

Sistemi operativi

Andrea Passerini
passerini@disi.unitn.it

Informatica

Sistema UNIX

- Il sistema UNIX è stato sviluppato da ricercatori dei laboratori AT&T e Bell Labs negli anni sessanta, ed è stato un passo avanti significativo nel panorama dei sistemi operativi
- E' un sistema multi-utente, multi-tasking, portabile su varie architetture hardware e orientato alla programmazione
- Il sistema si è diffuso notevolmente in ambito accademico, poiché veniva distribuito gratuitamente da AT&T ed era particolarmente adatto a supportare applicazioni scientifiche.
- E' attualmente alla base di numerosi sistemi operativi moderni, quali Mac OS X e GNU/Linux

Sistemi Microsoft Windows

- I sistemi Microsoft Windows hanno avuto larghissima diffusione tra gli utenti domestici e nelle aziende
- Come tutti i moderni sistemi operativi, si basano su un'interfaccia grafica a finestre e gestiscono il multitasking
- Microsoft ha rilasciato una serie di sistemi operativi a partire dalla metà degli anni '90: Windows 95, 98, 2000, XP, Vista
- Inizialmente concepiti per l'uso domestico privo di connessione Internet, tali sistemi sono sempre stati fortemente soggetti a problemi di sicurezza, anche in virtù della loro diffusione che incentivava lo sviluppo di virus ad-hoc.

Sistemi Mac

- I sistemi Mac sono sviluppati dalla Apple specificatamente per computer Macintosh
- Tali sistemi sono caratterizzati da una forte integrazione tra hardware e software, elevate prestazioni come sistemi utente e un'attenzione particolare alla grafica
- Le versioni attuali del sistema operativo sono varianti del sistema Mac OS X (leggi dieci)
- Il kernel del sistema è derivato dal sistema operativo UNIX

Sistema GNU/Linux

- Il sistema GNU/Linux è nato dal connubio del kernel *linux*, sviluppato da Linus Torvald a partire dal 1991, e del progetto *GNU*, lanciato da Richard Stallman nel 1983 per sviluppare un sistema operativo libero
- Il sistema è *libero* nel senso che la sua licenza, denominata *GNU General Public License* (GNU GPL) è concepita appositamente per proteggere la libertà di utilizzo e modifica del sistema
- Il sistema viene fornito comprensivo del *codice sorgente* (a differenza dei sistemi proprietari, quali Microsoft Windows, che forniscono solo gli eseguibili, ossia il codice macchina), ed è permesso modificare il codice e redistribuirlo liberamente, con il vincolo che la versione modificata mantenga la licenza GNU GPL

Sistema GNU/Linux

- La politica del software libero ha permesso la nascita di una grande comunità di sviluppatori, che lavorano per fornire funzionalità , applicativi e modifiche al sistema per migliorarne l'utilizzo
- GNU/Linux è derivato dal sistema UNIX, e ne condivide le caratteristiche di multiutenza, programmabilità , etc.
- Rispetto a UNIX, i moderni sistemi GNU/Linux hanno notevolmente migliorato l'interfaccia grafica e la semplicità di utilizzo, permettendone la diffusione anche tra i non esperti.
- Esistono svariate *distribuzioni* GNU/Linux, che se condividono kernel e (tendenzialmente) filosofia di base, si differenziano per gli applicativi forniti, le interfacce grafiche, etc.

Sistema GNU/Linux: Ubuntu

- *Ubuntu* è una recente distribuzione GNU/Linux che ha acquistato notevole popolarità per la facilità di utilizzo ed il supporto hardware
- Il supporto hardware in particolare implica la notevole capacità di riconoscere e configurare le componenti hardware del sistema, e renderlo quindi totalmente fruibile
- Il supporto hardware è sempre stato un problema complesso per i sistemi liberi, in quanto i produttori di hardware tendono a fornire driver per i sistemi proprietari (in particolar modo Microsoft Windows) ma non per quelli liberi, e spesso non rilasciano le specifiche tecniche sufficienti per permettere alla comunità di sviluppatori di programmare i driver adatti

Installazione

- Ubuntu è liberamente scaricabile dai siti di supporto alla distribuzione. Il sito italiano è :
`http://www.ubuntu-it.org/`
- Ubuntu è disponibile in versione *desktop*, adatta all'uso domestico su portatile o fisso, o *server*, per l'installazione su macchine che forniscono servizi
- L'installazione consiste nello scaricare un file che è un'immagine ISO di un CD di installazione, e nel masterizzare un CD a partire da tale immagine (istruzioni sul sito)
- Una volta creato il CD di installazione, si inserisce nel lettore CD e si riavvia la macchina, e l'installazione guidata parte automaticamente

