

# Introduzione all'uso di Linux

Violetta Lonati

Università degli studi di Milano  
Dipartimento di Scienze dell'Informazione

Laboratorio di algoritmi e strutture dati  
Corso di laurea in Informatica

20 ottobre 2010

## Alcune parole chiave

- ▶ **software applicativo**: costituito da programmi che svolgono funzionalità rivolte prevalentemente agli utilizzatori finali
- ▶ **software di base/di sistema**: insieme di programmi necessari al funzionamento del computer
  - ▶ sistema operativo
  - ▶ driver delle periferiche
- ▶ **Algoritmo**: procedimento formato da una sequenza finita di passi elementari che conducano alla soluzione di un problema, o più in generale allo svolgimento di un compito.
- ▶ **Programma**: descrizione di un algoritmo in un linguaggio adatto ad essere eseguito da un computer (linguaggio di programmazione). Si tratta di un'entità statica.
- ▶ **Processo**: istanza di un programma in esecuzione. Si tratta di un'entità dinamica.

# Sistema operativo

- ▶ è necessario per il funzionamento del computer
  - ▶ di solito è installato su disco fisso
  - ▶ può anche essere avviato tramite un altro supporto (es: cd, chiavetta usb, floppy)

# Sistema operativo

- ▶ è necessario per il funzionamento del computer
  - ▶ di solito è installato su disco fisso
  - ▶ può anche essere avviato tramite un altro supporto (es: cd, chiavetta usb, floppy)
- ▶ serve per:
  - ▶ gestire risorse
    - ▶ gestione periferiche
    - ▶ esecuzione simultanea di programmi (processore)
    - ▶ memorizzazione e sicurezza dei dati (file system)
    - ▶ gestione multi-utente (login)

# Sistema operativo

- ▶ è necessario per il funzionamento del computer
  - ▶ di solito è installato su disco fisso
  - ▶ può anche essere avviato tramite un altro supporto (es: cd, chiavetta usb, floppy)
- ▶ serve per:
  - ▶ gestire risorse
    - ▶ gestione periferiche
    - ▶ esecuzione simultanea di programmi (processore)
    - ▶ memorizzazione e sicurezza dei dati (file system)
    - ▶ gestione multi-utente (login)
  - ▶ facilitare l'uso del computer
    - ▶ agli utenti in genere: fornendo un'interfaccia (grafica e/o testuale)
    - ▶ ai programmatori: fornendo funzionalità ad alto livello che mascherano l'hardware

## Nelle aule sigma/tau

- ▶ **Dual boot**: sono installati due sistemi operativi che si possono usare alternativamente.
  - ▶ Windows Xp
  - ▶ Gnu/Linux Gentoo

## Nelle aule sigma/tau

- ▶ **Dual boot**: sono installati due sistemi operativi che si possono usare alternativamente.
  - ▶ Windows Xp
  - ▶ Gnu/Linux Gentoo
- ▶ All'avvio bisogna scegliere e indicare quale sistema operativo usare.

## Nelle aule sigma/tau

- ▶ **Dual boot**: sono installati due sistemi operativi che si possono usare alternativamente.
  - ▶ Windows Xp
  - ▶ Gnu/Linux Gentoo
- ▶ All'avvio bisogna scegliere e indicare quale sistema operativo usare.
- ▶ Noi useremo Linux

## Nelle aule sigma/tau

- ▶ **Dual boot**: sono installati due sistemi operativi che si possono usare alternativamente.
  - ▶ Windows Xp
  - ▶ Gnu/Linux Gentoo
- ▶ All'avvio bisogna scegliere e indicare quale sistema operativo usare.
- ▶ Noi useremo Linux
- ▶ Se è già attivo Windows bisogna fare un **reboot** (riavvio) della macchina con **CTRL + ALT + CANC** , quindi scegliere Linux

