From DOS/Windows to Linux HOWTO

di Guido Gonzato <guido@ibogfs.cineca.it> Versione 1.2.4. 29 Dicembre 1997.

Questo HOWTO e' dedicato a tutti gli utenti DOS che hanno deciso di passare a Linux, il clone Unix per PC 386 e superiori. Date le analogie tra DOS e Unix, lo scopo di questo lavoro e' di aiutare il lettore a trasportare le sue conoscenze di DOS nell'ambiente Linux, cosi' da poter lavorare da subito.

Contents

1 Introduzione

1.1 Linux fa per voi?

Volete passare dal DOS a Linux? Benissimo, ma attenzione: potrebbe non esservi utile. Credo che "il computer migliore" o "il migliore sistema operativo" non esistano: dipende dall'uso che se ne fa. Ecco perché non credo che Linux sia la soluzione migliore per tutti, nonostante sia tecnicamente superiore a molti sistemi operativi commerciali. Avrete grandi benefici da Linux se vi serve sw per la programmazione, Internet, TeX... sw tecnico in generale, ma se vi serve soprattutto sw commerciale, o se non vi piace l'idea di studiare i comandi, e' meglio lasciare perdere.

Linux non e' (per ora) facile da usare o da configurare come Windows o il Mac, quindi siate preparati a smanettare un po'. Nonstante questi avvertimenti, sono sicuro al 100% che se appartienete alla giusta categoria di utenti troverete in Linux il vostro Nirvana informatico. E comunque Linux e DOS/Windows possono convivere sulla stessa macchina.

Prerequisiti per questo howto: daro' per scontato che

- conoscete i principali comandi e concetti del DOS;
- Linux, e magari anche X Window System, e' correttamente installato;
- la vostra shell—l'equivalente di `COMMAND.COM`—e' `bash`;
- capite che questo lavoro e' solo un inizio. Per maggiori informazioni, guardatevi "Linux Installation and Getting Started" di Matt Welsh e/o "Linux User Guide" di Larry Greenfield (*sunsite.unc.edu:/pub/Linux/docs/LDP*).

Questo howto sostituisce il vecchio mini–howto di uguale titolo.

1.2 Si', fa per me. Spiegami

Avete installato Linux e i programmi che vi servono. Vi siete fatti un account (se non l'avete fatto, scrivete `adduser` *subito!*) e Linux sta girando. Avete inserito nome e password e ora state guardando lo schermo e pensate: "Beh, e adesso?"

Adesso, non disperate. Siete quasi pronti per fare le stesse cose che facevate col DOS, e molte altre in piu'. Se steste lavorando col DOS anziché con Linux, ora fareste una di queste cose:

- eseguire programmi e creare, copiare, visualizzare, cancellare, stampare e rinominare files;
- spostarsi tra directory, crearne di nuove, cancellarle, elencarne i contenuti;
- formattare floppy e copiarci file su/da;
- sistemare `AUTOEXEC.BAT` e `CONFIG.SYS`;
- scrivere i vostri .BAT files e/o programmi in `QBasic` o C/Pascal;
- il rimanente 1%.

Sarete contenti di sapere che queste cose si fanno con Linux in un modo molto simile al DOS. Sotto DOS, l'utente medio usa solo pochi tra i 100 e passa comandi disponibili; lo stesso vale per Linux, almeno fino ad un certo punto.

Alcune cose da aver chiare prima di proseguire:

- primo, come uscire da Linux. Se si vede una schermata non grafica, premere CTRL–ALT–DEL, aspettare che il sistema dica che tutto e' a posto, poi spegnere pure. Se si sta lavorando sotto X Window System, prima premere CTRL–ALT–BACKSPACE, poi CTRL–ALT–DEL. Non spegnere *mai* il PC direttamente: si potrebbe danneggiare il filesystem;
- a differenza del DOS, Linux ha meccanismi di sicurezza, a causa della sua natura multiutente. I file e le directory hanno dei permessi, e quindi ad alcuni l'utente normale non puo' accedere (vedi la sezione 2.3 (Permessi)). Solo l'utente il cui nome di login e' "root" puo' fare cio' che vuole (root e' l'amministratore di sistema. Se usate Linux sul vostro PC, sarete anche root). Il DOS, al contrario, vi lascia cancellare tutto l'hard disk per sbaglio;
- siete incoraggiati a sperimentare, giocare, provare: di certo male non fa. Potete ottenere aiuto in questo modo:
  - per ottenere aiuto sui "comandi interni" della shell, scrivete `help`;
  - per ottenere aiuto su un comando, scrivete `man command` che richiama la pagina di manuale (man page) del comando in questione. In alternativa, scrivete `info command` che richiama, se c'e', la pagina info relativa al comando. Info e' un sistema di documentazione ad ipertesti, non molto intuitivo da usare le prime volte. Potete provare inoltre a dare i comandi `whatis command` o `apropos command` e premere 'q' per uscire.
- buona parte della potenza e flessibilita' di Unix derivano dai semplici concetti di redirezione e piping, piu' potenti che non sotto DOS. Semplici comandi possono essere combinati per eseguire operazioni complesse. Usate questa caratteristica!
- convenzioni: <...> indica qualcosa che deve essere specificato, mentre `[...]` indica qualcosa di opzionale. Esempio:

  ```
  $ tar -tf <file.tar> [> redir_file]
  ```

  `file.tar` deve essere specificato, mentre la redirezione su `redir_file` e' opzionale.
- d'ora in avanti "LMP" significa "leggere la man page per ulteriori informazioni".

1.3 Per l'impaziente

Volete partire subito? Date un'occhiata a questa tabella:

```
DOS                     Linux                   Note
------------------------------------------------------------------------------

BACKUP                  tar -Mcvf device dir/   totalmente diversi
CD dirname\             cd dirname/             quasi la stessa sintassi
COPY file1 file2        cp file1 file2          idem
DEL file                rm file                 attenzione - niente undelete
DELTREE dirname         rm -R dirname/          idem
DIR                     ls                      non proprio la stessa sintassi
DIR file/s              find . -name file       totalmente diverso
EDIT file               vi file                 credo che non vi piacera'
                        emacs file              questo e' migliore
                        jstar file              quasi come l'editor del DOS
FORMAT                  fdformat,
                        mount, umount           sintassi molto diversa
HELP command            man command             stessa filosofia
MD dirname              mkdir dirname/          quasi la stessa sintassi
MOVE file1 file2        mv file1 file2          idem
NUL                     /dev/null               idem
PRINT file              lpr file                idem
PRN                     /dev/lp0,
                        /dev/lp1                idem
RD dirname              rmdir dirname/          quasi la stessa sintassi
REN file1 file2         mv file1 file2          non per file multipli
RESTORE                 tar -Mxpvf device       sintassi diversa
TYPE file               less file               molto migliore
WIN                     startx                  un mondo a parte!
```

Se vi serve di piu' che una tabella di comandi, leggetevi le prossime sezioni.