Autenticazione utente

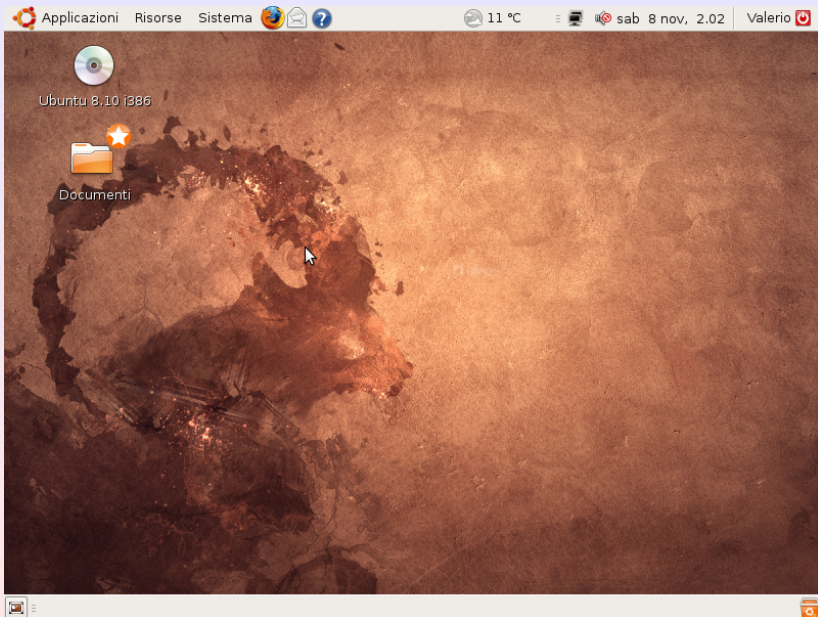
- Come tutti i sistemi GNU/Linux (e UNIX), Ubuntu è un sistema multi-utente.
- In fase di installazione, viene creato un utente che corrisponderà all'utilizzatore standard del sistema.
- L'utente è caratterizzato da un nome e da una password, che vanno forniti all'avvio per poter accedere al sistema

Autenticazione amministratore

- Oltre alla modalità utente, esiste una modalità *amministratore* che ha i permessi necessari ad eseguire operazioni particolari quali aggiornare il sistema, installare software , etc.
- Al momento di eseguire operazioni di amministrazione, il sistema richiede la password di amministratore. Di default, Ubuntu garantisce all'utente standard la possibilità di agire da amministratore, quindi la password da fornire è la password utente stessa.
- La cosa non sarà vera sui sistemi configurati per la multiutenza (quali i computer di laboratorio), in cui l'amministratore ha login e password propri

Gestore di finestre

- L'interfaccia con cui Ubuntu si presenta all'utente è una interfaccia grafica a finestre e menù a tendina, come succede per i sistemi Microsoft Windows e Mac OS X
- Tramite il menù *applicazioni*, è possibile visionare le applicazioni disponibili e lanciarle
- Il menù *risorse* permette di esplorare il contenuto del file system tramite un classico gestore grafico
- Il menù *sistema* fornisce una serie di funzionalità per personalizzare il sistema, configurare nuovo hardware, installare aggiornamenti, etc



File system

- Il programma grafico di esplorazione del file system, eseguibile dal menù *risorse*, permette di navigare nel file system ed aprire file
- Il sistema fornisce una rappresentazione ad icone dei file con una *preview* del loro contenuto
- Inoltre, associa di default ad ogni file un programma per aprirlo, che dipende dal tipo del file stesso (file di testo, immagine, etc)
- Cliccando due volte con il tasto sinistro sull'icona di un file, si apre con l'applicazione di default
- Cliccando con il tasto destro sull'icona, si possono recuperare informazioni sul file o aprirlo con un programma diverso da quello di default.

