Programmazione I

Corso di Laurea in Informatica a.a. 2017-2018

Dr. Gabriele Tolomei

Dipartimento di Matematica Università degli Studi di Padova gtolomei@math.unipd.it



Livelli di Astrazione





Sistema Operativo

Instruction Set Architecture (ISA)

Implementazione della Macchina Fisica (CPU, memoria, porte logiche, circuiti, transistors, etc.)

Software (SW)

Interfaccia HW/SW

Hardware (HW)

Sistema Operativo



- Gestisce le risorse della macchina fisica (CPU, memoria principale, memoria secondaria, dispositivi di I/O, etc.)
- Fornisce un'interfaccia di servizi utili alle applicazioni software dei livelli superiori
- Espone un'interfaccia terminale (shell dei comandi) o grafica "a finestre" (GUI) nei sistemi in cui è prevista l'interazione con l'utente
- Tra i più noti esempi di S.O. ricordiamo:

Unix, Solaris, Linux, mac OS, Windows, ... (desktop)

Symbian OS, Android, iOS, Windows Phone, ... (mobile)

Chi è UNIX?



UNIX è il nome di una famiglia di sistemi operativi, con diverse implementazioni per le varie architetture HW.

- Multitasking e multiutente
- Ottima integrazione in rete
- Interfaccia utente modificabile
- Modularità
- File system gerarchico
- Vari strumenti di ausilio alla programmazione

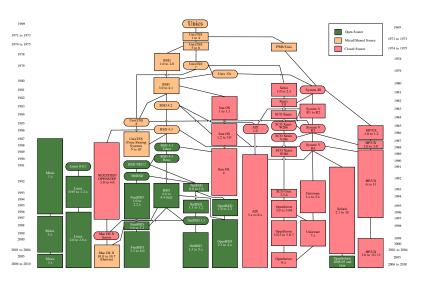
Breve storia di Unix



- nasce nel 1969 negli AT&T Bell Labs
- nel 1973 viene riscritto completamente in C
 - struttura modulare e altamente portabile
- il codice delle prime versioni di UNIX era liberamente disponibile e modificabile
- dagli anni '80 in poi le versioni diverse di UNIX si moltiplicano

Breve storia di Unix

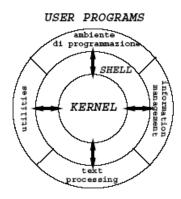




Parte 1: La shell di un S.O. Unix

La shell di un S.O. Unix





La **Shell** è dedita all'interazione con l'utente:

- l'utente impartisce i comandi digitandoli ad un apposito prompt
- il sistema mostra i risultati dell'esecuzione dei comandi, facendo poi riapparire il prompt, in modo da continuare l'interazione.

Le moderne versioni di Unix offrono in alternativa un'interfaccia grafica a finestre

Una sessione di lavoro



- Apertura di una finestra di shell
 - CTRL + T, click su icona corrispondente
- Fine di una sessione
 - CTRL + D, exit, logout (dipende dall'interprete dei comandi)

TIP: all'interno della shell i caratteri maiuscoli sono diversi dai minuscoli!

I comandi in Unix



■ Sintassi, in generale, di un comando UNIX

```
comando [-opzioni] argomenti
```

- I comandi troppo lunghi possono essere continuati sulla riga successiva battendo "\" come ultimo carattere della riga
- Si possono dare più comandi sulla stessa riga separandoli con ";" (saranno eseguiti in sequenza)

```
comando1 ; comando2 ; ...
```

Si possono dare comandi in "background" tra loro e rispetto la shell con "&"

```
comando1 & comando2 & ...
```

Help in linea



- Tutti i comandi di UNIX sono documentati in linea
 man comando
- A volte, quando vi è un conflitto di nomi, lo stesso nome può apparire in più sezioni del manuale

man N comando secondo la seguente organizzazione:

- 1 Commands
- 2 System Calls
- 3 Library Functions
- 4 Administrative Files
- 5 Miscellaneous Information
- 6 Games
- 7 I/O and Special Files
- 8 Maintenance Commands

Help in linea



Oltre a man sono disponibili altri comandi di aiuto:

- elenca le pagine del manuale contenente chiave apropos chiave
- indica le sezioni in cui si trova una pagina dedicata a comando
 what is comando

Suggerimento:

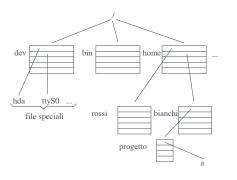
Se non vi ricordate i parametri di un comando, usate man!

