (*mount*).
- ▶ La prima volta crea una directory per montarlo nel sistema:
`mkdir /mnt/usbdisk` (esempio)
- ▶ Ora, montalo:
`mount -t vfat /dev/sda1 /mnt/usbdisk`
`/dev/sda1`: dispositivo fisico
`-t`: specifica il tipo di filesystem
(`ext2`, `ext3`, `vfat`, `reiserfs`, `iso9660`...)



Montare i dispositivi (2)

- ▶ Sono disponibili molte opzioni, in particolare per scegliere i permessi o il proprietario dei file ed il gruppo di appartenenza... Vedi la pagina del manuale di `mount` per i dettagli.
- ▶ Le opzioni di montaggio possono essere scritte nel file `/etc/fstab`.
- ▶ Si può montare anche una immagine di un filesystem presente in un file (*loopback device*)
 - ▶ Utile per accedere al contenuto di una immagine ISO di un cdrom senza doverlo masterizzare.
 - ▶ Utile per creare una partizione Linux su un hard disk con sole partizioni Windows

```
cp /dev/sda1 usbkey.img  
mount -o loop -t vfat usbkey.img /mnt/usbdisk
```



Elenco dei filesystem montati

- ▶ Usa il comando `mount` senza alcun parametro:

```
/dev/hda6 on / type ext3 (rw,noatime)
none on /proc type proc (rw,noatime)
none on /sys type sysfs (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/hda4 on /data type ext3 (rw,noatime)
none on /dev/shm type tmpfs (rw)
/dev/hda1 on /win type vfat (rw,uid=501,gid=501)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
```

- ▶ Oppure visualizza il file `/etc/mtab`
(stesso risultato: è aggiornato da `mount` e `umount` ogni volta che vengono eseguiti)



Smontare i dispositivi

▶ `umount /mnt/usbdisk`

Esegue tutte le scritture in sospeso e smonta il dispositivo, che può essere rimosso in maniera sicura.

▶ Per poter smontare un dispositivo, bisogna chiudere tutti i file aperti su di esso:

▶ Chiudi una applicazione che accede a dati nella partizione

▶ Verifica che nessuna shell abbia la sua directory di lavoro su questo dispositivo.

▶ Puoi eseguire il comando `lsof` (**list open files**) per vedere quali processi hanno file aperti nella partizione montata.



Introduzione a Unix e GNU/Linux

Per continuare...



Aiuto sui comandi

Alcuni comandi Unix e la gran parte dei comandi GNU/Linux offrono almeno un argomento per avere un aiuto:

▶ `-h`

(`-` è usato per passare argomenti da 1 carattere)

▶ `--help`

(`--` è usato per passare i corrispondenti parametri con il nome lungo, per rendere gli script più leggibili)

Si ottiene un breve elenco di opzioni anche quando si inserisce un'opzione invalida.



Pagine del manuale

`man <keyword>`

Mostra una o più pagine corrispondenti a `<keyword>`

▶ `man man`

Mostra le pagine di manuale disponibili su comandi Unix, ma anche alcune relative a funzioni C, header o strutture di dati, o anche su file di configurazione di sistema!

▶ `man stdio.h`

▶ `man fstab (for /etc/fstab)`

Le pagine del manuale vengono cercate nelle directory specificate dalla variabile d'ambiente `MANPATH`.



Pagine info

- ▶ In GNU, le pagine del manuale si stanno sostituendo con le pagine info. Alcune pagine di manuale rimandano alle relative pagine info.

`info <command>`

- ▶ caratteristiche di `info`:

- ▶ Documentazione strutturata in sezioni (“nodi”) e sottosezioni (“sottonodi”)
- ▶ Possibilità di navigare in questa struttura: `indice`, `succ`, `prec`, `su`
- ▶ Le pagine info sono generate dagli stessi sorgenti `texinfo` delle pagine di documentazione HTML



Cercare risorse in internet (1)

Come cercare

- ▶ Molti gruppi di discussione e archivi di mailing list sono pubblici, e sono indicizzati spesso da **Google**.
- ▶ Se cerchi un messaggio d'errore, copialo esattamente in un motore di ricerca, con i doppi apici (“messaggio d'errore”). Ci sono molte possibilità che qualcun altro abbia già trovato lo stesso problema.
- ▶ Non dimenticare i gruppi di Google: <http://groups.google.com/>
Questo sito indicizza più di 20 anni di messaggi.



Cercare risorse in internet (2)

Cercare documentazione

Cerca `<tool>` o `<tool>` page per trovare il programma o il sito del progetto e da qui la documentazione aggiornata.

▶ Cerca `<tool>` documentation o `<tool>` manual nel motore di ricerca preferito.