Aggiornamenti

- Ubuntu si installa con tutta una serie di applicativi di utilità , ben più numerosi di una tipica installazione Microsoft Windows (in quanto si tratta di software libero)
- Spesso è però necessario installare applicativi specifici, che non sono disponibili in una installazione standard
- Ubuntu fornisce un sistema di aggiornamento ed installazione di software estremamente semplice ed efficace tramite il *Synaptic Package Manager* (raggiungibile da `System` → `Administration` → `Synaptic Package Manager`)
- Tale sistema permette di cercare in un grande archivio online di applicazioni disponibili ma non installate, in cui si trova in genere la maggior parte del software necessario
- E' sempre possibile scaricare da Internet ed installare software non disponibile tramite tale interfaccia

Suite Office

- Ubuntu fornisce una versione libera della suite Office, denominata *OpenOffice*, tendenzialmente compatibile con i formati Microsoft Office
- Tale versione fornisce:
 - un software per la realizzazione di testi formattati tipo *Word*
 - un software per la realizzazione di presentazioni stile *Powerpoint*
 - un foglio di calcolo stile *Excel*
 - un software per la realizzazione di semplici basi di dati stile *Access*

Editor di testi

- *Word* serve a realizzare brevi documenti con semplici formattazioni quali titolo sottolineatura, etc (per documenti più complessi quali articoli, libri, etc, altri strumenti quali *latex* sono più adatti), e salva tali file in un formato proprio non puramente testuale
- E' spesso necessario scrivere file di testo puro che non siano formattati come documento, ad esempio nella programmazione (che vedremo approfonditamente)
- A tale scopo si devono usare editor di testi che non includano informazione di formattazione, e salvino in testo puro.
- *emacs* è un utile esempio di editor di testi, che ha il vantaggio di fornire automaticamente funzionalità che facilitano la programmazione (vedremo)

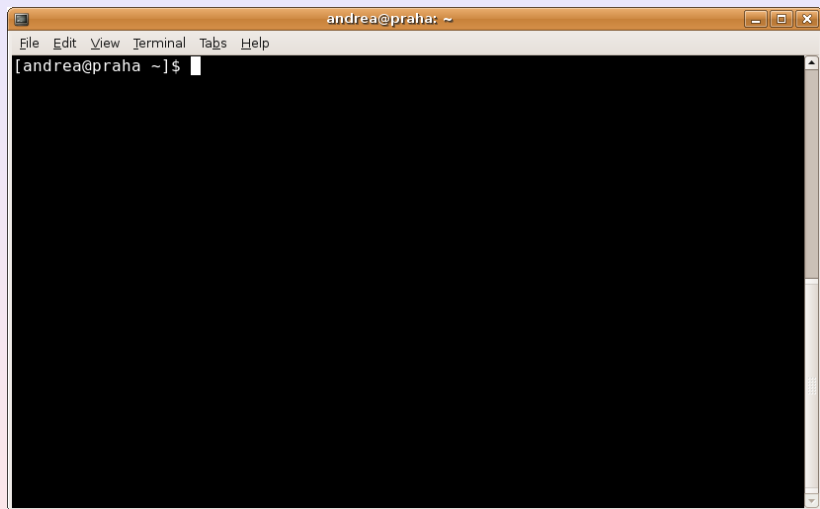
Applicazioni Internet

- *Firefox* web browser per la navigazione su Internet
- *Mozilla thunderbird* o *Evolution* per la gestione della posta elettronica
- *Skype* per la comunicazione multimediale a distanza
- e molti altri applicativi

Interprete di comandi

- E' possibile interagire con il sistema tramite un interprete di comandi a linea aprendo un'applicazione *terminale* (Applicazioni → Accessori → Terminale)
- L'interprete a linea di comandi, rispetto a quello grafico, è meno accattivante, ma tipicamente più rapido e versatile
- Infatti l'interprete a linea di comandi si basa sull'intera tastiera per scrivere comandi (e permette quindi di usare tutte e dieci le dita), rispetto all'interfaccia grafica che si basa sul mouse (e tendenzialmente sull'utilizzo di combinazioni di tasti cui vengono associate certe operazioni)
- E' sempre possibile lanciare un programma disponibile tramite l'interfaccia grafica (e.g. office) scrivendo il nome dell'eseguibile sulla linea di comando (e premendo *Invio*)

Interprete di comandi



Interprete di comandi

- L'interprete dei comandi agisce in maniera interattiva, eseguendo immediatamente un comando che l'utente scrive al prompt (una volta premuto *Invio*)
- Ubuntu, come tutti i sistemi derivati da UNIX, fornisce una vasta gamma di comandi di base disponibili da linea di comando, che permettono di esplorare il file system, modificare files, recuperare informazione selezionata, e fare un gran numero di operazioni
- Ogni comando ha un set di *opzioni* per decidere i dettagli di cosa deve fare, ed un certo numero (anche nessuno) di argomenti su cui opera