## Nelle aule sigma/tau

- ▶ **Dual boot**: sono installati due sistemi operativi che si possono usare alternativamente.
  - ▶ Windows Xp
  - ▶ Gnu/Linux Gentoo
- ▶ All'avvio bisogna scegliere e indicare quale sistema operativo usare.
- ▶ Noi useremo Linux
- ▶ Se è già attivo Windows bisogna fare un **reboot** (riavvio) della macchina con **CTRL + ALT + CANC** , quindi scegliere Linux
- ▶ Una volta avviato il sistema operativo bisogna autenticarsi con una procedura chiamata **login**. Il sistema è multiutente: ciascun utente ha un proprio **account** individuato da
  - ▶ **username** (in italiano **nome\_utente**)
  - ▶ **password**

# Gnu/Linux

- ▶ Cos'è GNU/Linux?
  - ▶ è un sistema operativo libero di tipo Unix
  - ▶ è un software rilasciato con una licenza che permette a chiunque di utilizzarlo e che ne incoraggia lo studio, le modifiche e la redistribuzione
  - ▶ si può installare senza costo su ormai quasi tutti i PC

# Gnu/Linux

- ▶ Cos'è GNU/Linux?
  - ▶ è un sistema operativo libero di tipo Unix
  - ▶ è un software rilasciato con una licenza che permette a chiunque di utilizzarlo e che ne incoraggia lo studio, le modifiche e la redistribuzione
  - ▶ si può installare senza costo su ormai quasi tutti i PC
- ▶ Perché GNU/Linux?
  - ▶ ci permette di introdurre concetti fondamentali e strumenti potenti molto utili per un informatico
  - ▶ probabilmente conoscete già Windows... siete qui per imparare cose nuove!
  - ▶ questa è solo un'infarinatura: nel corso di sistemi operativi approfondirete meglio questi concetti

# Gnu/Linux

- ▶ Cos'è GNU/Linux?
  - ▶ è un sistema operativo libero di tipo Unix
  - ▶ è un software rilasciato con una licenza che permette a chiunque di utilizzarlo e che ne incoraggia lo studio, le modifiche e la redistribuzione
  - ▶ si può installare senza costo su ormai quasi tutti i PC
- ▶ Perché GNU/Linux?
  - ▶ ci permette di introdurre concetti fondamentali e strumenti potenti molto utili per un informatico
  - ▶ probabilmente conoscete già Windows... siete qui per imparare cose nuove!
  - ▶ questa è solo un'infarinatura: nel corso di sistemi operativi approfondirete meglio questi concetti
- ▶ Cos'è una distribuzione Linux?
  - ▶ è una distribuzione software che include un kernel Linux e un insieme variabile di altri strumenti e applicazioni software, compresi strumenti che guidano l'utente nella fase di installazione

# Interprete dei comandi (shell)

È un programma che permette all'utente di interagire con la macchina, impartendo comandi e chiedendo l'esecuzione di programmi.

## GUI - graphic user interface

- ▶ desktop, icone, finestre
- ▶ uso del mouse
- ▶ intuitivo, facile per l'utente inesperto
- ▶ consuma risorse
- ▶ scomodo in rete
- ▶ es: Windows Explorer, Gnome o KDE per Linux

## CLI - command line interface

- ▶ digitazione comandi
- ▶ regole di sintassi
- ▶ richiede conoscenza più avanzata
- ▶ op. ripetitive e complesse
- ▶ strumento potente e veloce
- ▶ es: prompt di MS-DOS, bash per Linux

# Bash

È una shell testuale del progetto GNU, usata in Unix e Linux

- ▶ contiene una serie di comandi predefiniti
- ▶ permette di richiedere l'esecuzione di programmi
- ▶ mette a disposizione un linguaggio di programmazione → **script**

# Bash

È una shell testuale del progetto GNU, usata in Unix e Linux

- ▶ contiene una serie di comandi predefiniti
- ▶ permette di richiedere l'esecuzione di programmi
- ▶ mette a disposizione un linguaggio di programmazione → **script**

Introduzione all'uso della bash:

- ▶ documentazione online → **man**
- ▶ come impartire comandi e eseguire programmi
- ▶ facilitazioni → **completamento e history**
- ▶ sintassi dei comandi
- ▶ comandi per la gestione dei file
- ▶ comandi per la gestione dei processi
- ▶ redirectione di input/output

