Seconda Esercitazione

Introduzione alla shell Unix/Linux

Stefano Monti
stefano.monti6@unibo.it

argomenti

- introduzione
- shell
- file system
- file
- protezione
- compilatore gcc
- Hello World
- processi

Il sistema operativo Linux

introduzione

Linux

- Implementazione di Unix open source.
- Sviluppo ‘a più mani’ (internet). Prima versione: 0.02 (1991). Vers. 1.0: 1994
- S.O. multi-utente, multi-processo e multi-thread con le caratteristiche di Unix
- Portabilità: Intel x86, SUN sparc, motorola 68k, PowerPC, Alpha, MIPS.
- È disponibile gratuitamente
- Distribuzioni (assemblaggi di kernel linux + componenti): Slackware, Debian, Red Hat, ...

www.linux.org

- lista delle FAQ (Frequently Asked Questions: problematiche comuni)
- lista degli HOWTO (risoluzione di problemi particolari, per argomenti)
- Linux Documentation Project (manuali: installazione, amministrazione, ...)
- Applicazioni

Linux / Unix: la shell

utenti e gruppi, shell, comandi

Utenti e gruppi

- Sistema multiutente ⇒ problemi di privacy (possibili interferenze): necessità di proteggere le informazioni
- Concetto di gruppo (es. staff, users, students, ...): possibilità di lavorare sugli stessi documenti
- Ogni utente appartiene a un gruppo ma può far parte anche di altri a seconda delle esigenze e configurazioni

Accesso a Linux: *login*

- Per iniziare una sessione bisogna essere in possesso di un ***account*** :
  - username (es. **x135462, dll28493,** ...)
  - password (es. **dfh@2#q, **a890, aPP&x.,** ...)
- nota: maiuscole / minuscole sono caratteri diversi!! (la password ****a890** è diversa da ****A890**)
- Accesso al sistema: **login: x135462**
  password: ******

- Eseguire il login alla macchina virtuale remota tramite SSH

- senza X forwarding

```
ssh root@vmXXX.vm.labx
```

- con X forwarding

```
ssh –X root@vmXXX.vm.labx
```


shell...

- Una volta superata la fase di login, l'utente è collegato al sistema Linux. Di norma è presente una finestra di *shell*
- La shell è l'interprete del linguaggio comandi; e`un programma che consente di far interagire l'utente col sistema. Resta in attesa di comandi (da digitare con la tastiera), che manderà in esecuzione una volta ricevuto l'<ENTER>

...shell

- interfaccia di alto livello tra utente e S.O.
- processore comandi evoluto: interpreta e mette in esecuzione comandi da:
  - standard input
  - file comandi
- linguaggio comandi con elevato potere espressivo

Varie shell di Unix

- esistono diverse Shell in Unix:
  - Bourne Shell (standard)
  - C Shell
  - Korn Shell
  - Tc Shell
  - etc
- L'implementazione della Bourne shell sotto Linux si chiama *bash* (Bourne-Again shell).

uscita da una shell

- per uscire dal ciclo di una shell di login si può:
  - usare il comando **`logout`**, oppure
  - digitare CTRL+D (carattere di end-of-file)
- una volta effettuato il logout, per riprendere a usare linux bisogna inserire nuovamente username e password (login)
- per uscire da una shell anche non di login esiste il comando **`exit`**.