Parte 2: File, percorsi e directory

I file in Unix



I file sono organizzati in una struttura gerarchia ad albero:



- / è la radice (o root) del file-system
- i nodi interni sono le directory
- le foglie sono i file

I percorsi (o path)



- Ogni file e directory è identificato da un percorso:
 - Path assoluto = /dir1/dir2/... parte dalla radice del file system
 - Path relativo = dir1/dir2/... parte dalla cartella corrente
- Percorsi speciali:
 - è la directory corrente./Documenti/prova.txt
 - .. è la directory padre di quella corrente../../dir2/file3.txt
- NOTA: i file che iniziano con . sono nascosti

Il comando 1s



■ Visualizza il contenuto di una directory

```
ls [-opzioni] file ...
```

Opzioni

- -a visualizza anche i file che iniziano con il punto
- -1 output in formato esteso
- -g include/sopprime l'indicazione del proprietario
- -t ordine per tempo di modifica del file (altrimenti si usa ordine alfabetico)
- r ordine inverso (alfabetico o temporale)
- -R elenca anche i file nelle sottodirectory

Manipolazione dei file



Copia uno o più file

```
cp [-fir] srci1 src2 ... dest
```

■ Cancella i file elencati

```
rm [-fir] file1 file2 ...
```

■ Sposta uno o più file/cambia il nome di un file

```
mv [-fi] file1 file2 ... dest
```

Opzioni

- -f non chiede mai conferma (attenzione!!!)
- -i chiede conferma per ciascun file
- r opera ricorsivamente nelle sottodirectory

Manipolazione di directory



cambia la directory in quella indicata

```
cd directory
se non si specifica la directory va nella home dell'utente
```

- mostra directory corrente pwd
- crea la directory specificata mkdir directory
- cancella una o più directory (devono essere vuote)
 rmdir dir1 dir2 ...
- cancella una o più directory (anche se piene)

Esempi



■ Elenca i file:

■ Creazione/rimozione di directory:

```
mkdir d1 rmdir d1
```

■ Copia il file f1 in f2:

```
cp f1 f2
```

Esempi



■ Sposta/rinomina il file f1 in f2:

```
mv f1 f2
```

cp f1 f2 f3 d1

cp e mv come primo argomento possono prendere una lista di file in tal caso il secondo argomento deve essere una directory:

```
copia f1, f2, f3 nella directory d1
```

Visualizzazione di file di testo



- concatena i file sul flusso di standard output cat file1 file2 ...
- visualizza le prime righe del file head [-n N] file1 file2
 - -n N visualizza le ultime N righe
- visualizza le ultime righe del file

```
tail [-n N -rf] file1 file2 ...
```

- -r visualizza in ordine inverso
- -f rilegge continuamente il file
- -n N visualizza le ultime N righe

La storia dei comandi



La storia dei comandi tiene traccia dei comandi utilizzati:

- la storia memorizza gli ultimi 500 comandi inseriti dall'utente;
- la storia viene salvata nel file .bash_history al momento del logout (e riletta al momento del login);
- il comando history consente di visualizzare la lista dei comandi:

```
$ history 4
512 ls -al
513 cd /etc
514 more passwd
515 history 4
```

ogni riga prodotta dal comando history è detta evento ed è preceduta dal numero dell'evento.