## Primi passi con la bash

- ▶ Per poter usare la bash è necessario avviare un **emulatore di terminale** (es: **Konsole**), ovvero un programma che fornisce una finestra testuale attraverso la quale impartire i comandi.
  - ▶ **Avviate un terminale tramite il menu.**
- ▶ Il terminale attende vostre istruzioni, e ve lo dice mostrandovi prompt: `(pct1-03:~) v1123456%`

## Primi passi con la bash

- ▶ Per poter usare la bash è necessario avviare un **emulatore di terminale** (es: **Konsole**), ovvero un programma che fornisce una finestra testuale attraverso la quale impartire i comandi.
  - ▶ **Avviate un terminale tramite il menu.**
- ▶ Il terminale attende vostre istruzioni, e ve lo dice mostrandovi prompt: `(pct1-03:~) v1123456%`

### Che succede se:

- ▶ scrivete qualche carattere a casaccio, seguito dal tasto **INVIO** ?
- ▶ digitate il tasto **UP** (freccia su)?
- ▶ digitate la lettera **a** seguita dal **TAB** ?
- ▶ usate il mouse all'interno della finestra?
- ▶ evidenziate del testo tenendo cliccato il tasto sinistro del mouse, poi spostate il mouse e cliccate col tasto centrale (se avete solo due tasti, cliccate contemporaneamente i due tasti destro e sinistro)?

# Comandi, argomenti, parametri

La sintassi generale di un comando è la seguente

`nome_comando` `opzioni` `argomenti`

opzioni e argomenti possono anche non esserci

# Comandi, argomenti, parametri

La sintassi generale di un comando è la seguente

`nome_comando` `opzioni` `argomenti`

opzioni e argomenti possono anche non esserci

- ▶ Il comando `man` seguito dal nome di un comando mostra le pagine della guida relative a quel comando. **Provate a digitare il comando `man man` seguito da invio. Cosa scoprite?**
- ▶ Una versione ridotta del `man` si ottiene digitando un comando con l'opzione `--help`

# Comandi, argomenti, parametri

La sintassi generale di un comando è la seguente

`nome_comando` `opzioni` `argomenti`

opzioni e argomenti possono anche non esserci

- ▶ Il comando `man` seguito dal nome di un comando mostra le pagine della guida relative a quel comando. **Provate a digitare il comando `man man` seguito da invio. Cosa scoprite?**
- ▶ Una versione ridotta del `man` si ottiene digitando un comando con l'opzione `--help`

## Esercizio:

- ▶ Scoprite a cosa servono i comandi `uname` , `whoami` , `cal` .
- ▶ Provate ad usare gli stessi comandi con dei parametri.
- ▶ Provate a digitare il comando `mozilla-firefox` . Che succede?

## Navigare nel file system - percorsi assoluti

- ▶ Il file system memorizza i dati usando dei **file** organizzati in **directory** e **subdirectory**, secondo una struttura gerarchica ad albero, con una radice detta **root** e indicata con `/`.
- ▶ Ogni file all'interno del file system è individuato in base alla sua posizione nell'albero, cioè al cammino (**path assoluto**) che si deve percorrere per raggiungerlo a partire dalla radice.
  - ▶ Es: `/home/violi/mio_file.txt` → uso di `/`

## Navigare nel file system - percorsi assoluti

- ▶ Il file system memorizza i dati usando dei **file** organizzati in **directory** e **subdirectory**, secondo una struttura gerarchica ad albero, con una radice detta **root** e indicata con `/`.
- ▶ Ogni file all'interno del file system è individuato in base alla sua posizione nell'albero, cioè al cammino (**path assoluto**) che si deve percorrere per raggiungerlo a partire dalla radice.
  - ▶ Es: `/home/violi/mio_file.txt` → uso di `/`
- ▶ Al momento del login ci si trova nella propria directory che ha pathname `/home/nome_utente`. Usate il comando `pwd` (**print work directory**) per stampare il path della directory corrente.
- ▶ Per cambiare la directory, basta usare il comando `cd` seguito dal path della directory desiderata (il tasto tab può essere d'aiuto).
  - ▶ Provate a digitare il comando `cd /usr/bin/`. Come è cambiato il prompt?
  - ▶ Verificate in che directory siete usando di nuovo il comando `pwd`.