Interprete di comandi

- Un singolo comando viene tipicamente lanciato con questa sintassi

```
<nomecomando> [<opzioni>] [<argom1>] ... [<argomN>]
```

- [`<opzioni>`] sono le opzioni con cui eseguire il comando (se diverse dal default), tipicamente una lettera per opzione, preceduta da `-` (e.g. `ls -l`)
- Quando si specificano più opzioni, è possibile specificare una singola `-` seguita dalla stringa di opzioni richieste (e.g. `ls -lh`)
- [`<argom1>`] ... [`<argomN>`] sono gli argomenti del comando (se presenti), che possono essere nomi di file, espressioni regolari, etc. (e.g. `cd Figures`)

Esplorazione del file system: `ls`

- `ls` (LiSt) elenca il contenuto di una directory
- All'avvio di un terminale, la directory corrente viene fissata alla *home* dell'utente, al di sotto della quale si trovano tutti i file dell'utente stesso, organizzati in directories.

```
[andrea@praha ~]$ ls
acc_vs_roc.pdf          octave-core
allthingstodo.doc      Personal
AUC_comparison.pdf    Programs
Backup                 public_html
bin                    roc_comparison.pdf
Calendario_2008-09.pdf SCHEDULE
chimera                singer_pegasos.pdf
COMMON                SRC
data                   TIPS
Data                   tmp
DESIDERATA             tmp1
Desktop                TODO
```

Esplorazione del file system: `ls`

- L'opzione `-l` permette di stampare informazioni aggiuntive su ciascun file, inclusi:
 - operazioni permesse, per proprietario, gruppo ed altri
 - proprietario e gruppo
 - dimensione
 - data dell'ultima modifica

```
[andrea@praha 08_sistemi_operativi]$ ls -l
total 1420
-rw-r--r-- 1 andrea andrea  22645 2009-01-14 11:09 content.tex
-rw----- 1 andrea andrea   5427 2009-01-13 17:43 definitions.tex
drwxr-sr-x 2 andrea andrea   4096 2009-01-14 11:02 Figures
-rw-r--r-- 1 andrea andrea 1251873 2009-01-14 11:03 talk.pdf
```

Esplorazione del file system: `ls`

- L'opzione `-h` permette di stampare in maniera più comprensibile l'informazione sulla dimensione dei file (implica l'opzione `-l`)
- Le opzioni possono essere combinate arbitrariamente (e.g. `-lh`)

```
[andrea@praha 08_sistemi_operativi]$ ls -lh
total 1.4M
-rw-r--r-- 1 andrea andrea  23K 2009-01-14 11:12 content.tex
-rw----- 1 andrea andrea  5.3K 2009-01-13 17:43 definitions.tex
drwxr-sr-x 2 andrea andrea  4.0K 2009-01-14 11:02 Figures
-rw-r--r-- 1 andrea andrea 1.2M 2009-01-14 11:12 talk.pdf
```

Esplorazione del file system: `ls`

- Di default, i file nella directory vengono stampati da `ls` in ordine lessicografico
- E' possibile specificare delle opzioni che modificano tale ordine. Ad esempio `-t` ordina i file per data di modifica (dal più recente al più vecchio)

```
[andrea@praha 08_sistemi_operativi]$ ls -lth
total 1.4M
-rw-r--r-- 1 andrea andrea 1.3M 2009-01-14 11:17 talk.pdf
-rw-r--r-- 1 andrea andrea 25K 2009-01-14 11:17 content.tex
drwxr-sr-x 2 andrea andrea 4.0K 2009-01-14 11:02 Figures
-rw----- 1 andrea andrea 5.3K 2009-01-13 17:43 definitions.tex
```

Esplorazione del file system: `ls`

- Oltre alle opzioni, `ls` può anche prendere una lista di argomenti in ingresso
- In tal caso, lista il contenuto delle directory e file presi come argomento, invece che quello della directory corrente

```
[andrea@praha 08_sistemi_operativi]$ ls -lh Figures/ definitions.tex
-rw----- 1 andrea andrea 5.3K 2009-01-13 17:43 definitions.tex
```