2 File e programmi

2.1 File: nozioni preliminari

Linux ha un file system—intendendo con cio' "la struttura delle directory e dei file in esse contenuti"—molto simile a quello del DOS. I file hanno dei nomi che seguono certe regole, sono messi in directory, alcuni sono eseguibili, e tra questi ultimi molti hanno degli switch. Inoltre, ci sono i caratteri wildcards, la redirezione e il piping. Ci sono solo alcune piccole differenze:

- sotto DOS, i nomi dei file seguono la regola dell'8.3; per esempio, ILMIOFIL.TXT. Sotto Linux si puo' fare meglio. Se avete installato Linux con un filesystem come ext2 o umsdos, potete usare nomi piu' lunghi, e con piu' di un punto: per esempio, `Ecco.un_NOME_molto.LUNGO`. Notare che ho usato lettere maiuscole e minuscole: infatti...
- maiuscole e minuscole sono considerate diverse. Quindi, `FILENAME.tar.gz` e `filename.tar.gz` sono due file diversi. `ls` e' un comando, `LS` e' un errore;

- gli utenti di Windows 95 possono usare i nomi di file lunghi con Linux, naturalmente. Se il nome di un file contiene spazi (pratica sconsigliata ma possibile), bisogna racchiudere il nome del file tra apici quando vi ci si riferisce. Per esempio:

```
$ # questo comando crea una directory di nome "I miei file vecchi"
$ mkdir "I miei file vecchi"
$ ls
I miei file vecchi      bin     tmp
```

  Alcuni caratteri non dovrebbero ma possono essere usati; tra gli altri, `!*$&`. Non vi spiego come.

- non ci sono estensioni obbligatorie come .COM ed .EXE per i programmi, o .BAT per i file batch. I files eseguibili sono contrassegnati da un asterisco `'*'` alla fine del loro nome quando si da' il comando `ls -F`. Per esempio:

```
$ ls -F
Direttorio/   cindy.jpg    cjpg*    lettera.txt    script*    vecchio~
```

  I files `cjpg*` e `script*` sono eseguibili—"programmi". Sotto DOS, i file di backup hanno il nome che finisce in .BAK, sotto Linux finiscono con una tilde '~'. Inoltre, un file il cui nome inizia con un punto viene considerato un file nascosto. Per esempio, il file `.io.sono.nascosto` non apparira' dopo il comando `ls`.

- gli switch dei programmi DOS si ottengono con `/switch`, con i programmi Linux si ottengono con `-switch` or `-switch`. Esempio: `dir /s` diventa `ls -R`. Notare che molti programmi DOS, come PKZIP o ARJ, hanno gli switch in stile Unix.

Ora potete saltare alla sezione 2.4 (Tradurre i comandi dal DOS a Linux), ma se fossi in voi continuerei a leggere.

2.2 Link simbolici

Unix ha un tipo di file che il DOS non ha: il link simbolico. Questo e' un puntatore ad un file o directory, e puo' essere usato al posto del file o directory a cui punta; e' molto simile ai colegamenti di Windows 95. Esempi di link simbolici sono `/usr/X11`, che punta a `/usr/X11R6`; `/dev/modem`, che punta a `/dev/cua0` o `/dev/cua1`.

Per fare un link simbolico:

```
$ ln -s <file_o_dir> <nomelink>
```

Esempio:

```
$ ln -s /usr/doc/g77/DOC g77manual.txt
```

Ora potete riferirvi a `g77manual.txt` al posto di `/usr/doc/g77/DOC`. Ecco come appaiono i link quando date `ls`:

```
$ ls -F
g77manual.txt@
$ ls -l
(various things...)           g77manual.txt -> /usr/doc/g77/DOC
```

2.3 Permessi e Proprieta'

I file e le directory DOS hanno i seguenti attributi: A (archivio), H (hidden, nascosto), R (read-only, solo lettura), and S (system, file di sistema). Solo H e R hanno senso sotto Linux: i files nascosti iniziano con un punto, e per quanto riguarda l'attributo R, continuate a leggere.

Sotto Unix un file ha dei "permessi" e un proprietario, che appartiene ad un "gruppo". Guardate questo esempio:

```
$ ls -l /bin/ls
-rwxr-xr-x  1  root  bin  27281 Aug 15 1995 /bin/ls*
```

Il primo campo contiene i permessi del file `/bin/ls`, che appartiene a root, gruppo bin. Tralasciando le altre informazioni (il libro di Matt Welsh e' li' per quello), ricordate che `-rwxr-xr-x` significa (da sinistra a destra):

- e' il tipo di file (- = file normale, d = directory, l = link, etc.); `rwx` sono i permessi per il proprietario del file (read, write, execute); `r-x` sono i permessi per il gruppo cui il proprietario del file appartiene (non spieghero' il concetto di gruppo, ne potete fare a meno finché siete principianti ;-) `r-x` sono i permessi per tutti gli altri utenti (read, execute).

Anche la directory `/bin` ha dei permessi: vedi la sezione 3.2 (Permessi delle directory) per ulteriori dettagli. Ecco perché non potete cancellare il file `/bin/ls`, a meno che non siate root: non avete i permessi necessari. Per cambiare i permessi di un file, il comando e':

```
$ chmod <whoXperm> <file>
```

dove who e' `u` (user, cioe' proprietario), `g` (group, gruppo), `o` (other, altri), X e' `+` o `-`, perm e' `r` (read, lettura), `w` (write, scrittura), o `x` (execute, esecuzione). Esempi:

```
$ chmod u+x file
```

setta il permesso di esecuzione per il proprietario. Scorciatoia: `chmod +x file`.

```
$ chmod go-rw file
```

toglie i permessi di lettura e scrittura per tutti tranne il proprietario.

```
$ chmod ugo+rwx file
```

da' a tutti i permessi di lettura, scrittura ed esecuzione.

```
# chmod +s file
```

rende un file "setuid" o "suid": ciascuno lo puo' eseguire con i permessi del proprietario. Di solito si incontrano file "suid root".

Una maniera piu' breve di riferirsi ai permessi e' con i numeri: `rwxr-xr-x` puo' essere espresso con 755 (ogni lettera corrisponde a un bit: `--` e' 0, `-x` e' 1, `-w-` e' 2, `-wx` e' 3...). Sembra difficile, ma con un po' di pratica capirete il concetto.

root, essendo il cossiddetto superutente, puo' cambiare i permessi di ogni file. C'e' molto di piu' sull'argomento—LMP.