1] gestione di shell (terminale):

- aprire una shell remota mediante comando SSH
- eseguire il comando "whoami“ per verificare l’identità con la quale ci si è loggati
- uscire dallo shell

2] aprire il browser web (es. mozilla firefox) su macchina virtuale con esportazione dell’interfaccia grafica

- aprire una shell remota mediante comando SSH
- lanciare il browser
- caricare la pagina del corso

http://lia.deis.unibo.it/Courses/sola0809-info/

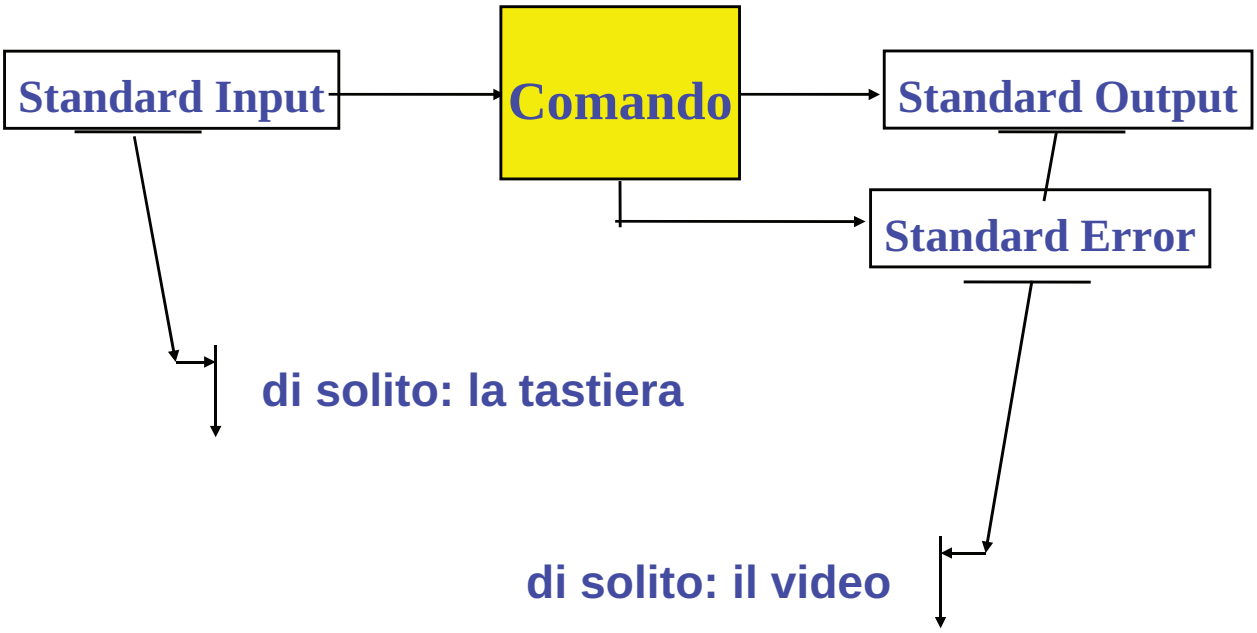
Comandi della shell di Unix

standard input, output, error;
tipi di comandi

Lo Shell di Unix:
Comandi

- 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 (direttorio **`/bin`**)
- possibilità di realizzare nuovi comandi: programmazione in shell

input / output di un comando



alcuni tipi di comandi

- interazione con il file system:
  - gestione di file e direttori
- gestione del sistema:
  - informazioni sulle risorse
  - modifica di dati di sistema

esempi di comandi

```
pippo@lab3-linux:~$ date
Mon Apr 27 21:48:24 CEST 2009

pippo@lab3-linux:~$ who (connected users info)
root    pst/3     Apr 9 14:02
root    pst/4     Apr 22 17:11 (:0.0)
paolo   pst/12    Apr 27 12:21 (deis32...

pippo@lab3-linux:~$ whoami
paolo     pst/12  Apr 27 12:21 (deis32...
```


File System

struttura logica del file system: tipi di file, persorsi assoluti e relativi, comando cd

file

- logicamente, un file è una sequenza di bit, a cui viene dato un nome
- in pratica, il file è una astrazione molto potente che consente di trattare allo stesso modo entità fisicamente diverse come: file di testo, dischi rigidi, stampanti, direttori, soft link, la tastiera, il video, etc.