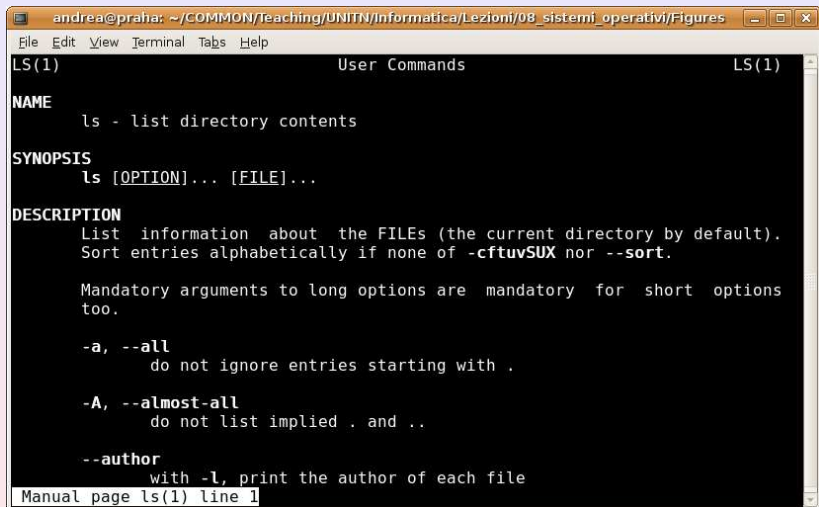
```
Figures/:
```

```
total 2.4M
```

```
-rw-r--r-- 1 andrea andrea 1.1M 2009-01-13 19:17 desktop.pdf
-rw-r--r-- 1 andrea andrea 866K 2009-01-13 19:17 desktop.png
-rw-r--r-- 1 andrea andrea 231K 2009-01-14 11:22 man.eps
-rw-r--r-- 1 andrea andrea 49K 2009-01-14 11:22 man.pdf
-rw-r--r-- 1 andrea andrea 92K 2009-01-14 11:02 terminale.eps
-rw-r--r-- 1 andrea andrea 15K 2009-01-14 11:02 terminale.pdf
```

Aiuto sui comandi: `man`

- Il comando `man` (`MANual`) seguito dal nome di un comando stampa informazione relativa al comando stesso
- L'informazione contiene la spiegazione di cosa il comando fa, come va lanciato, la descrizione delle opzioni disponibili ed in genere esempi di utilizzo
- `man` è molto utile per esplorare il funzionamento dei vari comandi di shell, e trovare opzioni nuove o che non si ricordano
- E.g. eseguendo `man ls` si ottiene una pagina di manuale relativa ad `ls`
- Tipicamente l'output di `man` è più lungo dello spazio disponibile nel terminale. Si scorre l'output premendo la *barra*, se ne esce premendo *q* (*quit*)



The image shows a terminal window titled "andrea@praha: ~/COMMON/Teaching/UNITN/Informatica/Lezioni/08_sistemi_operativi/Figures". The terminal displays the manual page for the 'ls' command. The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal content is as follows:

```
LS(1) User Commands LS(1)
NAME
  ls - list directory contents
SYNOPSIS
  ls [OPTION]... [FILE]...
DESCRIPTION
  List information about the FILES (the current directory by default).
  Sort entries alphabetically if none of -cftuvSUX nor --sort.

  Mandatory arguments to long options are mandatory for short options
  too.

  -a, --all
      do not ignore entries starting with .

  -A, --almost-all
      do not list implied . and ..

  --author
      with -l, print the author of each file
Manual page ls(1) line 1
```

Esplorazione del file system: `cd`

- Per spostarsi all'interno della struttura a directory del file system, si usa il comando `cd` (Change Directory)
- `cd` ha come unico argomento la directory in cui si desidera spostarsi
- come per tutti i comandi che prendono come argomenti file, a `cd` si deve specificare il *percorso* del file

```
[andrea@praha 08_sistemi_operativi]$ cd /home/andrea/Data/  
[andrea@praha Data]$ ls  
Fasta  Labels  README  Results  
[andrea@praha Data]$ cd Labels/  
[andrea@praha Labels]$
```

Percorsi di file

- Qualunque file, regolare o directory che sia, viene identificato all'interno del file system tramite un percorso di directory, dalla radice al file stesso
- Nei sistemi UNIX, la radice del file system è indicata con / ed i separatori tra directory si indicano anch'essi con /
- Esistono due modi per specificare il percorso di un file:
 - percorso assoluto** è il percorso complessivo del file, dalla radice fino alla sua posizione nel file system (e.g. /home/andrea/Data/)
 - percorso relativo** è il percorso del file a partire dalla directory corrente (e.g. Labels/label.txt)

Percorsi di file

- Esistono alcuni caratteri speciali che hanno un significato particolare se usati nel percorso di un file
- Il carattere tilde / indica la home dell'utente, e può essere usato ogni qualvolta si voglia riferirsi ad essa (e.g. nel mio computer / corrisponde a /home/andrea)

```
[andrea@praha Labels]$ cd ~  
[andrea@praha ~]$
```

