Introduzione a Unix

Barbara Guidi[1]

[1]Dipartimento di Informatica
Università di Pisa

Corso Informatica I - 2012/2013

Outline

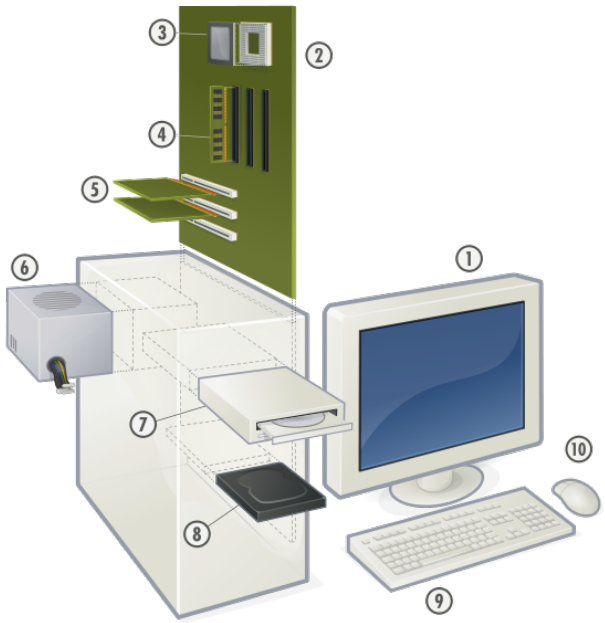
Struttura di un calcolatore

Nei calcolatori si distinguono due componenti fondamentali:

- **l'Hardware**, che e' costituito dalla parte elettronica (es. circuiti integrati) e meccanica (es. bracci che spostano le testine dell'Hard disk) del calcolatore;
- **il Software**, che e' costituito dall'insieme dei programmi che permettono di usare un calcolatore.

Architettura hardware di un calcolatore

Un computer è composto da un gran numero di elementi interagenti.



1.Video (Display) 2. Scheda madre (Motherboard) 3. CPU (Microprocessore) 4. RAM (Memoria Primaria) 5. Schede di espansione (grafica, rete, etc.) 6. Alimentatore 7. CD/DVD 8. Drive 9. HD (Memoria secondaria) 10. Tastiera Mouse

I programmi (Software)

I programmi servono per fare funzionare l'hardware (parte fisica) del computer.

Informalmente, un programma puo' essere definito come una serie di operazione elementari che il calcolatore esegue in sequenza, trasformando un insieme di dati di ingresso (INPUT) in un insieme di dati di uscita (OUTPUT).

L'insieme dei programmi (es. Word Processor, browser per Internet, etc.) e dei dati presenti sul calcolatore ne costituiscono il Software (parte logica).
In un calcolatore si distinguono vari "strati" di software:

- il sistema operativo (come Linux, Windows, Mac Os, ...)
- le applicazioni (come editori di testo, sistemi di gestione di basi di dati, giochi,...)

Il sistema operativo

Un Sistema Operativo è un particolare programma che agisce da intermediario fra utente e hardware di un computer.
Il sistema operativo (S.O.) gestisce:

- le risorse del computer (CPU, memorie, ecc.)
- le periferiche
- le applicazioni

Il sistema operativo si colloca fra l'hardware e il software applicativo permettendo ai programmi applicativi di poter interagire con le risorse hardware.

Outline

Storia di Unix (1)

Il primo sistema Unix fu sviluppato nei laboratori Bell AT&T alla fine degli anni sessanta (1 Gennaio 1970).
Unix fu progettato con le seguenti caratteristiche:

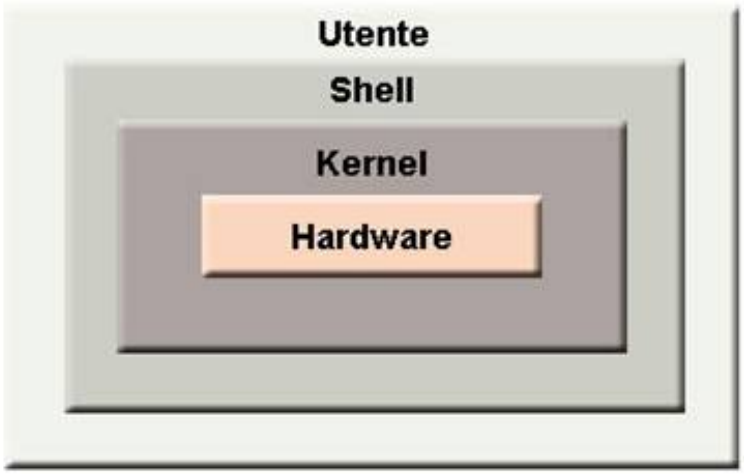
- ambiente di programmazione;
- semplice interfaccia utente;
- semplici utility che possono essere combinate per realizzare potenti funzioni;
- file system gerarchico (ad albero);
- semplice interfacciamento con i dispositivi;
- sistema multi-utente e multi-processo: più utenti possono collegarsi al sistema ed eseguire processi contemporaneamente;
- architettura indipendente e trasparente all'utente.

Storia di Unix (2)

- Nel 1973 Unix è riscritto prevalentemente in **C**, un linguaggio di programmazione ad alto livello sviluppato da Dennis Ritchie.
- Dal 1974 Unix si diffonde prevalentemente in campo accademico grazie ad una licenza stipulata con le Università per scopi educativi.
- **Come arriviamo a Linux?** Richard Stallman nel 1980 circa, iniziò a scrivere un sistema operativo chiamato GNU (Gnu's Not Unix). Nel 1991 lo studente finlandese Linus Torvalds, creò un kernel e lo chiamò Linux. Il kernel Linux venne inserito dentro GNU dando vita cosi' al sistema operativo libero GNU/Linux, piu' conosciuto come **Linux**.

Outline

UNIX in generale



- Unix è un sistema operativo a strati,
- Tutte le informazioni sono organizzate in **file,**
- Il file system è una disposizione gerarchica di file e directory (cartelle) in cui livello più alto è la **root** (o radice),
- I programmi utente interagiscono con il kernel attraverso un set di system call standard.

UNIX in generale

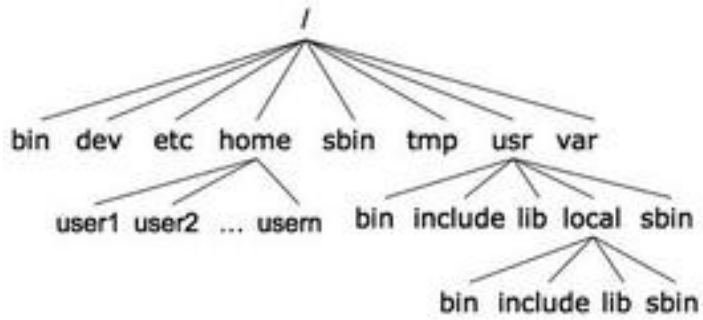
I sistemi UNIX sono costituiti di due parti principali:

- il nocciolo o kernel;
- il guscio o shell.

Il kernel è il cuore del sistema e la shell è un programma che interpreta i comandi dell'utente; detti comandi vengono acquisiti dalla shell e rigirati al kernel che ha il diretto controllo delle periferiche quali ad esempio i dischi e le stampanti.

Outline

Il file system

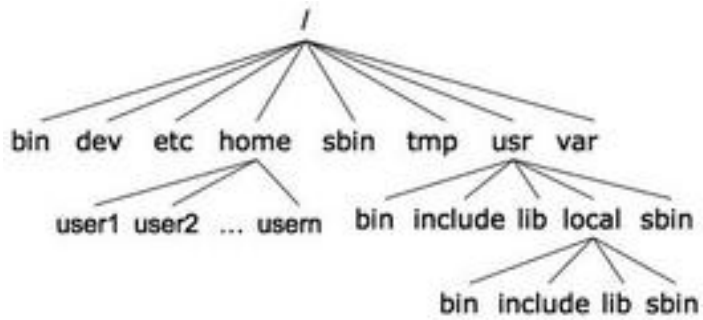


Un file system è il meccanismo fornito dal sistema operativo che regola l'organizzazione fisica e logica delle informazioni.