La storia dei comandi



Conoscendo il numero dell'evento che vogliamo ripetere, possiamo eseguirlo usando il metacarattere !:

```
$ !515
history 4
513 cd /etc
514 more passwd
515 history 4
516 history 4
```

■ Se l'evento è l'ultimo della lista è sufficiente usare !!:

```
$ !!
history
514 more passwd
515 history 4
516 history 4
517 history 4
```

Editing dei comandi



La shell mette a disposizione dell'utente dei semplici comandi di editing per facilitare la ripetizione degli eventi:

- utilizzando i tasti cursore:
 - con la freccia verso l'alto ↑ si scorre la storia dei comandi a ritroso (un passo alla volta) facendo apparire al prompt il comando corrispondente all'evento;
 - analogamente con la freccia verso il basso ↓ si scorre la storia nella direzione degli eventi più recenti.
 - le frecce sinistra ← e destra → consentono di spostare il cursore sulla linea di comando verso il punto che si vuole editare:
- le combinazioni di tasti Ctrl-A e Ctrl-E spostano il cursore, rispettivamente all'inizio ed alla fine della linea di comando;
- il tasto Backspace consente di cancellare il carattere alla sinistra del cursore.

Completamento automatico



Una caratteristica molto utile della shell è la sua abilità nel tentare di completare ciò che stiamo digitando:

```
$ pass<Tab>
```

cosa succede?

- La pressione del tasto <Tab> fa in modo che la shell cerchi un comando che inizi con pass.
- Siccome l'unica scelta possibile è il comando passwd, questo sarà riportato automaticamente nel prompt.

Fare una prova con:

- \$ mak<Tab>
- \$ mas<Tab>

Completamento automatico



Oltre a poter completare i comandi, la shell bash può anche completare i nomi dei file:

```
$ tail -2 /etc/p<Tab><Tab>
passwd printcap profile
$tail -2 /etc/pa<Tab><Invio>
bianchi:fjKppCZxEvouc:500:500::/home/bianchi:/bin/basrossi:Ytla4ffkGr02:501:500::/home/rossi:/bin/bash
```

- In questo caso alla prima doppia pressione del tasto <Tab>, la shell presenta tre possibili alternative.
- Digitando una a e premendo il tasto <Tab>, la shell può determinare in modo univoco il completamento del nome del file.

Parte 3: Il primo programma in C++

Hello World!



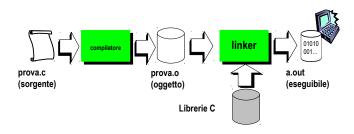
- Il nostro primo programma C++ fa una cosa semplicissima: scrive Hello World! sullo schermo
- Utilizziamo gedit per creare un file helloworld.cpp con il programma

helloworld.cpp

```
#include<iostream>
int main()
{
   std::cout << "Hello World!" << std::endl;
}</pre>
```

Il flusso di compilazione





- Il compilatore traduce il codice C++ in linguaggio macchina
- Il Linker svolge due funzioni essenziali:
 - Collegamento a librerie di codice precedentemente scritto
 - Binding degli indirizzi simbolici in indirizzi rilocabili
- In ambiente UNIX/Linux, compilazione + link realizzati da un singolo programma (compilatore C++): g++

Il compilatore C++



- g++
 - GNU C++ Compiler
 - Non è un comando UNIX standard!
- Uso di base:
 - g++ nomefile.cpp genera un file eseguibile a.out
- Opzioni (combinabili tra loro):
 - g++ -g: genera le informazioni per il debugging
 - g++ -o nomefile: genera un eseguibile con il nome nomefile
 - g++ -c: genera solo il file .o senza fare linking
 - g++ -I directory: cerca i file da compilare anche in directory
 - q++ -lnome: fai il link con la libreria libnome.a

Il compilatore C++



Esempi:

- g++ prova.cpp

 Genera un eseguibile con il nome a.out
- ./a.outEsegue il programma di nome a.out
- g++ -o prova.x prova.cpp
 Genera un eseguibile con il nome prova.x
- ./prova.xEsegue il programma di nome prova.x
- g++ -o prova.x -g prova.cpp
 Genera prova.x con info di debugging
- g++ -c prova.cpp Genera il file prova.o, cioè non effettua il linking
- g++ -o prova.g -g -lm prova.cpp Genera un eseguibile con il nome prova.g, info di debugging e usando la libreria libm.a.