## Navigare nel file system - percorsi relativi

Al posto di usare i path assoluti è possibile individuare un file definendo il cammino che si deve percorrere per raggiungerlo partendo dalla directory corrente, anzichè dalla radice. Tale cammino è detto **path relativo**.

## Navigare nel file system - percorsi relativi

Al posto di usare i path assoluti è possibile individuare un file definendo il cammino che si deve percorrere per raggiungerlo partendo dalla directory corrente, anzichè dalla radice. Tale cammino è detto **path relativo**.

- ▶ Si possono usare due simboli speciali per rappresentare le posizioni all'interno dell'albero dei file:
  - ▶ `.` si riferisce alla directory corrente
  - ▶ `..` si riferisce alla directory **padre**, cioè di un livello superiore
- ▶ Scorciatoia: il comando `cd` senza argomenti, vi porta nella vostra home; il simbolo `~nome_utente` è un'abbreviazione per `/home/nome_utente`; se non è specificato il `nome_utente`, il simbolo `~` è un'abbreviazione per la propria home.

## Navigare nel file system - percorsi relativi

Al posto di usare i path assoluti è possibile individuare un file definendo il cammino che si deve percorrere per raggiungerlo partendo dalla directory corrente, anzichè dalla radice. Tale cammino è detto **path relativo**.

- ▶ Si possono usare due simboli speciali per rappresentare le posizioni all'interno dell'albero dei file:
  - ▶ `.` si riferisce alla directory corrente
  - ▶ `..` si riferisce alla directory **padre**, cioè di un livello superiore
- ▶ Scorciatoia: il comando `cd` senza argomenti, vi porta nella vostra home; il simbolo `~nome_utente` è un'abbreviazione per `/home/nome_utente`; se non è specificato il `nome_utente`, il simbolo `~` è un'abbreviazione per la propria home.
- ▶ Posizionatevi nella vostra home, quindi nella directory `/usr/bin` poi nella home del vostro vicino, senza mai usare percorsi assoluti.

# Visualizzare file e directory

- ▶ Visualizzate il contenuto della vostra home e della directory `/usr/bin` sperimentando il comando `ls` e le sue opzioni `-l`, `-a`, `-r`, `-u`
- ▶ Scoprite a cosa serve il comando `file`.
- ▶ Provate ad usare il comando `less` per visualizzare il contenuto di un file di testo.
  - ▶ Usate i tasti `UP` `DOWN` `PageUP` `PageDOWN` per scorrere il testo
  - ▶ Provate a digitare le lettere `/a`. Cosa succede?
  - ▶ Cosa succede se subito dopo digitate `n`?
  - ▶ Per uscire dal `less`, digitate `q`

## Manipolare file

NOTA: i nomi dei file e delle dir sono **case sensitive**:

MAIUSCOLO  $\neq$  minuscolo.

NOTA: sconsiglio fortemente di usare gli spazi nei nomi di file e dir!

# Manipolare file

NOTA: i nomi dei file e delle dir sono **case sensitive**:

MAIUSCOLO  $\neq$  minuscolo.

NOTA: sconsiglio fortemente di usare gli spazi nei nomi di file e dir!

## Esercizio

- ▶ Create nella vostra home 3 file `pippo1` , `pippo2` e `pippo3` con il comando `touch`
- ▶ Create nella vostra home una directory chiamata `lab_algoritmi` usando il comando `mkdir`
- ▶ Tornate nella vostra home e, da lì, create una nuova cartella `intro_linux` all'interno della dir `lab_algoritmi`
- ▶ Con il comando `mv` spostate i file `pippo1` , `pippo2` e `pippo3` nella cartella `intro_linux` appena creata
- ▶ Con il comando `rm` cancellate il file `pippo3`

## Wildcards

Spesso si ha bisogno di specificare in modo veloce gruppi di file in base al loro nome. Per fare questo la bash mette a disposizione alcuni caratteri chiamati **wildcards**, ovvero Jolly.