tipi di file

- ordinari: archivi di dati, testi, comandi, programmi sorgente, eseguibili, ...
- directory: file gestiti direttamente solo dal S.O., che contengono riferimenti ad altri file
- speciali: dispositivi hardware, memoria centrale, hard disk, ...
- FIFO (pipe): file per la comunicazione tra processi
- soft link: riferimenti (puntatori) ad altri file o direttori

file: nomi

- È possibile nominare un file con una qualsiasi sequenza di caratteri (max. 255), a eccezione di '.' e '..' (sono nomi che hanno un significato particolare)
- È sconsigliabile utilizzare per il nome di file dei caratteri speciali, ad es. metacaratteri e segni di punteggiatura

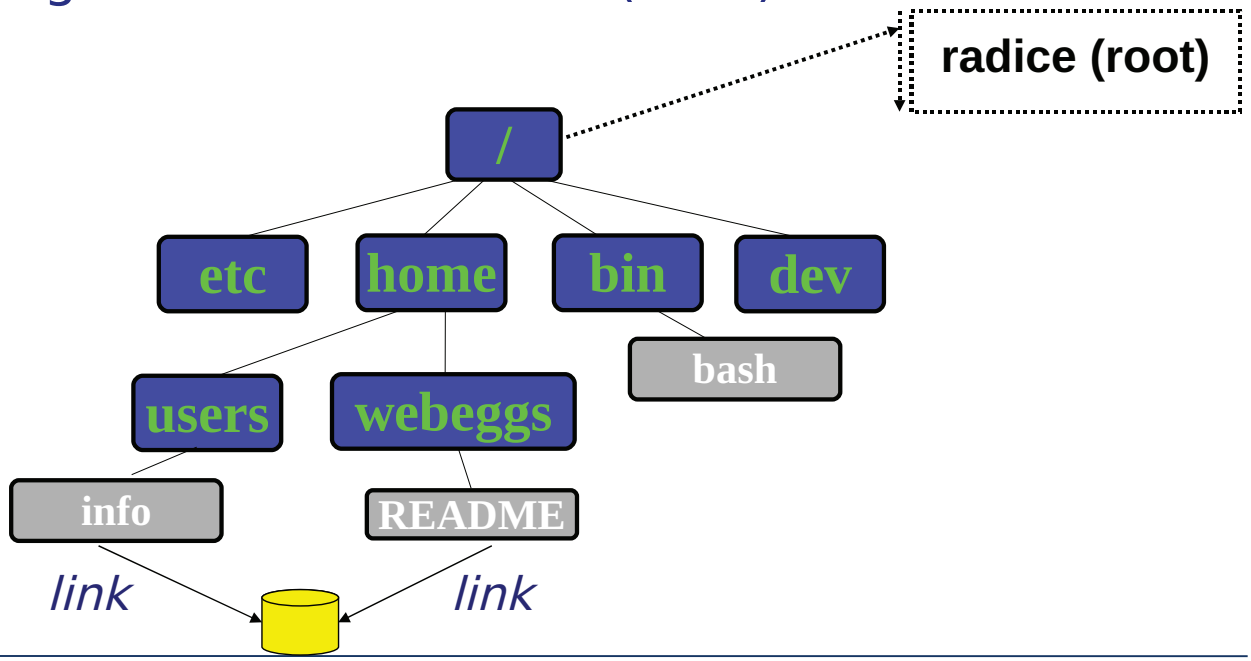
- ad ogni file possono essere associati **uno o più** nomi simbolici (link)

ma

ad ogni file è associato **uno ed un solo descrittore** (i-node) identificato da un intero (i-number)

direttori (directory)

- Il file system è organizzato come un grafo diretto aciclico (DAG).