2.4 Convertire i comandi dal DOS a Linux

Alla sinistra, i comandi DOS; a destra, i corrispondenti comandi Linux.

```
COPY:          cp
DEL:           rm
MOVE:          mv
REN:           mv
TYPE:          more, less, cat
```

Operatori per la redirezione e il plumbing: < > >> |

Wildcards: `* ?`

`nul:  /dev/null`

`prn, lpt1:  /dev/lp0 or /dev/lp1; lpr`

\- ESEMPI -

```
DOS                                    Linux
---------------------------------------------------------------------

C:\GUIDO>COPY JOE.TXT JOE.DOC          $ cp joe.txt joe.doc
C:\GUIDO>COPY *.* TOTAL                $ cat * > total
C:\GUIDO>COPY FRACTALS.DOC PRN         $ lpr fractals.doc
C:\GUIDO>DEL TEMP                      $ rm temp
C:\GUIDO>DEL *.BAK                     $ rm *~
C:\GUIDO>MOVE PAPER.TXT TMP\           $ mv paper.txt tmp/
C:\GUIDO>REN PAPER.TXT PAPER.ASC       $ mv paper.txt paper.asc
C:\GUIDO>PRINT LETTER.TXT              $ lpr letter.txt
C:\GUIDO>TYPE LETTER.TXT               $ more letter.txt
C:\GUIDO>TYPE LETTER.TXT               $ less letter.txt
C:\GUIDO>TYPE LETTER.TXT > NUL         $ cat letter.txt > /dev/null
        n/a                            $ more *.txt *.asc
        n/a                            $ cat section*.txt | less
```

Note:

- * e' migliore sotto Linux: * prende tutti i file tranne quelli nascosti; .* tutti i file nascosti (ma anche la directory corrente '`.`' e la directory genitrice '`..`'); *.* prende tutti i file che hanno un '.' nel mezzo o che finiscono per punto; p*a prende sia 'pera' che 'palla'; *l* prende sia 'mela' che 'filo';
- usando `more`, premere SPAZIO per leggere il file, 'q' o CTRL-C per uscire. `less` e' piu' intuitivo, si possono usare i tasti freccia;
- non c'e' `UNDELETE`, quindi *pensarci bene* prima di cancellare un file;
- oltre a < > >> del DOS, Linux ha `2>` per redirigere i messaggi di errore (stderr); inoltre, `2>&1` redirige stderr su stdout, mentre `1>&2` redirige stdout su stderr;
- Linux ha un altro wildcard: la coppia `[]`. Uso: `[abc]*` prende i file che cominciano con a, b, c; `*[I-N,1,2,3]` prende i file che finiscono con I, J, K, L, M, N, 1, 2, 3;
- non c'e' un `RENAME` alla DOS; cioe', `mv` *.xxx *.yyy non funziona. Potete usare questo script; dettagli alla sezione 7.1 (Shell Scripts).

```
#!/bin/sh
# ren: rename multiple files according to several rules

if [ $# -lt 3 ] ; then
  echo "usage: ren \"pattern\" \"replacement\" files..."
  exit 1
fi

OLD=$1 ; NEW=$2 ; shift ; shift

for file in $*
do
  new=echo ${file} | sed s/${OLD}/${NEW}/g
  mv ${file} $new
done
```

Attenti pero': non e' equivalente al `REN` del DOS, poiché usa le "regular expressions" che non conoscete ancora. In breve, se volete cambiare le estensioni dei file, scrivete: `ren "htm$" "html" *htm`. Non dimenticate il `$`.

- usare `cp -i` e `mv -i` per essere avvisati se un file sta per essere sovrascritto.

2.5 Programmi: Multitasking e Sessioni

Per far partire un programma, si scrive il suo nome come col DOS. Se la directory (Sezione 3 (Directories)) dove il programma risiede e' nel PATH (Sezione 6.1 (Files di inizializzazione)), il programma parte. Eccezione: sotto Linux, un programma che sta nella directory corrente non parte se la directory non e' inclusa nel PATH. Scappatoia: se `prog` e' il programma, scrivere `./prog`.

Questa e' una tipica linea di comando:

```
$ command -s1 -s2 ... -sn par1 par2 ... parn < input > output
```

dove `-s1`, ..., `-sn` sono gli switch del programma, `par1`, ..., `parn` sono gli argomenti del programma. Si possono dare piu' comandi sulla stessa linea:

```
$ command1 ; command2 ; ... ; commandn
```

Tutto qui per quanto riguarda i programmi, ma e' facile fare dei passi avanti. Uno dei vantaggi di Linux e' il multitasking: puo' far girare piu' programmi (d'ora in poi, processi) allo stesso tempo. Si possono lanciare programmi in background e continuare a lavorare. Inoltre, Linux mette a disposizione piu' sessioni di lavoro contemporanee: e' come avere tanti computer allo stesso tempo!

- Per passare di sessione in sessione (1..6) nelle console virtuali:

  ```
  $ ALT-F1 ... ALT-F6
  ```

- Per far partire un'altra sessione nella stessa console virtuale senza lasciare quella corrente:

  ```
  $ su - <loginname>
  ```

  Esempio:

```
$ su - root
```

Questo e' utile, tra l'altro, per usare i floppy (Sezione 4 (Floppies)): normalmente, solo root lo puo' fare.

- Per chiudere una sessione:

```
$ exit
```

Se ci sono dei job sospesi (vedi piu' avanti) si viene avvisati.

- Per lanciare normalmente un processo in primo piano:

```
$ progname [-switches] [parameters] [< input] [> output]
```

- Per lanciare un processo in background, aggiungere una 'e commerciale' '&' alla fine della linea di comando:

```
$ progname [-switches] [parameters] [< input] [> output] &
[1] 123
```

la shell identifica i processi dando loro un numero (es. [1]; vedi sotto) e un PID (123 nel nostro esempio).

- Per vedere quanti processi ci sono:

```
$ ps -a
```

Questo comando da' una lista dei processi attualmente in esecuzione.

- Per uccidere (terminare) un processo:

```
$ kill <PID>
```

Potreste dover uccidere un processo se non sapete come uscirne normalmente... ;-). A volte, un processo si puo' uccidere solo con:

```
$ kill -SIGKILL <PID>
```

Oltre a questo, la shell consente di fermare o sospendere un processo, mandare un processo in background, e portare un processo dal background in primo piano. In questo contesto, i processi sono chiamati 'job'.

- Per vedere quanti job ci sono:

```
$ jobs
```

qui i job sono identificati dal loro numero, non dal PID.

- Per fermare un job che gira in primo piano (non sempre funziona):

```
$ CTRL-C
```

- Per sospendere un processo che gira in primo piano (idem):

```
$ CTRL-Z
```

- Per mandare in background un processo sospeso:

```
$ bg <job>
```

- Per portare un job in primo piano:

```
$ fg <job>
```

- Per uccidere un job:

```
$ kill <%job>
```

dove <job> puo' essere 1, 2, 3, ... Usando questi comandi si puo' formattare un disco, zippare dei files, compilare un programma e decompattare un archivio tutto allo stesso tempo, e ancora avere il prompt a disposizione. Provate a farlo col DOS! E provate con Windows, giusto per vedere la differenza in performance.