- ▶ **\*** indica una qualunque sequenza di zero o più caratteri
- ▶ **?** indica un qualunque carattere
- ▶ **[A-Z]** indica un qualunque carattere tra A e Z
- ▶ **[0-9]** indica una qualunque cifra decimale
- ▶ ...

Ad esempio, il comando `ls -l /usr/bin/a*` visualizza tutti i file della dir `/usr/bin` che iniziano per `a`

# Wildcards

Spesso si ha bisogno di specificare in modo veloce gruppi di file in base al loro nome. Per fare questo la bash mette a disposizione alcuni caratteri chiamati **wildcards**, ovvero Jolly.

- ▶ \* indica una qualunque sequenza di zero o più caratteri
- ▶ ? indica un qualunque carattere
- ▶ [A-Z] indica un qualunque carattere tra A e Z
- ▶ [0-9] indica una qualunque cifra decimale
- ▶ ...

Ad esempio, il comando `ls -l /usr/bin/a*` visualizza tutti i file della dir `/usr/bin` che iniziano per `a`

## Esercizio

- ▶ Visualizzate l'elenco dei file in `/usr/bin` il cui nome finisce per `x`
- ▶ Visualizzate l'elenco dei file in `/usr/bin` il cui nome finisce per `a` e contiene una cifra

## Redirezione dell'Output

- ▶ La maggior parte dei comandi manda il proprio output ad un dispositivo chiamato **standard output**. Per default questo è il **monitor**.
- ▶ Lo standard output può essere **rediretto** verso altri dispositivi (es: file, stampante, ecc), attraverso l'uso del simbolo **>**

## Redirezione dell'Output

- ▶ La maggior parte dei comandi manda il proprio output ad un dispositivo chiamato **standard output**. Per default questo è il **monitor**.
- ▶ Lo standard output può essere **rediretto** verso altri dispositivi (es: file, stampante, ecc), attraverso l'uso del simbolo >

### Esercizio

## Redirezione dell'Output

- ▶ La maggior parte dei comandi manda il proprio output ad un dispositivo chiamato **standard output**. Per default questo è il **monitor**.
- ▶ Lo standard output può essere **rediretto** verso altri dispositivi (es: file, stampante, ecc), attraverso l'uso del simbolo `>`

### Esercizio

- ▶ Cosa ottenete digitando il comando

```
ls -l /usr/bin ?
```

- ▶ Ora digitate il comando

```
ls -l /usr/bin > lista_comandi.txt
```

Di che tipo è il file `lista_comandi.txt` ? Cosa contiene questo file?

- ▶ Ora digitate il comando

```
ls -lr /usr/bin > lista_comandi.txt
```

Cosa è successo al file `lista_comandi.txt` ?

## Redirezione dell'Output

- ▶ La maggior parte dei comandi manda il proprio output ad un dispositivo chiamato **standard output**. Per default questo è il **monitor**.
- ▶ Lo standard output può essere **rediretto** verso altri dispositivi (es: file, stampante, ecc), attraverso l'uso del simbolo **>**

### Esercizio

- ▶ Il comando **sort** ordina le righe di un testo dato in input.
- ▶ Digitate **sort** seguito da **INVIO** , poi scrivete 5 righe contenente ciascuna una frase a scelta. Per indicare la fine dell'input **CTRL + d** . Cosa succede?
- ▶ Ripetete il comando precedente redirigendo l'output in modo da salvarlo in un file **frasi.txt**

## Redirezione dell'Input

Analogamente, la maggiorparte dei comandi riceve il proprio input da un dispositivo chiamato **standard input**. Per default questo è la **tastiera**. Lo standard input può essere **rediretto** verso altri dispositivi (es: file), attraverso l'uso del simbolo **<**