- La sequenza .. indica la directory subito sopra a quella corrente nella gerarchia. Tale sequenza può essere usata più di una volta per risalire la gerarchia in un solo passo

```
[andrea@praha Labels]$ cd ../  
[andrea@praha Data]$ ls  
Fasta  Labels  README  Results  
[andrea@praha Data]$ cd ../../  
[andrea@praha home]$ ls  
andrea  lost+found  pankow
```

Esplorazione del file system: `cd`

- Se specificato senza argomenti, `cd` sposta nella home dell'utente
- L'argomento speciale `-` indica di spostarsi nella directory in cui si era prima di aver eseguito l'ultimo `cd` (e stampa il nome di tale directory in output)

```
[andrea@praha Labels]$ cd  
[andrea@praha ~]$ cd -  
/home/andrea/Data/Labels  
[andrea@praha Labels]$
```

Modifica del file system: `cp`

- `cp` (CoPY) serve a copiare file da un posto all'altro
- `cp` prende come argomenti: un file da copiare, ed un file o una directory di destinazione
- Come sempre quando si specifica un nome di file, si può usare un percorso assoluto, o relativo alla directory corrente
- Se la destinazione è un file, `cp` crea un file con quel nome e vi copia il contenuto del file dato come primo argomento
- Se la destinazione è una directory, `cp` crea in tale directory un file con lo stesso nome del file da copiare, e vi copia il contenuto

Modifica del file system: cp

```
[andrea@praha Data]$ ls
Fasta  Labels  README  Results
[andrea@praha Data]$ cp README README2
[andrea@praha Data]$ ls
Fasta  Labels  README  README2  Results
[andrea@praha Data]$ cp README ~
[andrea@praha Data]$ ls ~
allthingstodo.doc          Data                      Lion.pdf
AUC_comparison.pdf        DESIDERATA                Mail
Backup                    Desktop                   octave-core
bin                        fernando06-CGM.pdf       Personal
Calendario_2008-09.pdf    JavaNNS.properties       Programs
CHANGES_DISTRO          kitchin.JPG               public_html
chimera                    Kitchovitch_new.pdf      README
```

Modifica del file system: `cp`

- `cp` permette di copiare più files con un solo comando
- `cp` si aspetta infatti di avere un numero variabile di argomenti, minimo due
- l'ultimo argomento sarà sempre la destinazione in cui copiare
- tutti gli altri argomenti precedenti sono i file da copiare
- Il risultato è che ciascun file viene copiato nella destinazione
- Se più di un file viene copiato, la destinazione dovrà essere un nome di directory (in cui verranno copiati i file)

Modifica del file system: `cp`

```
[andrea@praha Data]$ ls
Fasta  Labels  README  README2  Results  tmp  TODO
[andrea@praha Data]$ cp README README2  TODO  tmp
[andrea@praha Data]$ ls tmp/
README  README2  TODO
```

- Se la directory di destinazione non esiste, si genera un errore
- Se la destinazione è un file normale e non una directory, si genera un errore

Modifica del file system: `cp`

- Di base, `cp` copia files e non directories.
- L'opzione `-r` (*ricorsivo*) permette di copiare ricorsivamente il contenuto di un'intera directory
- In tal caso, la destinazione deve essere un nome di directory, che verrà creata ed in cui verranno copiati ricorsivamente tutte le directory ed i file contenuti nella sorgente

```
[andrea@praha Data]$ ls
Fasta  Labels  README  Results
[andrea@praha Data]$ ls Results
AbInitio  computeAverage.pl  computeResults.sh
[andrea@praha Data]$ cp -r Results Results2
[andrea@praha Data]$ ls
Fasta  Labels  README  Results  Results2
[andrea@praha Data]$ ls Results2/
AbInitio  computeAverage.pl  computeResults.sh
```

Modifica del file system: `mv`

- Il comando `mv` (MoVe) serve a spostare file all'interno del sistema, o a rinominarli
- `mv` prende come argomenti uno o più file o directory, ed un file o directory di destinazione, come `cp`
- con un argomento di ingresso (file o directory) ed un argomento di destinazione che non esiste, `mv` rinomina l'ingresso con il nome della destinazione

```
[andrea@praha Data]$ ls
Fasta  Labels  README  Results  tmp  TODO
[andrea@praha Data]$ mv README README2
[andrea@praha Data]$ ls
Fasta  Labels  README2  Results  tmp  TODO
[andrea@praha Data]$ mv tmp/ tmp2
[andrea@praha Data]$ ls
Fasta  Labels  README2  Results  tmp2  TODO
```