2.6 Eseguire programmi su computer remoti

Per eseguire un programma su un computer remoto il cui indirizzo IP e' `remote.bigone.edu`, si fa:

```
$ telnet remote.bigone.edu
```

Dopo il login, si fa partire il programma. Ovviamente, bisogna avere uno shell account sul computer remoto.

Se avete X11, si possono far girare anche applicazioni X sul computer remoto, e queste verranno visualizzate sul vostro schermo. Siano `remote.bigone.edu` il computer remoto e `local.linux.box` il vostro PC. Per far girare da `local.linux.box` un programma X che sta su `remote.bigone.edu`, si fa:

- far partire X11 ed un `xterm` o equivalente emulatore di terminale, poi digitare:

```
$ xhost +remote.bigone.edu
$ telnet remote.bigone.edu
```

- dopo il login, digitare:

```
remote:$ DISPLAY=local.linux.box:0.0
remote:$ progname &
```

(invece di `DISPLAY...`, potreste dover scrivere: `setenv DISPLAY local.linux.box:0.0`. Dipende dalla shell remota.)

Et voila! Ora `progname` parte su `remote.bigone.edu` e viene visualizzato sulla vostra macchina. È pero' meglio non provarci tramite modem, perché e' assolutamente troppo lento.

3 Usare le directory

3.1 Directory: nozioni preliminari

Abbiamo visto le differenze tra i file sotto DOS e sotto Linux. Per quanto riguarda le directory, sotto DOS la directory principale e' \, sotto Linux e' /. In maniera analoga, le directory sono separate da \ sotto DOS e da / sotto Linux. Esempio:

```
DOS:    C:\PAPERS\GEOLOGY\MID_EOC.TEX
Linux:  /home/guido/papers/geology/mid_eocene.tex
```

Come al solito, `..` e' la directory genitrice, `.` e' la directory corrente. Ricordate che il sistema non vi lascia fare `cd`, `rd` o `md` ovunque si vuole. Ogni utente "risiede" in una sua directory chiamata 'home', che viene assegnata dall'amministratore di sistema. Per esempio, sul mio PC la mia home directory e' `/home/guido`.

3.2 Permessi delle directory

Anche le directory hanno i permessi. Quanto visto in Sezione 2.3 (Permessi) vale anche per le directory (user, group, e other). Per una directory, `rx` significa che potete fare `cd` nella directory, e `w` significa che potete cancellare i file nella directory, o la directory stessa. Per esempio, per impedire ad altri utenti di curiosare in `/home/guido/text`:

```
$ chmod o-rwx /home/guido/text
```

3.3 Tradurre i comandi dal DOS a Linux

```
DIR:            ls, find, du
CD:             cd, pwd
MD:             mkdir
RD:             rmdir
DELTREE:        rm -R
MOVE:           mv
```

- ESEMPI -

```
DOS                                     Linux
---------------------------------------------------------------------

C:\GUIDO>DIR                            $ ls
C:\GUIDO>DIR FILE.TXT                   $ ls file.txt
C:\GUIDO>DIR *.H *.C                    $ ls *.h *.c
C:\GUIDO>DIR/P                          $ ls | more
C:\GUIDO>DIR/A                          $ ls -l
C:\GUIDO>DIR *.TMP /S                   $ find / -name "*.tmp"
C:\GUIDO>CD                             $ pwd
        n/a - vedi nota                 $ cd
        idem                            $ cd ~
        idem                            $ cd ~/temp
C:\GUIDO>CD \OTHER                      $ cd /other
C:\GUIDO>CD ..\TEMP\TRASH               $ cd ../temp/trash
C:\GUIDO>MD NEWPROGS                    $ mkdir newprogs
C:\GUIDO>MOVE PROG ..                   $ mv prog ..
C:\GUIDO>MD \PROGS\TURBO                $ mkdir /progs/turbo
C:\GUIDO>DELTREE TEMP\TRASH             $ rm -R temp/trash
C:\GUIDO>RD NEWPROGS                    $ rmdir newprogs
C:\GUIDO>RD \PROGS\TURBO                $ rmdir /progs/turbo
```

Note:

1. quando usate `rmdir`, la directory da cancellare deve essere vuota. Per cancellare una directory e tutto il suo contenuto, usate `rm -R` (a vostro rischio e pericolo).

2. il carattere '~' e' una scorciatoia per il nome della home directory. I commandi `cd` o `cd ~` portano nella home directory ovunque voi siate; il comando `cd ~/tmp` vi porta in `/home/la_vostra_home/tmp`.
3. `cd -` "annulla" l'ultimo `cd`.

4 Floppy, hard disk, e cosi' via

4.1 Gestire i dispositivi

Forse non ci avete mai pensato, ma il comando DOS `FORMAT A:` fa molte piu' cose di quanto sembri. Infatti, quando date il comando `FORMAT` quello 1) formatta fisicamente il disco; 2) crea la directory A:\ (crea un filesystem); 3) rende il disco accessibile all'utente (= fa il cosiddetto "mount").

Questi tre passaggi si fanno separatamente con Linux. Si possono usare floppy formattati da DOS, ma ci sono altri formati che di solito e' meglio usare (il filesystem DOS non consente di usare i nomi lunghi per i file.)

Ecco come si prepara un floppy (bisogna essere root):

- per formattare un floppy standard da 1.44: (A:):

  ```
  # fdformat /dev/fd0H1440
  ```

- per creare un filesystem:

  ```
  # mkfs -t msdos -c /dev/fd0H1440
  ```

  Per creare un filesystem MS-DOS, usate `msdos` invece di `ext2`. Prima di usare il disco, bisogna "montarlo".

- per montare il floppy:

  ```
  # mount -t ext2 /dev/fd0 /mnt
  ```

  oppure

  ```
  # mount -t msdos /dev/fd0 /mnt
  ```

  Ora si puo' accedere ai file del floppy. Quando si ha finito, prima di estrarre il floppy *bisogna* "smontarlo".

- per smontare il floppy:

  ```
  # umount /mnt
  ```

Ora potete estrarre il disco. Ovviamente, eseguire `fdformat` e `mkfs` solo per floppy nuovi, non per quelli gia' preparati. Se volete usare il floppy B:, basta sostituire `fd1H1440` e `fd1` al posto di `fd0H1440` e `fd0` negli esempi visti sopra.

Tutto quello che facevate con A: o B: si fa ora usando `/mnt`. Esempi:

```
DOS                                     Linux
---------------------------------------------------------------------

C:\GUIDO>DIR A:                         $ ls /mnt
C:\GUIDO>COPY A:*.*                     $ cp /mnt/* .
```

```
C:\GUIDO>COPY *.ZIP A:              $ cp *.zip /mnt
C:\GUIDO>A:                         $ cd /mnt
A:>_                                /mnt/$ _
```

Se non vi piace questa faccenda di montare/smontare i dischi, usate la suite `mtools`: si tratta di un insieme di comandi equivalenti a quelli del DOS, ma che iniziano con la 'm': `mformat`, `mdir`, `mdel` e cosi' via. Funzionano anche con i nomi lunghi, ma si perdono i permessi dei file. Usate questi comandi come usereste quelli DOS, e siete a posto.

