Lezione 1: Introduzione alla Shell di Unix

Laboratorio di Elementi di Architettura e Sistemi Operativi

7 Marzo 2012

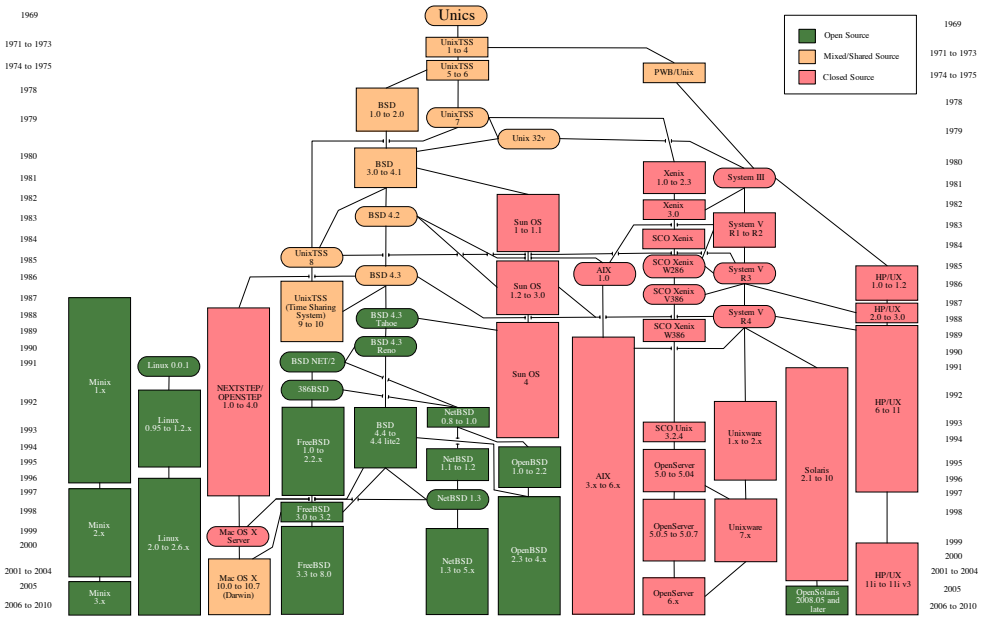
Chi è UNIX?

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

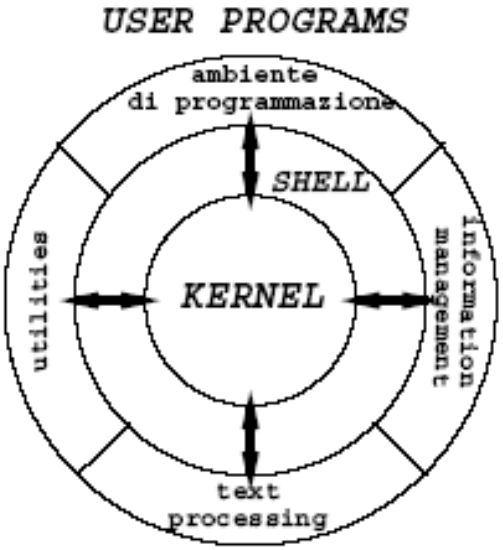


La filosofia UNIX

- UNIX è più che una famiglia di sistemi operativi:
  - Un insieme di programmi
  - Una filosofia basata su di essi
- Scopo di questa parte del corso è fornire una introduzione a questa filosofia
- Per una dettagliata descrizione dei comandi si rimanda ai manuali

Parte 1: I comandi di base

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
  - nei sistemi a finestre è sufficiente clickare sull'icona corrispondente
- Fine di una sessione
  - CTRL-d, exit, logout (dipende dall'interprete dei comandi)
- NOTA: 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 & ...
```

Il comando `ls`

- Visualizza il contenuto di una directory

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

- Opzioni
  - `-a` visualizza anche i file che iniziano con il punto
  - `-l` 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)

```
rm -r dir1 dir2 ...
```

Esempi

- Elenca i file:

```
ls
ls -l
ls -a
ls -al
ls -l /bin
```

- Creazione/rimozione di directory:

```
mkdir d1
rmdir d1
```

- Copia il file f1 in f2:

```
cp f1 f2
```

- Sposta/rinomina il file f1 in f2:

```
mv f1 f2
```

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

```
cp f1 f2 f3 d1
```

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

Help in linea

- Tutti i comandi di UNIX sono documentati in linea

  ```
  man comando
  ```

- A volte la stessa stringa si riferisce ad argomenti diversi ed occorre specificare la sezione 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

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`

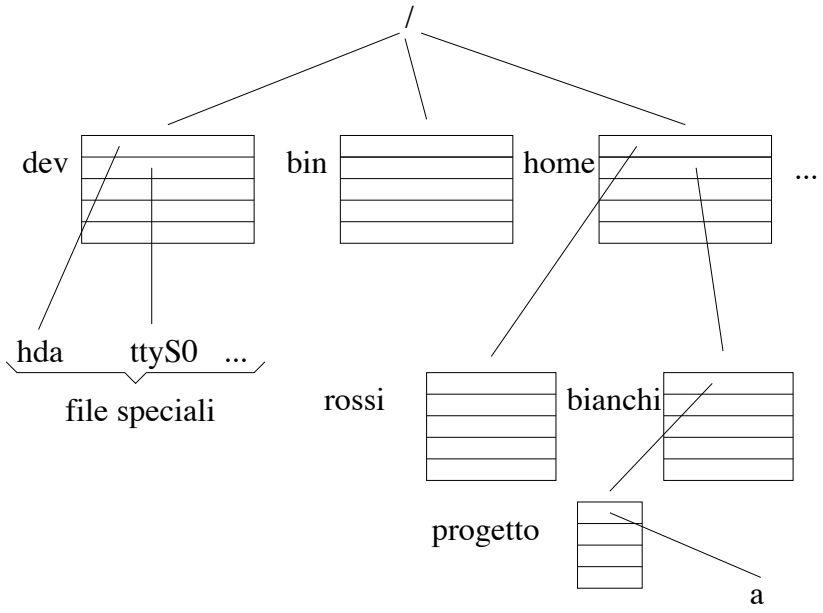
  ```
  whatis comando
  ```

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

Parte 2: File, percorsi e metacaratteri

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 identificata 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
  - `..` è la directory padre di quella corrente
- *NOTA:* i file che iniziano con `.` sono nascosti

I metacaratteri

- La shell riconosce alcuni caratteri speciali, chiamati *metacaratteri*, che possono comparire nei percorsi.
- Quando l'utente invia un comando, la shell lo scandisce alla ricerca di eventuali metacaratteri, che processa in modo speciale.
- Una volta processati tutti i metacaratteri, viene eseguito il comando.
- Questi metacaratteri (o *wildcard*) sono:

  `*` una stringa qualsiasi di 0 o più caratteri

  `?` un singolo carattere

Esempi

- `cp /JAVA/Area*.java /JAVA_backup`

  copia tutti i file il cui nome inizia con la stringa `Area` e termina con l'estensione `.java` nella directory `JAVA_backup`
- `ls *.txt`

  elenca tutti i file con estensione `.txt` presenti nella directory corrente
- `ls /dev/tty?`

  `/dev/ttya /dev/ttyb`

Tipi di file in Unix

- Un solo tipo di file fisico: *byte stream*
- Quattro tipi di file logici:

  Directory Contiene nomi e indirizzi di altri file
  Special file Oggetto per agire su un dispositivo di I/O
  Link Collegamento ad un altro file
  File ordinario Tutti gli altri file

File speciali

- Ogni dispositivo di I/O visto come un file
- I programmi non sanno se operano su file o device di I/O
- Lettura/scrittura su special file causano operazioni di I/O sul relativo device
- Indipendenza dai dispositivi reali !

I link

- **Hard link**
  - Un nome di file che punta a un i-node puntato anche da altri nomi di file
  - Non c'è differenza tra il nome originale e l'hard link: entrambi puntano allo stesso i-node !
- **Symbolic link**
  - Un file che contiene il nome di un altro file
- Particolarità
  - Non si può fare hard link di directory
  - Non si può fare hard link a file su partizioni diverse
  - Un file viene rimosso solo quando tutti i suoi hard link sono stati rimossi

Creazione e distruzione di link

- Hard link `ln fileesistente nomelink`
- Symbolic link `ln -s fileesistente nomelink2`
- Notare le dimensioni dei tre file (con `ls -al`)
- Per distruggere un link è sufficiente usare il comando `rm`
- Se il file originale viene rimosso, il link simbolico rimane pendente