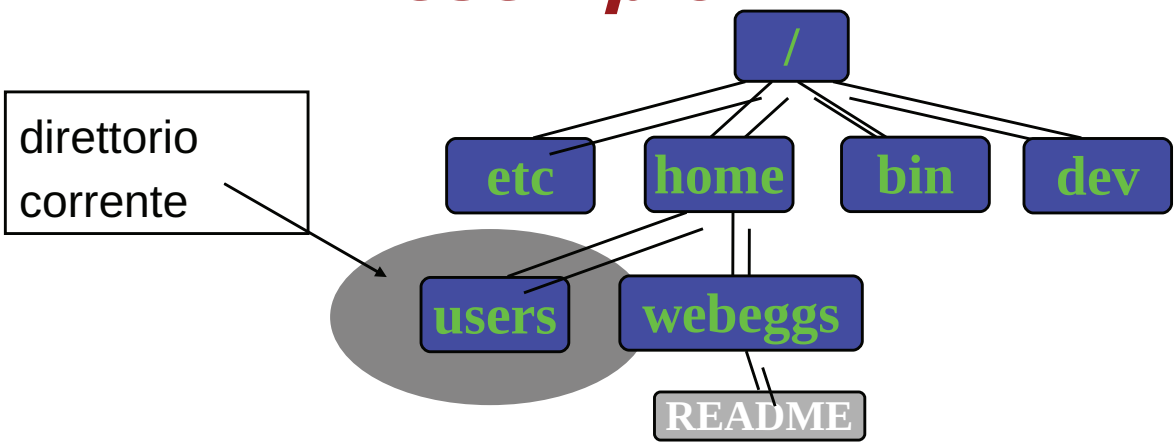
gerarchie di direttori

- all'atto del login, l'utente può cominciare a operare all'interno di uno specifico direttorio (la sua home). In seguito è possibile cambiare direttorio.
- È possibile visualizzare il percorso completo attraverso il comando **pwd** (print working directory)
- Essendo i file organizzati in gerarchie di direttori, il sistema operativo mette a disposizione dei comandi per muoversi all'interno di essi

nomi relativi / nomi assoluti

- ogni utente può specificare un file attraverso:
  - nome relativo: è riferito alla posizione dell'utente nel file system (direttorio corrente)
  - nome assoluto: è riferito alla radice della gerarchia (/)
- nomi particolari
  - **.** è il direttorio corrente (visualizzato da pwd)
  - **..** è il direttorio 'padre'

nomi relativi / assoluti: *esempio*



nome assoluto: **/home/webeggs/README**

nome relativo: **../webeggs/README**

file

concetto di file, comando ls, metacaratteri

file

- file = insieme (possibilmente vuoto) di byte organizzati in sequenza e identificato da un nome
- creazione di un file vuoto: **> *nome_file***
- esempio:
  pippo@lab3-linux:~$ > f1.txt

gestione dei file: comando ls

- consente di visualizzare nomi di file
- varie opzioni: es. **ls -l** per avere più informazioni (non solo il nome del file)
- possibilità di usare metacaratteri (*wildcard*)
- Per es. se esistono i file f1, f2, f3, f4,
  - ci si può riferire ad essi scrivendo: f*,
  - oppure: f[1-4]

formato dei comandi

- in generale:

  nome –opzioni argomenti

- esempio: **`ls -l temp.txt`**
- convenzione nella sintassi dei comandi:
  - se un'opzione / argomento può essere omesso, si mette tra quadre: [opzione]
  - se due opzioni / argomenti sono mutuamente esclusivi, vengono separati da |: arg1 | arg2
  - quando un arg. può essere ripetuto n volte, si aggiungono dei puntini: arg...

opzioni del comando ls...