Inutile dire che quanto visto per i floppy vale anche per altri dispositivi; per esempio, si possono montare un altro disco fisso o un lettore di CD-ROM. Ecco come si monta il CD-ROM:

```
# mount -t iso9660 /dev/cdrom /mnt
```

Questa era la maniera "ufficiale" di montare i dischi, ma c'e' un trucchetto. Dover essere root per usare un floppy e' una scocciatura, quindi per consentire ad ogni utente di accedervi si puo' fare cosi':

- come root, date questi comandi:

  ```
  ~# mkdir /mnt/a: ; mkdir /mnt/a ; mkdir /mnt/cdrom
  ~# chmod 777 /mnt/a* /mnt/cd*
  ~# # assicuratevi che il device del CD-ROM e' quello giusto
  ~# chmod 666 /dev/hdb ; chmod 666 /dev/fd*
  ```

- aggiungere in `/etc/fstab` le linee seguenti:

  ```
  /dev/cdrom      /mnt/cdrom  iso9660 ro,user,noauto          0       0
  /dev/fd0        /mnt/a:     msdos   user,noauto             0       0
  /dev/fd0        /mnt/a      ext2    user,noauto             0       0
  ```

Ora per montare un floppy DOS, uno in formato ext2 e un CD-ROM si fa semplicemente:

```
$ mount /mnt/a:
$ mount /mnt/a
$ mount /mnt/cdrom
```

`/mnt/a`, `/mnt/a:` e `/mnt/cdrom` sono accessibili da tutti. Ricordate che consentire a tutti di montare dischi in questo modo e' un grosso problema di sicurezza, se la cosa vi puo' interessare.

4.2 Fare il backup

Ora che sapete come gestire i dispositivi, due righe per vedere come si fa il backup. Ci sono molti programmi per questo scopo, ma il minimo necessario che si puo' fare per un backup multivolume su floppy e' quanto segue (diventate root):

```
# tar -M -cvf /dev/fd0H1440 /dir_to_backup
```

Assicuratevi di avere un floppy formattato nel drive, e altri pronti a disposizione. Per fare il restore, inserite il primo floppy nel drive e immettete:

```
# tar -M -xpvf /dev/fd0H1440
```

5 E Windows?

L'"equivalente" di Windows e' l'ambiente grafico X11. A differenza di Windows o del Mac, X11 non e' stato progettato per la facilita' d'uso o per risultare attraente, ma per fornire capacita' grafiche alle workstation UNIX. Ecco le differenze principali:

- Windows ha sempre lo stesso look and feel, X11 no: e' di gran lunga piu' configurabile. L'aspetto generale di X11 e' dato da un componente importantissimo chiamato "window manager"; ce ne sono molti tra cui scegliere. I piu' comuni sono `fvwm`, semplice ma gradevole ed efficiente in termini di memoria, `fvwm2-95` e `The Next Level` che danno a X11 un aspetto simile a Windows 95, piu' molti altri. Alcuni sono davvero bellissimi;
- il window manager puo' essere configurato in modo tale che una finestra si comporta come quelle di Windows: cliccate su di essa per portarla in primo piano. In alternativa, si puo' fare in modo che una finestra sia in primo piano quando il puntatore del mouse e' sopra di essa. Questa caratteristica ("focus") e molte altre si modificano adattando uno o piu' file di configurazione. Leggete la documentazione del vostro window manager;
- i programmi di X11 sono scritti usando speciali librerie ("widget set"); ce ne sono svariate, e quindi i programmi possono avere un aspetto diverso. I piu' elementari usano i widget Athena (aspetto 2–D; `xdvi`, `xman`, `xcalc`), altri usano Motif (`netscape`), altri ancora usano Tcl/Tk, XForms, Qt ed altri ancora. Alcuni di questi widget set danno ai programmi un aspetto simile a quello dei programmi Windows;
- questo riguardava il "look" dei programmi, ma il "feel"? Purtroppo, tutti i programmi si comportano in modo diverso. Per esempio, se selezionate una linea di testo e premete BACKSPACE vi aspettereste che la linea scomparisse, vero? Questo non funziona con i programmi Athena, ma funziona con quelli Motif, Qt e Tcl/Tk;
- barre di scorrimento, ridimensionamento, iconizzazione: anche queste cose dipendono dal window manager e dal widget set. Ci sarebbero troppe cose da dire, quindi ve ne indichero' solo una, poco intuitiva. Quando usate le applicazioni Athena, le barre di scorrimento si spostano usando il tasto centrale del mouse, oppure i tasti destro e sinistro insieme;
- i programmi non hanno un'icona per default, ma ne possono avere tante. Dipende dal window manager. Il desktop e' chiamato "root window" e se ne modifica l'aspetto con programmi come `xsetroot` o `xloadimage`;
- la clipboard puo' contenere solo testo e si comporta in modo strano. Quando del testo viene selezionato, e' anche automaticamente copiato nella clipboard: spostatevi in un altro punto e premete il tasto centrale per copiarlo nella nuova locazione. C'e' un programma, `xclipboard`, che fornisce buffer multipli per la clipboard;
- drag and drop e' un'opzione ed e' supportato solo da alcuni programmi.

Per risparmiare memoria, e' meglio usare applicazioni che usano gli stessi widget set, ma e' difficile da fare in pratica. C'e' un progetto chiamato K Desktop Environment che vuole rendere X11 coerente nel look and feel come lo e' Windows; attualmente e' solo in beta ma, credetemi, e' meraviglioso. Rendera' l'interfaccia di Windows una cosa di cui vergognarsi. Puntate il vostro browser su *http://www.kde.org* .

6 Configurare il sistema

6.1 File di inizializzazione

Due file importanti sotto DOS sono `AUTOEXEC.BAT` e `CONFIG.SYS`, e vengono usati al momento del boot per inizializzare il sistema, settare le variabili d'ambiente come PATH e FILES, e magari lanciare un programma o batch file. Sotto Linux ci sono tanti file di inizializzazione, e con molti di questi e' meglio non pasticciare finché non si sa esattamente cosa si sta facendo. Vi diro' comunque quali sono i piu' importanti:

```
FILES                                   NOTES

/etc/inittab                            non toccare!
/etc/rc.d/*                             idem
```

Se tutto quello che vi serve e' settare il PATH o qualche altra variabile, o se volete cambiare i messaggi di login o lanciare un programma subito dopo il login, date un'occhiata ai seguenti:

```
FILES                                   NOTES

/etc/issue                              setta i messaggi pre-login
/etc/motd                               setta i messaggi post-login
/etc/profile                            setta PATH e altre variabili, etc.
/etc/bashrc                             setta alias e funzioni globali
/home/your_home/.bashrc                 setta i vostri alias e funzioni
/home/your_home/.bash_profile  oppure
/home/your_home/.profile                setta il vostro environment, etc.
```

Se l'ultimo file esiste (notare che e' un file nascosto), verra' letto dopo il login e i suoi comandi vengono eseguiti.