# Redirezione dell'Input

Analogamente, la maggiorparte dei comandi riceve il proprio input da un dispositivo chiamato **standard input**. Per default questo è la **tastiera**. Lo standard input può essere **rediretto** verso altri dispositivi (es: file), attraverso l'uso del simbolo `<`

## Esercizio

- ▶ Il comando `wc` conta il numero di caratteri, parole e righe dell'input. Cosa si ottiene con i comando `wc < frasi.txt` ?
- ▶ Visualizzate le righe del file `frasi.txt` in ordine inverso usando il comando `sort` con le opportune opzioni.
- ▶ Ordinate le righe del file `frasi.txt` in ordine inverso, salvando il risultato in un file chiamato `frasi_ordinate.txt`

## La pipe

- ▶ È possibile collegare comandi tra loro usando la **pipe**, denotata dal simbolo `|`
- ▶ Attraverso la pipe, lo standard output di un comando diventa lo standard input di un altro comando
- ▶ Ad esempio, il comando `ls -l | less` invoca `ls` con l'opzione `-l`, e dà il risultato in pasto al comando `less`. In questo modo è possibile *scrollare* l'output e fare ricerche con `/n`.

# La pipe

- ▶ È possibile collegare comandi tra loro usando la **pipe**, denotata dal simbolo `|`
- ▶ Attraverso la pipe, lo standard output di un comando diventa lo standard input di un altro comando
- ▶ Ad esempio, il comando `ls -l | less` invoca `ls` con l'opzione `-l`, e dà il risultato in pasto al comando `less`. In questo modo è possibile *scrollare* l'output e fare ricerche con `/n`.

## Esercizio

- ▶ Salvate la homepage del nostro corso (file html) nella dir `/home/nome_utente/laboratorio_algoritmi/intro_linux`. Calcolate il numero di righe del file che contengono la parola *algoritmi*. Vi sarà utile il comando `grep`
- ▶ Calcolate il numero di file in `/usr/bin` che iniziano con due lettere seguite da una `a` e che terminano con `x`.

# Lanciare programmi grafici da linea di comando

- ▶ Provate a digitare il comando `mozilla-firefox` . Cosa succede al prompt?
- ▶ Per interrompere il processo è possibile usare `CTRL + C` oppure chiudere la finestra. Provate entrambe le vie... cosa succede al prompt?
- ▶ Una volta lanciato un programma è comunque possibile sospendere un processo digitando `CTRL + Z` . Provate a lanciare `acroread` e poi sospendere il processo
- ▶ Si può riattivare un processo mettendolo in background con il comando `bg` . Questo permette di avere il prompt nuovamente disponibile.
- ▶ È possibile lanciare un programma grafico mettendolo subito in background usando il simbolo `&` in modo da mantenere disponibile il prompt: provate a digitare `mozilla-firefox &`

# Controllo dei processi

- ▶ sistema multitasking: gestisce più processi in modo (apparentemente) simultaneo.
  - ▶ Processo = programma in esecuzione
  - ▶ Come controllare i processi?

# Controllo dei processi

- ▶ sistema multitasking: gestisce più processi in modo (apparentemente) simultaneo.
  - ▶ Processo = programma in esecuzione
  - ▶ Come controllare i processi?
- ▶ Per elencare i processi, si può usare il comando `ps` con varie opzioni.
- ▶ Quando un programma non risponde, è possibile forzare l'interruzione con il comando `kill`.
  - ▶ Lanciate `acoread` da linea di comando e mettetelo in background.
  - ▶ Usate il comando `ps` con le opzioni opportune per individuare l'identificativo del processo (PID) corrispondente all'istanza in esecuzione di `acoread`. Quando ci sono troppi processi è utile usare i comandi `ps` e `grep` collegati da una pipe...
  - ▶ Digitate il comando `kill` con argomento il PID trovato
  - ▶ Verificate usando nuovamente `ps` che il processo non esista più.
  - ▶ Se il processo non sparisce lo stesso, è il caso di usare l'opzione `-9` !