- *sintassi (sempl.):*

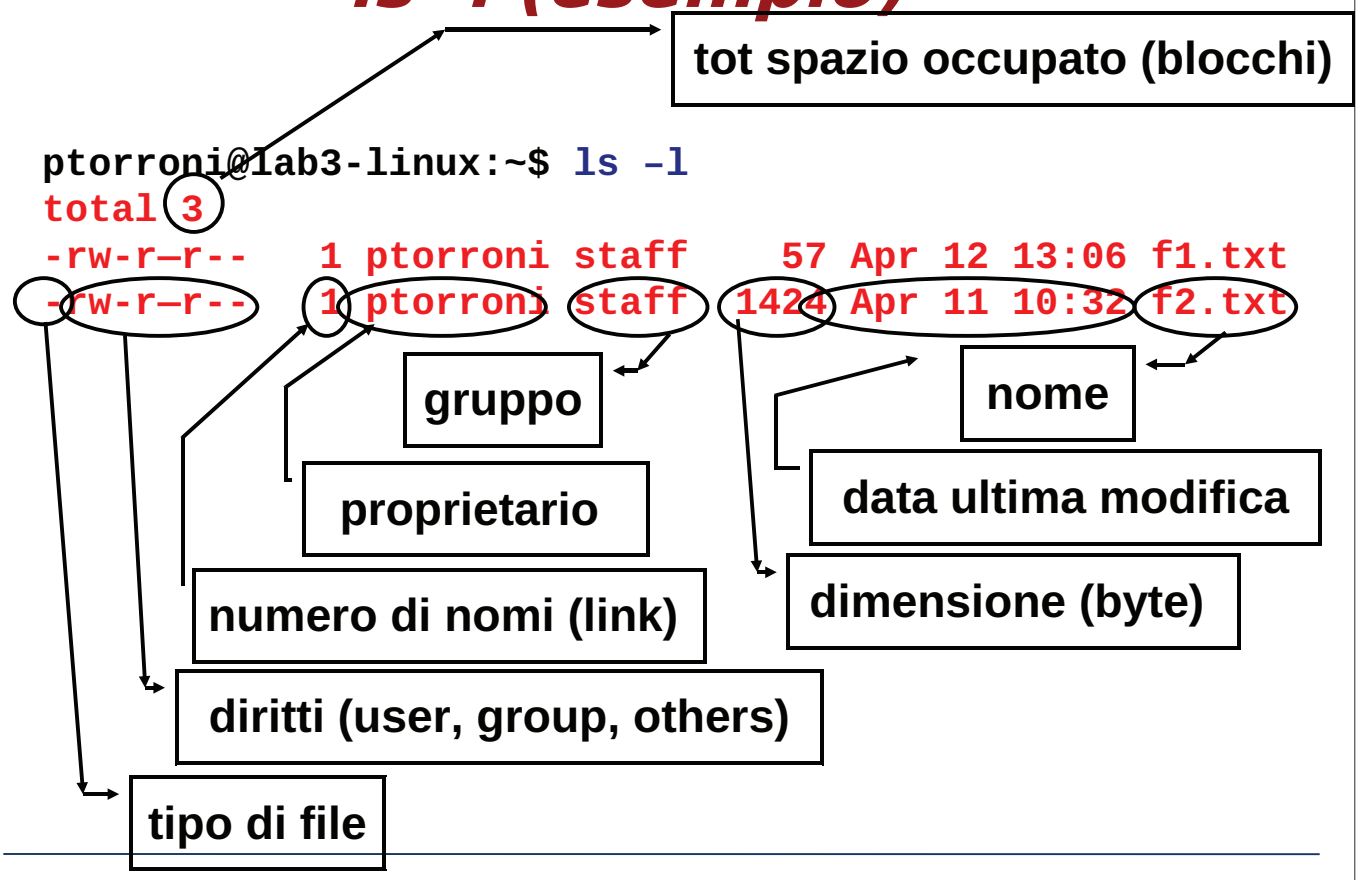
  `ls [-`*`opzioni...`*`] [`*`file...`*`]`

- opzioni:
  - l (long format): per ogni file una linea che contiene i diritti, il numero di link, il proprietario del file, il gruppo del proprietario, l'occupazione di disco (blocchi), la data e l'ora dell'ultima modifica o dell'ultimo accesso, e il nome
  - t (time): la lista è ordinata per data dell'ultima modifica

...opzioni del comando ls

- u: la lista è ordinata per data dell'ultimo accesso
- r (reverse order): inverte l'ordine
- a (all files): fornisce una lista completa (normalmente i file che cominciano con il punto non vengono visualizzati)
- F (classify): aggiunge al termine del nome del file un carattere che ne indica il tipo (eseguibile: *, direttorio: /, link simbolico: @, FIFO: |, socket: =, niente per file regolari)

ls -l (esempio)



comandi, opzioni ??