Esempio: date un occhio a questo `.profile`:

```
# Io sono un commento
echo Environment:
printenv | less   # equivalente del comando SET sotto DOS
alias d='ls -l'   # facile capire cos'e' un alias
alias up='cd ..'
echo "Ti ricordo che il path e' " $PATH
echo "Oggi e' `date`"  # usa l'output del comando 'date'
echo "Buongiorno " $LOGNAME
# Questa e' una "funzione della shell":
ctgz() # Lista i contenuti di un archivio .tar.gz.
{
  for file in $*
  do
    gzip -dc ${file} | tar tf -
  done
}
# fine di .profile
```

`$PATH` e `$LOGNAME`, avete indovinato, sono variabili d'ambiente. Ce ne sono molte altre con cui giochicchiare; LMP di programmi come `less` o `bash`.

6.2 File di inizializzazione dei programmi

Sotto Linux, praticamente tutto puo' essere personalizzato. Molti programmi hanno uno o piu' file di inizializzazione che potete modificare, spesso sotto forma di `.nome_del_programmarc` nella vostra home. I primi che vorrete modificare sono:

- `.inputrc`: usato da `bash` per definire i tasti.
- `.xinitrc`: usato da `startx` per initializzare X Window System.
- `.fvwmrc`: usato dal window manager `fvwm`. Un esempio e' in: `/usr/lib/X11/fvwm/system.fvwmrc`
- `.Xdefault`: usato da `rxvt`, un emulatore di terminale per X, e altri programmi.

Per tutti questi e gli altri che prima o poi incontrerete, LMP.

7 Un po' di programmazione

7.1 Shell script: super file .BAT

Se usavate i file .BAT per creare diminutivi di lunghe linee di comando (io lo facevo spesso), questo si puo' ottenere inserendo degli alias in `profile` o `.profile`. Ma se i vostri .BAT erano piu' complessi, vi piacera' il linguaggio di scripting della shell: e' potente come il QBasic, se non di piu'. Ha variabili, strutture come while, for, case, if... then... else, e molte altre caratteristiche: puo' essere una buona alternativa ad un "vero" linguaggio di programmazione.

Per scrivere un file script—l'equivalente di un .BAT file sotto DOS—non si fa altro che scrivere un file ASCII contenente le istruzioni, lo si salva e poi lo si rende eseguibile col comando `chgmod +x <scriptfile>`. Per eseguirlo si scrive il suo nome.

Attenzione pero'. L'editor di sistema si chiama `vi`, e ho visto che molti nuovi utenti lo trovano molto difficile da usare. Non spieghero' come usarlo perché non mi piace e non lo uso; consultate "Linux installation..." di Matt Welsh, a pag. 109. (Fareste meglio a procurarvi un altro editor come `joe` o `emacs` per X.) Basti dire che:

- per inserire del testo, premete 'i' e poi il testo;
- per uscire da `vi` senza salvare, premete ESC e poi :q!
- per salvare e uscire, premete ESC e poi :wq

Scrivere script sotto `bash` e' un argomento cosi' vasto che richiederebbe un libro per conto suo, e non daro' spiegazioni ma solo un esempio piuttosto completo di shell script, dal quale potrete intuire le regole di base:

```
#!/bin/sh
# esempio.sh
# Io sono un commento
# non cambiate la prima linea, deve restare cosi' com'e'
echo "Questa macchina e': `uname -a`" # usa l'output del comando
echo "Il mio nome e' $0" # variabile interna
echo "Mi hai dato i seguenti $# argomenti: " $*
echo "Il primo argomento e': " $1
```

```
echo -n "Come ti chiami? " ; read nome
echo guarda la differenza: "ciao $nome" # meccanismo di quoting con "
echo guarda la differenza: 'ciao $nome' # meccanismo di quoting con '
DIRS=0 ; FILES=0
for file in `ls .` ; do
  if [ -d ${file} ] ; then # se file e' una directory
    DIRS=`expr $DIRS + 1`  # DIRS = DIRS + 1
  elif [ -f ${file} ] ; then
    FILES=`expr $FILES + 1`
  fi
  case ${file} in
    *.gif|*jpg) echo "${file}: file grafico" ;;
    *.txt|*.tex) echo "${file}: file di testo" ;;
    *.c|*.f|*.for) echo "${file}: file sorgente" ;;
    *) echo "${file}: file generico" ;;
  esac
done
echo "ci sono ${DIRS} directories e ${FILES} files"
ls | grep "ZxY--!!!WKW"
if [ $? != 0 ] ; then # exit code dell'ultimo comando
  echo "non ho trovato ZxY--!!!WKW"
fi
echo "basta cosi'... scrivi 'man bash' se vuoi altre informazioni."
```

7.2 Programmare in C

Sotto Unix, il linguaggio di programmazione per eccellenza e' il C, vi piaccia o no. Ci sono anche molti altri linguaggi a disposizione (FORTRAN, Pascal, Lisp, Basic, Perl, awk...).

Dato per scontato che conosciate il C, ecco qui un paio di linee guida per quelli di voi che sono stati viziati dal Turbo C++ o analogo compilatore. Il compilatore C di Linux si chiama `gcc` e non ha tutte quelle cose che di solito accompagnano un compilatore per DOS: niente IDE, aiuto in linea, debugger integrato e cosi' via. È solo un compilatore a linea di comando, molto potente ed efficiente. Per compilare il classico `hello.c` si fa cosi':

```
$ gcc hello.c
```

che produce di default un eseguibile chiamato `a.out`. Per dargli un nome diverso:

```
$ gcc -o ciao hello.c
```

Per linkare una libreria al programma, si aggiunge lo switch -l<libname>. Per esempio, per linkare la libreria matematica:

```
$ gcc -o mathprog mathprog.c -lm
```

(Lo switch `-l<libname>` fa linkare a `gcc` la libreria `/usr/lib/lib<libname>.a`; quindi `-lm` linka `/usr/lib/libm.a`).

Finora, tutto bene. Ma se il vostro programma e' composto da molti file sorgenti, vi servira' l'utility `make`. Supponiamo che avete scritto un parser per espressioni: il sorgente si chiama `parser.c` e #include due file header, `parser.h` e `xy.h`. Volete usare le routine di `parser.c` in un programma, diciamo `calc.c`, che a sua volta #include `parser.h`. Che casino! Cosa bisogna fare per compilare `calc.c`?