Modifica del file system: `mv`

- con un file in ingresso ed un file destinazione già esistente, `mv` sposta l'ingresso nell'uscita, sovrascrivendo il suo contenuto

```
[andrea@praha Data]$ ls -l
total 24
drwxr-xr-x 2 andrea andrea 4096 2008-12-22 13:06 Fasta
drwxr-xr-x 2 andrea andrea 4096 2008-12-22 13:03 Labels
-rw-r--r-- 1 andrea andrea 1159 2008-12-22 13:58 README2
drwxr-xr-x 4 andrea andrea 4096 2008-12-22 13:57 Results
drwxr-xr-x 3 andrea andrea 4096 2009-01-14 13:29 tmp2
-rw-r--r-- 1 andrea andrea 3 2009-01-14 12:30 TODO
[andrea@praha Data]$ mv README2 TODO
[andrea@praha Data]$ ls -l
total 20
drwxr-xr-x 2 andrea andrea 4096 2008-12-22 13:06 Fasta
drwxr-xr-x 2 andrea andrea 4096 2008-12-22 13:03 Labels
drwxr-xr-x 4 andrea andrea 4096 2008-12-22 13:57 Results
drwxr-xr-x 3 andrea andrea 4096 2009-01-14 13:29 tmp2
-rw-r--r-- 1 andrea andrea 1159 2008-12-22 13:58 TODO
```

Modifica del file system: `mv`

- con uno o più file o directory in ingresso ed una destinazione che è una directory, `mv` sposta gli ingressi nella destinazione
- Se la directory di destinazione non esiste, si genera un errore
- Se la destinazione è un file normale e non una directory, si genera un errore

```
[andrea@praha Data]$ ls
Fasta  Labels  README  Results  tmp2  TODO
[andrea@praha Data]$ mv README  Labels/ tmp2/
[andrea@praha Data]$ ls tmp2/
Labels  README
[andrea@praha Data]$ ls
Fasta  Results  tmp2  TODO
```

Modifica del file system: `rm`

- il comando `rm` (ReMove) serve ad eliminare file
- come molti comandi visti finora, prende come argomenti uno o più file da cancellare
- a differenza della cancellazione di file fatta tramite l'interfaccia grafica, `rm` NON sposta i file cancellati nel cestino, ma li elimina in maniera definitiva
- per questa ragione, `rm` tipicamente viene rimappato dal sistema (con un *alias*) in `rm -i`, in cui l'opzione `-i` chiede una conferma per ogni file da cancellare

```
[andrea@praha tmp2]$ ls
Labels README
[andrea@praha tmp2]$ rm README
rm: remove regular file `README'? y
[andrea@praha tmp2]$ ls
Labels
```

Modifica del file system: `rm`

- se si è sicuri dei file che si vogliono cancellare, l'opzione `-f` elimina le richieste di conferma, ed inoltre non produce messaggi di errore se alcuni dei file specificati non esistono

```
[andrea@praha tmp2]$ ls
Labels  README  TODO
[andrea@praha tmp2]$ rm -f TODO  README  nulla
[andrea@praha tmp2]$ ls
Labels
```

Modifica del file system: `rm`

- `rm` cancella file, non directory. La cancellazione di directory è critica in quanto si cancellano intere porzioni di file system, e deve essere specificata tramite l'opzione `-r` (ricorsivo).

```
[andrea@praha tmp2]$ rm -f Labels/  
rm: cannot remove 'Labels/': Is a directory  
[andrea@praha tmp2]$ rm -rf Labels/  
[andrea@praha tmp2]$ ls  
[andrea@praha tmp2]$
```

Modifica del file system: `mkdir`

- il comando `mkdir` (MaKeDIRectory) crea una directory con il nome specificato

```
[andrea@praha tmp2]$ mkdir tmp3
[andrea@praha tmp2]$ ls
tmp3
```

- se le directory nel percorso fornito come argomento non esistono, il comando restituisce un errore. Per creare un intero percorso di directory (se necessario), si usa l'opzione `-p` (*parents*)

```
[andrea@praha tmp2]$ mkdir tmp/tmp/tmp
mkdir: cannot create directory `tmp/tmp/tmp':
  No such file or directory
[andrea@praha tmp2]$ mkdir -p tmp/tmp/tmp
[andrea@praha tmp2]$ ls tmp/tmp/
tmp
```