- esiste un manuale on-line (**man**), che si può consultare ogni volta che si hanno dubbi su un comando Linux. Fornisce le seguenti informazioni:
  - formato del comando (input)
  - risultato atteso (output)
  - descrizione delle opzioni
  - possibili restrizioni
  - file di sistema interessati dal comando
  - comandi correlati
  - eventuali difetti (bugs)
- per uscire dal manuale, digitare ‘q’ (quit)

✎ 1] visualizzare il direttorio corrente con il comando pwd

2] provare il comando ls:

- ls
- $ ls –l
- ls –la
- ls a*
- ...

3] provare il comando man:

- man
- man ls
- man man
- ...

il comando passwd

- È possibile cambiare la propria password di utente, mediante il comando **`passwd`**
- Verrà prima chiesta la vecchia password (per motivi di sicurezza)
- Se ci si dimentica della password, bisogna chiedere all'amministratore di sistema (utente *root*)

Gestione della password:

- modificare la password con il comando passwd.
- ripristinare la vecchia password

protezione

proprietà, accessi, bit di protezione

proprietà di file

- Come abbiamo visto, a ciascun utente viene assegnato uno username e una password, e gli utenti sono classificati in gruppi. Es:

  Username: **anna** [User-id: 1530]

  Group: **staff** [Group-id: 22]

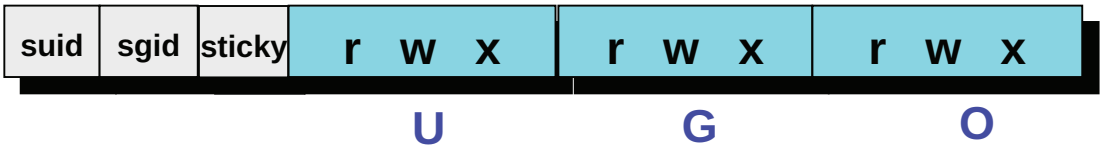
- ad ogni file è associato lo username ed il gruppo dell'utente **proprietario** (inizialmente, chi lo crea)
- In Unix è possibile **cambiare la proprietà di un file** (assegnandola a un altro utente / gruppo): comandi **chown, chgrp**

accesso ai file

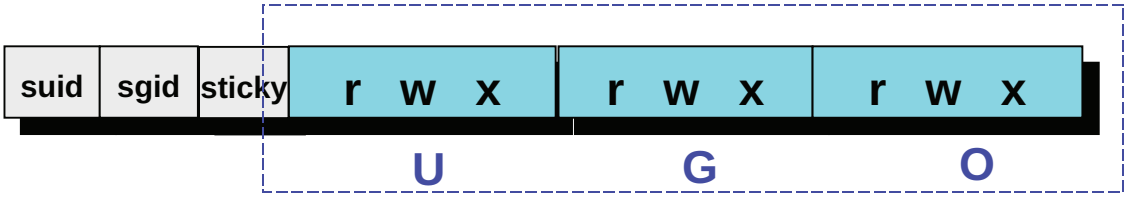
- esistono tre modalità di accesso ai file: **lettura, scrittura, esecuzione**
- il proprietario può concedere o negare agli altri utenti il permesso di accedere ai propri file
- esiste un utente privilegiato (**`root`**) che ha accesso incondizionato ad ogni file del sistema

bit di protezione

- Ad ogni file sono associati 12 bit di protezione:



Bit di Protezione:
lettura, scrittura, esecuzione

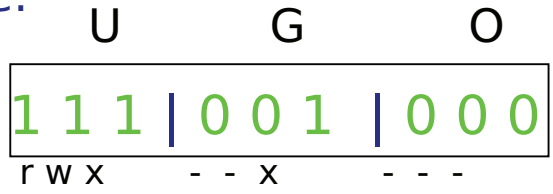


9 bit di lettura (read),scrittura (write), esecuzione(execute) per:

- utente proprietario (**U**ser)
- utenti del gruppo (**G**roup)
- tutti gli altri utenti (**O**thers)

bit di protezione:
lettura, scrittura, esecuzione

Ad esempio, il file:



- è leggibile, scrivibile, eseguibile per il proprietario
- è solo eseguibile per gli utenti dello stesso gruppo
- nessun tipo di modalità per gli altri

- formato ottale: 111 => 7; 001 => 1; ... -rwx--x--- => 0710

comandi per la gestione del file system

cd, rm, cp, cat, mv, mkdir, rmdir, chmod, chgrp, chown

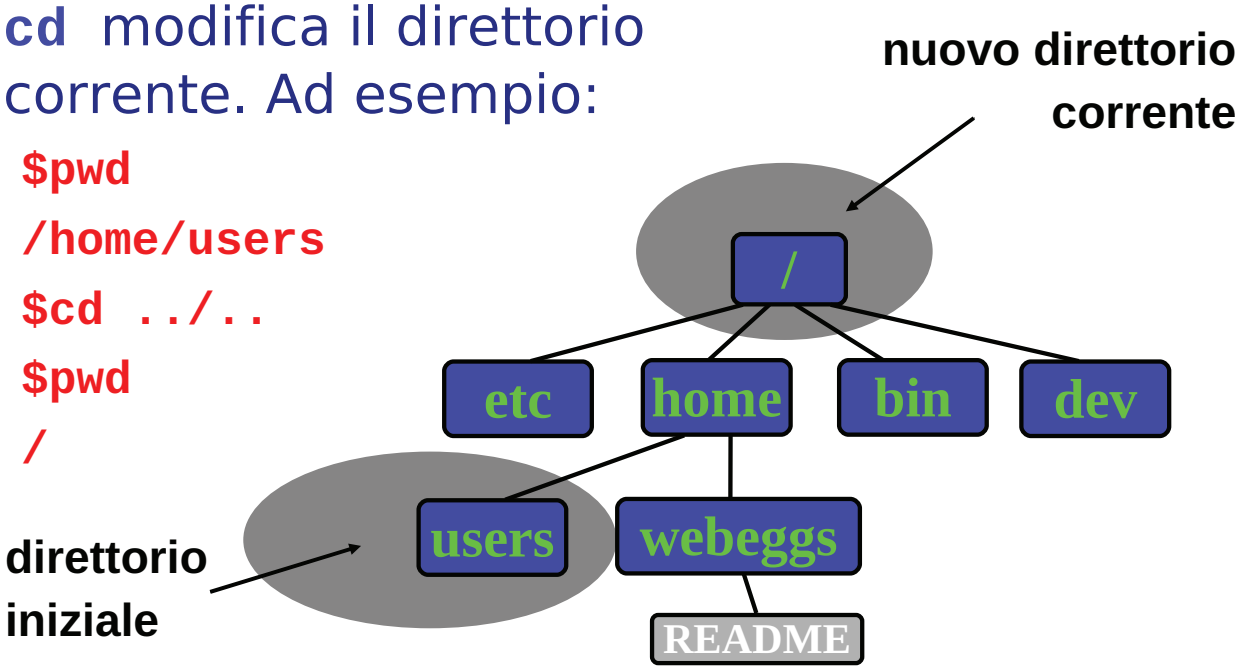
muoversi all'interno del file system

- Unix consente di 'navigare' la gerarchia di direttori costituita dal file system.
- Abbiamo già visto il comando **pwd**, che consente di visualizzare il direttorio in cui 'ci si trova'. È possibile 'spostarsi' da un direttorio a un altro attraverso il comando **cd**
- Es: **cd ..**

il comando cd...