Dovrete scrivere un cosiddetto `makefile`, che insegna al compilatore quali sono le dipendenze tra sorgenti e files oggetto. Nel nostro caso:

```
# Questo e' makefile, usato per compilare calc.c
# Premere il tasto <TAB> dove indicato

calc: calc.o parser.o
<TAB>gcc -o calc calc.o parser.o -lm
# calc dipende da due files oggetto: calc.o e parser.o

calc.o: calc.c parser.h
<TAB>gcc -c calc.c
# calc.o dipende da due files sorgenti

parser.o:  parser.c parser.h xy.h
<TAB>gcc -c parser.c
# parser.o dipende da tre files sorgenti

# end of makefile.
```

Salvate questo file come `makefile` e scrivete `make` per compilare il programma; in alternativa, salvatelo come `calc.mak` e scrivete `make -f calc.mak`. Ovviamente, LMP. Potete ottenere aiuto sulle funzioni del C, che sono illustrate da pagine man, sezione 3; per esempio,

```
$ man 3 printf
```

Ci sono tantissime librerie disponibili; tra le prime che vorrete usare ci sono `ncurses`, per gestire effetti in modo testo, e `svgalib`, per fare grafica. Se vi sentite abbastanza coraggiosi da affrontare la programmazione sotto X, procuratevi `XForms` (*bloch.phys.uwm.edu/pub/xforms*) e/o una delle tante librerie che rendono facile la programmazione sotto X. Date un'occhiata a *http://www.xnet.com/~blatura/linapp6.html* .

Molti editor possono fungere da IDE; `emacs` e `jed`, ad esempio, hanno l'evidenziazione della sintassi, l'indent automatico e altre cose. Oppure, prendete il programma `rhide` da *sunsite.unc.edu:/pub/Linux/devel/debuggers/* . E' un clone della IDE Borland, e probabilmente vi piacera'.

8 Il rimanente 1%

8.1 Usare `tar` & `gzip`

Sotto Unix ci sono alcune applicazioni usatissime per archiviare e comprimere i file. `tar` e' usato per fare archivi—e' come `PKZIP` ma non comprime, archivia soltanto. Per fare un nuovo archivio:

```
$ tar -cvf <archive_name.tar> <file> [file...]
```

Per estrarre files da un arhivio:

```
$ tar -xpvf <archive_name.tar> [file...]
```

Per listare il contenuto di un archivio:

```
$ tar -tf <archive_name.tar> | less
```

I file si comprimono con `compress`, che e' obsoleto e non dovrebbe essere piu' usato, o con `gzip`:

```
$ compress <file>
$ gzip <file>
```

che crea un file file compresso con estensione .Z (`compress`) o .gz (`gzip`). Questi programmi comprimono solo un file alla volta. Per decomprimere, scrivete

```
$ compress -d <file.Z>
$ gzip -d <file.gz>
```

LMP.

Ci sono anche `unarj`, `zip` e `unzip` (PK??ZIP compatibile). I files con estensione `.tar.gz` o `.tgz` (archivi fatti con `tar` e compressi con `gzip`) sono comuni nel mondo Unix come i files .ZIP sotto DOS. Per listare i contenuti di un file `.tar.gz`:

```
$ gzip -dc <file.tar.gz> | tar tf - | less
```

8.2 Installare le applicazioni

Prima di tutto: installare nuove applicazioni e' compito di root. Alcune applicazioni Linux sono distribuite come archivi `.tar.gz` o `.tgz`, fatti in modo da poter essere scompattati dalla directory / col seguente comando:

```
# gzip -dc <file.tar.gz> | tar xvf -
```

oppure, in modo equivalente,

```
$ tar -zxf <file.tar.gz>
```

I file vengono decompressi nella directory giusta, che viene creata sul momento. Gli utenti della distribuzione Slackware hanno il programmino `pkgtool`; un altro e' `rpm`, disponibile per tutte le distribuzioni grazie a Red Hat.

Altri package non possono essere installati da /; tipicamente, l'archivio contiene una directory chiamata `nome_programma/` e tanti files e/o sottodirectories sotto `nome_programma/`. Una regola e' quella di installare questi programmi da `/usr/local`. Inoltre, altri programmi sono distribuiti come sorgenti in C o C++ che vanno compilati per fare gli eseguibili. In molti casi, basta dare `make`; ovviamente vi servira' il compilatore `gcc`.

8.3 Trucchi indispensabili

Command completion: premere <TAB> mentre si scrive un comando al prompt completa la linea di comando. Esempio: dovete scrivere `gcc nome_file_molto_lungo.c`; scrivendo `gcc nome<TAB>` e' sufficiente. (Se avete altri file che cominciano con gli stessi caratteri, scrivete altre lettere per risolvere l'ambiguita').

Backscrolling: premendo SHIFT + PAG UP (tasto grigio) consente di fare lo scroll all'indietro dello schermo, a seconda di quanta memoria video avete;

Resettare lo schermo: puo' capitare di fare `more` o `cat` di un file binario, e come conseguenza lo schermo potrebbe riempirsi di schifezze. Per rimettere a posto, battere alla cieca `reset` o questa sequenza di caratteri: `echo CTRL-V ESC c RETURN`;

Incollare il testo: per la console, vedete sotto; in X, fate click e trascinate per selezionare il testo in una finestra `xterm`, poi premete il tasto di mezzo (o i due bottoni insieme se non avete tre tasti) per incollare il testo selezionato altrove. C'e' anche il programma `xclipboard` (purtroppo solo per il testo); non fatevi confondere dal suo lentissimo tempo di risposta;

Usare il mouse: installate `gpm`, un mouse driver per la console. Fate click e trascinate per selezionare il testo, poi fate click col tasto destro per incollare il testo selezionato. Funziona anche tra diverse VC.

Messaggi dal kernel: date un occhio a `/var/adm/messages` o `/var/log/messages` come root per vedere i messaggi del kernel, compresi i messaggi in fase di boot. Anche il comando `dmesg` e' molto utile.

8.4 Programmi e comandi utili

Ovviamente, questa lista riflette i miei gusti e le mie necessita' personali. Prima di tutto, dove trovarli: sapete tutti come usare la rete, `archie` ed `ftp`, quindi vi daro' solo gli indirizzi piu' importanti che riguardano Linux:*sunsite.unc.edu* , *tsx-11.mit.edu* , e *nic.funet.fi* . Usate il vostro mirror piu' vicino.