In Unix, il file system è paragonabile alla struttura rovesciata di un albero (in realtà è un grafo).

A differenza di Windows, non esiste il concetto di "disco". La radice del filesystem presente su un disco viene connessa all'unico filesystem presente sul sistema diventando sottodirectory di una directory presente.

Il file system



Esistono directory di sistema, che si ritrovano in tutti i sistemi Unix-like:

- **bin**: file eseguibili tipicamente da tutti gli utenti;
- **dev**: file speciali associati ai device;
- **etc**: file di configurazione;
- **home**: directory che contiene le home directory degli utenti;
- **sbin**: file eseguibili tipicamente eseguibili dall'amministratore di sistema;
- **tmp**: utilizzata per la memorizzazione di file temporanei dalla quasi totalità delle applicazioni di sistema;
- **var**: utilizzata per il logging e lo spooling.

Il file system

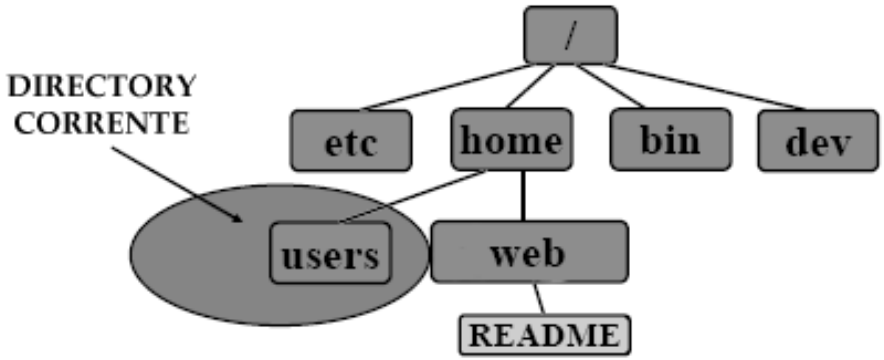
Ogni nodo dell'albero è o un file o una directory di file, dove quest'ultima può contenere altri file e directory.

- Un **file** è una sequenza non strutturata di bytes (unità logica di memorizzazione).
- Una **directory** è un file che indicizza altri file.

Un file è indicato da un **path name** ed ha i seguenti attributi: tipo, permessi (diritti di accesso), nome utente proprietario, nome gruppo proprietario, dimensione, data di creazione, ultima modifica, ultimo accesso.

Il path name di un file o di una directory può essere **assoluto**, riferito alla radice della gerarchia (/), oppure **relativo**, riferito alla posizione dell'utente nel file system.

Nomi assoluti/relativi: esempio



Nome assoluto: `/home/web/README`
Nome relativo: `../web/README`

Outline

Usare una shell

I sistemi Unix offrono diverse shell:

- **sh**: Bourne shell. La shell presente sui primi sistemi Unix.
- **bash**: Shell di default per gli utenti Linux. Estende la sh ed, in genere, gli script sh funzionano in bash. NON è vero il viceversa. È la shell di riferimento in questo corso.
- **csh**: La sintassi ricorda quella del linguaggio C. Richiesta, in alcuni casi, espressamente da programmatori.
- **tcsh**: "Turbo" csh. Estende la csh rendendola piu' "user-friendly".

Shell diverse hanno sintassi leggermente diverse. In molti sistemi Unix il file /etc/shells contiene l'elenco delle shell installate dall'amministratore e disponibili a tutti gli utenti.

Perchè usare una shell testuale

Perché usare i comandi in linea di una shell testuale?