Lettura di file: `cat`

- il comando `cat` (conCATenate) permette di leggere uno o più files
- prende come argomento una lista di nomi di file, e restituisce in uscita la concatenazione dei loro contenuti

```
[andrea@praha Data]$ cat README
Labels contains labeling of experimental data
[andrea@praha Data]$ cat TODO
- remember to clean un directory tmp after use

- DO NOT overwrite Labels
[andrea@praha Data]$ cat README TODO
Labels contains labeling of experimental data
- remember to clean un directory tmp after use

- DO NOT overwrite Labels
[andrea@praha Data]$
```

Wildcards

- L'interprete di comandi fornisce un modo molto semplice e potente per eseguire un comando contemporaneamente su più files
- Tale operazione è possibile per ogni comando che prende come argomenti liste di files e directory (ad esempio `cp`)
- L'operazione si basa sull'uso di caratteri speciali, detti *wildcards*, per specificare in maniera compatta un insieme di file e directory
- L'interprete di comandi interpreterà tali caratteri speciali, e li sostituirà con l'elenco di nomi di file che sono con essi compatibili (*espansione*), per poi lanciare il comando con tale elenco come argomenti
- In pratica, gli *wildcards* permettono di specificare semplici *espressioni regolari*, e tutti i file che le soddisfano verranno selezionati

Wildcards

* corrisponde ad una qualsiasi sequenza di caratteri (anche vuota)

? corrisponde ad un singolo carattere

[abcde] corrisponde ad uno qualsiasi dei caratteri specificati (abcde)

[0-9] corrisponde ad uno qualsiasi dei caratteri nel range specificato (ossia 0123456789)

[!0123] corrisponde ad un qualsiasi carattere che non sia nella lista (ossia non 0123)

[!a-e] corrisponde ad un qualsiasi carattere che non sia nel range (ossia non abcde)

{html,xml} corrisponde ad una delle parole specificate nella lista

Wildcards

```
[andrea@praha tmp]$ ls
README  README2  TODO
[andrea@praha tmp]$ cp * ../tmp1/
[andrea@praha tmp]$ ls ../tmp1/
README  README2  TODO
[andrea@praha tmp]$ rm -f ../tmp1/READ*
[andrea@praha tmp]$ ls ../tmp1/
TODO
[andrea@praha tmp]$ ls
0 1 2 3 4 5 6 README Doc Util data.txt todo.txt nota.doc
[andrea@praha tmp]$ cp -d [0-9] Util
0 1 2 3 4 5 6 README Doc Util
[andrea@praha tmp]$ ls Util
0 1 2 3 4 5 6
[andrea@praha tmp]$ ls *.{txt,doc}
data.txt todo.txt nota.doc
```

Esecuzione di programmi generici

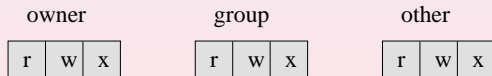
- L'interprete di comandi permette di eseguire qualunque comando sia disponibile sul calcolatore
- Vedremo esempi di uso della linea di comando per lanciare programmi di rete, o programmi scritti da noi in un linguaggio di programmazione
- E' possibile usare l'interprete per lanciare un programma con interfaccia grafica, nel qual caso l'interprete farà partire l'interfaccia corrispondente

Permessi

- Tutte le operazioni fatte sul file system sono vincolate ai permessi su file e directory
- Perché l'operazione vada a buon fine, l'utente che esegue il comando deve avere i permessi necessari
- Se l'utente tenta di eseguire un comando per cui non ha i permessi, il sistema operativo genera un errore, ed il comando non ha effetto

Protezione nel File System (UNIX)

- Si distingue tra:
 - proprietario del file
 - utente appartenente allo stesso gruppo del proprietario
 - altro utente
- Si distinguono i permessi di:
 - scrittura
 - lettura
 - esecuzione
- In totale 9 flags specificano i permessi di un file:



Significato dei permessi

Il significato dei permessi differisce se si tratta di file o di directory

file

lettura è possibile leggere il contenuto del file

scrittura è possibile modificare il contenuto del file

esecuzione è possibile eseguire il file (nel caso in cui il file contenga un programma)

directory

lettura è possibile recuperare l'elenco dei file contenuti nella directory

scrittura è possibile creare un nuovo file nella directory

esecuzione è possibile entrare nella directory o attraversarla per entrare in una sua sottodirectory