- `at` serve per eseguire comandi e programmi ad un'ora o data specifica;
- `awk` e' un linguaggio di programmazione, semplice ma potente, per manipolare file di dati (e non solo). Per esempio, se avete un file di dati a piu' campi chiamato `data.dat`,

  ```
  $ awk '$2 ~ "abc" {print $1, "\t", $4}' data.dat
  ```

  scrive i campi 1 e 4 di ogni linea in `data.dat` il cui secondo campo contiene “abc”.
- `delete-undelete` fanno quello che il loro nome suggerisce;
- `df` da' informazioni sui dischi montati;
- `dosemu` consente di far girare molte (anche se non tutte) applicazioni DOS, incluso Windows 3.x se ci smanettate un bel po';
- `file <filename>` dice che cos'e' `filename` (file ASCII, eseguibile, archivio, etc.);
- `find` (vedi anche Sezione 3 (Directories)) e' uno dei comandi piu' utili e potenti. Si usa per trovare file che rispondono a certi criteri ed eseguire azioni su si essi. Uso generale di `find`:

  ```
  $ find <directory> <espressione>
  ```

  dove <espressione> include criteri di ricerca ed azioni da eseguire. Esempi:

  ```
  $ find . -type l -exec ls -l {} \;
  ```

  trova i file che sono link simbolici e mostra a cosa puntano.

```
$ find / -name "*.old" -ok rm {} \;
```

trova i files che corrispondono al pattern e li cancella, chiedendo prima il permesso di farlo.

```
$ find . -perm +111
```

trova i file i cui permessi corrispondono con 111 (eseguibile).

```
$ find . -user root
```

trova i files che appartengono a root. Ci sono molte altre possibilita', LMP.

- `gnuplot` e' un bel programma per il plotting scientifico;
- `grep` trova pattern in file di testo. Per esempio,

  ```
  $ grep -l "geology" *.tex
  ```

  lista tutti i files *.tex che contengono la parola "geology". La variante `zgrep` agisce su file gzippati. LMP;
- `joe` e' un buon editor. Lanciandolo come `jstar` si ottengono le stesse combinazioni di tasti di WordStar e dei suoi discendenti, compresi l'editor del DOS e quello dei linguaggi Borland;
- `less` e' probabilmente il migliore visualizzatore di file di testo, e se configurato consente di visualizzare archivi gzip, tar e zip;
- `lpr` <file> stampa un file in background. Per controllare lo stato della coda di stampa, usate `lpq`; per cancellare un file dalla coda di stampa, usate `lprm`;
- `mc` e' un bellissimo file manager;
- `pine` e' un buon programma per la posta elettronica;
- `script <script_file>` copia su `script_file` tutto quello che appare sullo schermo fino a quando non date il comando `exit`. Utile per il debugging;
- `sudo` permette di eseguire alcuni dei compiti solitamente concessi solo a root (es. formattare e montare dischi; LMP);
- `tcx` comprime files eseguibili mantenendoli eseguibili;
- `uname -a` da' informazioni sul sistema;
- `zcat` e `zless` sono utili per visualizzare file gzippati senza decomprimerli. Per esempio:

  ```
  $ zless textfile.gz
  $ zcat textfile.gz | lpr
  ```
- I seguenti comandi risultano spesso utili: `bc`, `cal`, `chsh`, `cmp`, `cut`, `fmt`, `head`, `hexdump`, `nl`, `passwd`, `printf`, `sort`, `split`, `strings`, `tac`, `tail`, `tee`, `touch`, `uniq`, `w`, `wall`, `wc`, `whereis`, `write`, `xargs`, `znew`. LMP.

8.5 Estensioni di file e programmi collegati

Potrete incontrare tantissime estensioni di file. A parte le piu' esotiche (ad es. fonts, etc.), ecco una lista:

- `1 ... .8`: man page. Procuratevi `man`.
- `arj`: archivio fatto con `arj`. `unarj` per estrarre i file.
- `dvi`: output file prodotto da TeX (vedi sotto). `xdvi` per visualizzare; `dvips` per convertire in un file .ps (postscript)
- `gif`: file grafico. Procuratevi `seejpeg` o `xpaint`.
- `gz`: file compresso con `gzip`.
- `info`: file info (una specie di alternativa alle man pages.). Procuratevi `info`.
- `jpg, jpeg`: file grafico. Procuratevi `seejpeg`.
- `lsm`: Linux Software Map file. È un file ASCII contenente la descrizione di un package.
- `ps`: file postscript. Per visualizzare o stampare, usare `gs` e, opzionalmente, `ghostview`.
- `tgz, tar.gz`: archivio fatto con `tar` e poi compresso con `gzip`.
- `tex`: file di testo da comporre con TeX, un potente programma di impaginazione. Procurarsi `tex`, disponibile in molte distribuzioni; attenti alla distribuzione NTeX, che ha dei font corrotti ed e' parte di Slackware fino alla versione 96.
- `texi`: file texinfo (vedi `.info`). Procuratevi `texinfo`.
- `xbm, xpm, xwd`: file grafici. Procuratevi `xpaint`.
- `Z`: file compresso con `compress`.
- `zip`: archivio fatto con `zip`. Procuratevi `zip` e `unzip`.

9 La fine, per ora

Congratulazioni! Ora avete un po' di conoscenza di Unix e siete pronti per iniziare a lavorare. Ricordate che la vostra conoscenza del sistema e' ancora limitata, e che dovreste fare pratica per usare Linux senza problemi. Ma se tutto quello che dovete fare e' prendere un po' di programmi e darci sotto, sono sicuro che quanto ho incluso e' sufficiente.

Sono certo che usare Linux vi piacera' e continuerete ad imparare cose nuove—lo fanno tutti. Scommetto, inoltre, che non ritornerete al DOS! Spero di essere stato chiaro e di aver reso un buon servizio ai miei 3 o 4 lettori.

9.1 Copyright

impose additional restrictions on its distribution. Exceptions to these rules may be granted under certain conditions; please contact the Linux HOWTO coordinator at the address given below.

In short, we wish to promote dissemination of this information through as many channels as possible. However, we do wish to retain copyright on the HOWTO documents, and would like to be notified of any plans to redistribute the HOWTOs.

If you have questions, please contact Greg Hankins, the Linux HOWTO coordinator, at gregh@sunsite.unc.edu via email.

9.2 Disclaimer

"Dal DOS a Linux HOWTO" e' stato scritto da Guido Gonzato, *guido@ibogfs.cineca.it* . Ringraziamenti vanno a Matt Welsh, autore di "Linux Installation e Getting Started", a Ian Jackson, autore di "Linux frequently asked questions with answers", a Giuseppe Zanetti, autore di "Linux", a tutti quelli che mi hanno mandato mail di suggerimenti, e specialmente a Linus Torvalds e GNU che ci hanno dato Linux.

Questo documento viene fornito "as is". Mi sono sforzato di scriverlo con la massima accuratezza, ma usate le informazioni qui contenute a vostro rischio. Non saro' responsabile in alcun caso di danni provocati da questo documento.

Ogni forma di feedback e' benvenuta. Per richieste, suggerimenti, flames etc. contattatemi pure.

Divertitevi con Linux e godetevi la vita,

Guido =8-)