- **Potenza e semplicità**: I comandi UNIX sono progettati per risolvere problemi specifici. Sono semplici (senza menù e opzioni nascoste) e proprio per questo potenti (es. grep parola filename).
- **Velocità e flessibilità**: è più veloce scrivere pochi caratteri da tastiera piuttosto che cercare un programma opportuno e usare le operazioni che fornisce sulla base delle proprie specifiche esigenze.
- **Accessibilità**: permette di accedere efficientemente ad un sistema in remoto.

Sintassi dei comandi UNIX

La sintassi tipica dei comandi UNIX è la seguente

comando `<opzioni>` `<argomenti>`

- ogni comando richiede al nucleo l'esecuzione di una particolare azione
- i comandi esistono nel file system come file binari, generalmente eseguibili da tutti gli utenti

`<opzioni>` sono facoltative e influiscono sul funzionamento del comando. Generalmente consistono nel simbolo del – seguito da una sola lettera.
`<argomenti>` si possono avere più argomenti o anche nessuno.

Processi in foreground e backgroud

L'esecuzione di un comando avviene creando un nuovo processo, detto processo figlio, che esegue il programma richiesto.
I processi eseguiti all'interno di una shell possono essere:

- in **foreground**: sottraggono alla shell il controllo del terminale durante la loro esecuzione;
- in **background**, il controllo del terminale rimane alla shell: il prompt appare immediatamente dopo che il processo è stato avviato;

Per default, i processi vengono lanciati in foreground. Per segnalare alla shell che si vuole lanciare il comando (o lo script) in background, è necessario farne seguire il nome dal carattere & :

```
comando [argomento..]  &
```

Ulteriori informazioni sulla shell

- Funzioni autocompletamento (tasto TAB)
- history (freccia SU/GIU)

Attenzione i filesystem UNIX-like sono case sensitive maiuscole e minuscole sono importanti.

Esempio:
file1, **File1**, **FILE1**, **FiLe1**, sono tutti file diversi

Outline

Navigare nel file system

`cd [<dir>]` Serve per muoversi attraverso le directory.

Il parametro **`<dir>`** è opzionale. Se non viene indicato, il comando porta nella home directory.

Esempio:
albero delle directory:
`/home/barbara/documenti/personali/lezione1.txt`
posizione corrente: `/home/barbara`
per portarsi nella directory dove si trova il documento, digitare il seguente comando:
`cd documenti/personali`.
Per la navigazione risultano utili le directory: **“.”** (working directory) e **“..”** (directory padre).

Visualizzare una directory

`ls [-alsFR] [<dir1> ... <dirn>]`

Alcune opzioni:

`-a` visualizza anche i file nascosti (che iniziano con ".")

`-l` visualizza informazione estesa sui file

`-s` visualizza la dimensione

`-F` aggiunge un carattere finale al nome che ne denota il tipo ("/" per directory, "*" per eseguibile, "@" per link, "=" per socket, nessuno per i file ordinari).

`-R` mostra ricorsivamente le sottodirectory Se non viene specificata nessuna directory, ls si riferisce alla directory corrente.

Creare e manipolare file

`touch <nome_file>` cambia l'orario di accesso e di modifica di <nome_file> se esiste, altrimenti lo crea (ovviamente vuoto);
`rm <f1> ... <fn>` cancella i files <f1> ... <fn>
Opzioni:

- **`rm -r <dir>`** cancella la directory con il suo contenuto
- **`rm -i`** cancella l'argomento chiedendo conferma
- **`rm -f`** cancella l'argomento senza chiedere conferma

Creare e manipolare file

`cat [opzioni] [file]` visualizza il contenuto di un file
Opzioni generali:

- **`-n`** precede ogni linea con un numero
- **`-v`** visualizza i caratteri non stampabili, eccetto tab, new-line e form-feed
- **`-e`** visualizza $ alla fine di ogni linea (prima di new-line) (quando usato con l'opzione -v)

`cat file1 file2 file3` concatena il contenuto dei files seguendo lo stesso ordine di immissione e ne mostrerà il contenuto

`more [opzioni][filename]`, **`less [opzioni][filename]`**, **`pg [opzioni][filename]`** permettono di visualizzare [filename].