Cercare informazioni tecniche generiche

Wikipedia: <http://wikipedia.org>

Molte utili definizioni di informatica. Una vera enciclopedia!
Aperta al contributo di chiunque.



Introduzione a Unix e GNU/Linux

Per continuare ...
usando GNU/Linux a casa



Panoramica di applicazioni desktop

Da mostrare a video con un proiettore!

- ▶ Mozilla: navigatore internet, cliente di posta, editor HTML
- ▶ Firefox: versione leggera di Mozilla
- ▶ OpenOffice: completo, compatibile con il pacchetto MS Office: word processor, spreadsheet, presentazioni, grafica...
- ▶ GIMP: editor grafico molto potente
- ▶ Gqview: visualizza album fotografici
- ▶ Evolution: gestore della posta e agenda tipo Outlook



Alternative GNU/Linux a Windows

Internet Explorer

IIS

Money

MS Office

MS Outlook

MS Project

Nero

Photoshop

WinAmp

W. Media Player

Mozilla

Firefox

Apache

GNU Cash

OpenOffice

Evolution

Mr Project
(Planner)

k3b

The GIMP

xmms

xine

mplayer

Non conosco abbastanza
programmi

Windows.



Mandaci altre segnalazioni!



GNU/Linux a casa (1)

GNU/Linux è anche una ottima alternativa a Windows a casa

Sicurezza

- ▶ **Nessun Virus**
La maggior parte dei virus sono disegnati per sfruttare problemi di sicurezza di Windows e non intaccano GNU/Linux
- ▶ **A prova di Virus**
Anche se esegui un virus compatibile con Linux, dovrebbe avere permessi di root per modificare il sistema.
- ▶ **A prova di errore**
Altri membri della famiglia non possono modificare il sistema o i file di qualcun'altro. Possono solo danneggiare i propri file.
- ▶ **Respinge i Cracker**
Anche se sempre connesso in rete, il vostro sistema attrae meno cracker.



GNU/Linux a casa (2)

Riservatezza

- ▶ Il tuo sistema non raccoglie dati di nascosto per trasmettere informazioni sui film che vedi o sui tuoi siti web preferiti.

Facile da usare

- ▶ I tuoi programmi sono fatti da utenti per altri utenti. Sono più facilmente adatti alle tue esigenze.
- ▶ Gli sviluppatori possono essere contattati per suggerire nuove caratteristiche.

Libertà

- ▶ I dati che produci sono tuoi per sempre. Non sono legati a una applicazione proprietaria con un formato proprietario (a volte brevettato!).
- ▶ Sei libero di aiutare i tuoi amici condividendo i tuoi programmi con loro.
- ▶ Sei libero di usare a casa i programmi che hai in ufficio!



GNU/Linux a casa (3)

Puoi passare a GNU/Linux per:

- ▶ Lavoro con OpenOffice: edito di testi, foglio elettronico, presentazioni, database
- ▶ Internet: navigazione web e posta elettronica
- ▶ Multimedia: video, suono e grafica (include fotocamere digitali)
- ▶ Imparare a usare i computer e a programmare

Se hai ancora una copia di Windows, puoi tenerla (assieme a Linux) per:

- ▶ Giocare. Molti produttori supportano solo Windows o Mac.
- ▶ Usare particolari programmi proprietari o CD educazionali
- ▶ Usare hardware non supportato da GNU/Linux



Prova GNU/Linux senza rischi

Knoppix è un cdrom live di GNU/Linux

<http://knoppix.net>

- ▶ Carica GNU/Linux in RAM, niente è installato sull'HD.
- ▶ Riconosce tantissimo hardware.
- ▶ Più di 2 GB di applicazioni disponibili!
- ▶ Puoi accedere ai tuoi file di Windows, aprirli e editarli con applicazioni GNU/Linux.
- ▶ Un modo fantastico di provare GNU/Linux!
- ▶ Offre la possibilità di fare una installazione permanente su HD



Usare distribuzioni GNU/Linux

Distribuzioni GNU/Linux

- ▶ Consentono di installare GNU/Linux su spazio libero del tuo hard disk, e tenere una copia di Windows (“doppio boot”)
- ▶ Avere una interfaccia di installazione molto facile da usare che riconosce automaticamente gran parte dell'hardware. Non devi installare alcun driver!
- ▶ Ti lascia scegliere quali tipi di applicazioni installare
- ▶ Fornisce interfacce di configurazione facili da usare
- ▶ Distribuzioni raccomandate per principianti:
Fedora Core o Mandrake



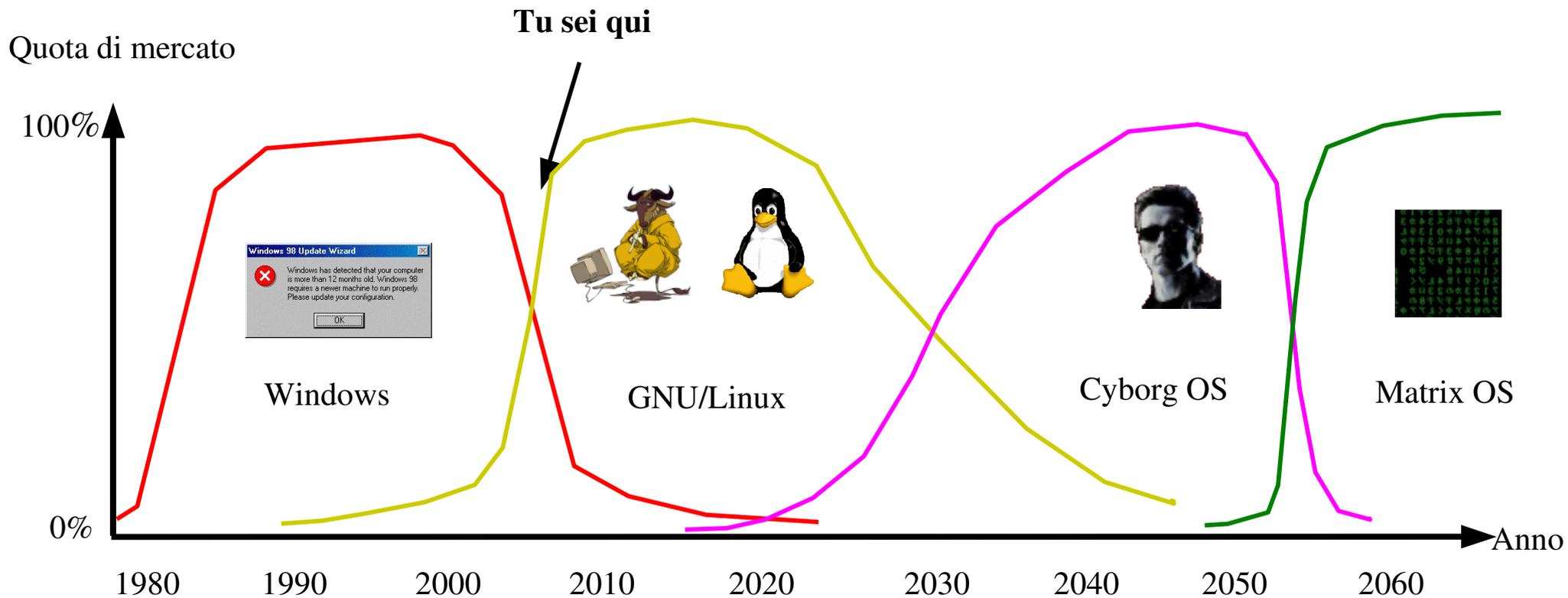
Introduzione a GNU/Linux

Conclusione



È ora di saltare sul treno!

Mappa dei Sistemi operativi



Introduzione a Unix e GNU/Linux

© Copyright 2006-2004, Michael Opdenacker

Creative Commons Attribution-ShareAlike 2.0 license

<http://free-electrons.com>

9 ago 2006



Documenti collegati

Questo documento è parte di più di 900 pagine di materiale relativo a corsi su embedded GNU/Linux tenuti da Free Electrons, disponibile con licenza libera.

- ▶ Introduzione a Unix e GNU/Linux
http://free-electrons.com/training/intro_unix_linux
- ▶ Sviluppo kernel e driver per Linux Embedded
<http://free-electrons.com/training/drivers>
- ▶ Strumenti di sviluppo per sistemi Linux embedded
<http://free-electrons.com/training/devtools>
- ▶ Audio in sistemi Linux embedded
<http://free-electrons.com/training/audio>
- ▶ Multimedia in sistemi Linux embedded
<http://free-electrons.com/training/multimedia>
- ▶ Java in sistemi Linux embedded
<http://free-electrons.com/articles/java>
- ▶ Novità in Linux 2.6?
<http://free-electrons.com/articles/linux26>
- ▶ Introduzione a uClinux
<http://free-electrons.com/articles/uclinux>
- ▶ Linux: estensioni in tempo reale
<http://free-electrons.com/articles/realtime>



Servizi di consulenza e istruzione

Queste lezioni o presentazioni sono finanziate dai clienti di Free Electrons che inviano i loro dipendenti ai nostri corsi o sessioni di consulto.

Se sei interessato a partecipare a corsi presentati dall'autore di questi documenti, sei invitato a chiedere alla tua organizzazione di ordinare corsi del genere.

Vedi <http://free-electrons.com/training> per altri dettagli.

Se vuoi aiutare in questo lavoro, non esitare a parlarne ad amici e colleghi e a gruppi locali di Free Software.