Creare e rimuovere directory

`mkdir [opzioni] dir1 [dir2 ...]` serve per creare una directory
I parametri **`dir`** indicano i nomi delle directory da creare come pathname assoluti e/o relativi.
Opzioni principali:

- **`-m`** specifica i permessi d'accesso da attribuire alle directory create, con la stessa notazione usata per il comando chmod.
- **`-p`** Crea anche eventuali directory intermedie esplicitate nei parametri dir.

`rmdir [opzioni] dir1 [dir2 ...]` serve per rimuovere una directory (vuota)
I parametri **`dir`** indicano i nomi delle directory da eliminare, che debbono essere vuote.
L'unica opzione degna di nota è **-p** che rimuove anche le directory intermedie esplicitate nel parametri dir.

Comandi per la gestione di file/directory

Comando/Sintassi	Cosa fa
cp file1 file2	copia file1 in file2. file2 viene creato o sovrascritto
cp <f1>..<fn> <dir>	copia <f1>..<fn> nella directory <dir>
cp -i ...	copia chiedendo conferma prima di sovrascrivere
cp -f ...	copia senza chiedere conferma prima di sovrascrivere

Comando/Sintassi	Cosa fa
mv file1 file2	muove file1 in file2 e rimuove file1
mv <f1> ... <fn> <dir>	sposta <f1>..<fn> nella directory <dir>
mv -i ...	muove chiedendo conferma prima di sovrascrivere
mv -f ...	muove senza chiedere conferma prima di sovrascrivere

Altri comandi utili

- `pwd` (**print working directory**) visualizza il percorso assoluto della directory corrente.
- `tail [opzioni] [file]...` : mostra le ultime linee di dati provenienti da uno o più file di testo.
- `head [opzioni] [file]...` : mostra le prime linee di dati provenienti da uno o più file di testo.
- `sort <f1> ... <fn>` : mostra le righe in `<f1> ... <fn>` ordinate lessicograficamente

Esercizio 1

- A partire dalla vostra home directory, creare una cartella `temp`
- Entrare nella cartella appena creata
- Creare due sottocartelle `sorgente` e come sottolivello, `destinazione` (destinazione sarà una sottodirectory di sorgente)
- Creare nella cartella sorgente un file di nome `esempio.txt`
- Editare il file con emacs e scrivere all'interno del file la riga contenuto
- Controllare da shell il percorso assoluto della cartella corrente (sorgente) e scriverlo nel file

Esercizio 2

- Posizionatevi (se non ci siete già) all'interno della cartella sorgente
- Cancellate il file `esempio.txt` creato durante l'esercizio precedente (attenzione il file non è vuoto)
- Create un nuovo file di testo `lista1.txt` ed inserite all'interno 5 nomi di amici
- Create un nuovo file di test `lista2.txt` ed inserite all'interno 5 nomi di amici
- Muovi il file `lista1.txt` dalla cartella sorgente alla cartella destinazione
- Copia il file `lista2.txt` dalla cartella sorgente alla cartella destinazione

Esercizio 3

- Posizionatevi all'interno della cartella `destinazione`
- Visualizzate tutti i file contenuti nella directory corrente
- Concatenare i due file e visualizzare il risultato

Esercizio 4

- Posizionandovi nella vostra home directory,
- Create una nuova sottodirectory chiamata `Num_Utili` e copiateci il file di testo `Rubrica.txt` che trovate nella home directory della prof. Gori (~gorir).
- Editare il file con emacs in modo da cancellare tutte le righe che non contengono informazioni utili (es. righe vuote, righe di asterischi,....).
- Stampare a video il contenuto del file `Rubrica.txt`. Provate ad usare tutti e tre i comandi a vostra disposizione per questo, pg, more e less.
- Stampate ora a video il contenuto ordinato del file.