- **cd** modifica il direttorio corrente. Ad esempio:

```
$pwd
/home/users
$cd ../..
$pwd
/
```



...il comando cd

la sintassi è:

```
cd [<nuovo direttorio>]
```

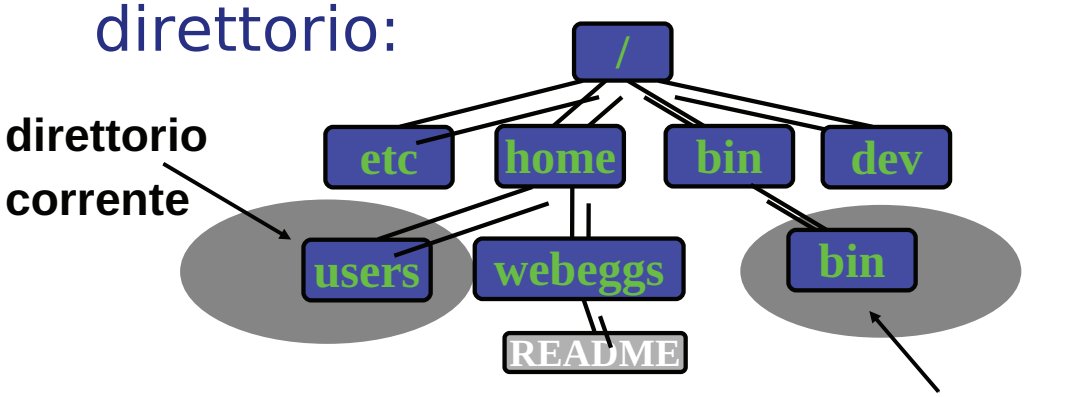
- il direttorio destinazione si può esprimere con il nome relativo oppure assoluto
- se l'argomento non viene specificato, il nuovo direttorio è la home directory dell'utente
- per spostarsi all'interno di un determinato direttorio bisogna avere per tale direttorio i diritti di esecuzione

modifica del file system: direttori

- creazione di un direttorio: **`mkdir <nomedir>`**
- eliminazione di un direttorio : **`rmdir <nomedir>`**
- per creare un direttorio è necessario avere i diritti di scrittura nel direttorio all'interno del quale lo si vuole inserire
- per eliminare un direttorio è necessario avere i diritti di scrittura di tale direttorio

esempio di mkdir

- `mkdir` crea un nuovo direttorio:




```
$mkdir /bin/bin

$mkdir ../../bin/bin
```

esplorazione ricorsiva

- Il parametro `-R` del comando `ls` specifica l'applicazione del comando stesso a tutti i direttori di un sottoalbero: `ls -R`
- I soft link non vengono inclusi nella ricorsione (non sono direttori): viene visualizzato il nome del soft link, e anche se si tratta di un link a direttorio il contenuto di questo direttorio non viene esplorato

lettura di file di testo

- è necessario avere i diritti di lettura per visualizzare il contenuto di un file di testo
- **`cat [<nomefile>...]`:** visualizza l'intero file
- **`more [<nomefile>...]`:** visualizza per videate
- altri comandi:
  - **`grep <stringa> [<nomefile>...]`** (ricerca di una stringa in un file),
  - **`wc [-lwc] [<nomefile>...]`** (conteggio di righe / parole / caratteri)

esempi di lettura file di testo

```
anna@lab3-linux:~$ cat quasimodo
ciascuno sta solo sul cuor della terra
trafitto da un raggio di sole:
ed è subito sera

anna@lab3-linux:~$ wc quasimodo
   3  17 87 quasimodo

anna@lab3-linux:~$ grep sera quasimodo
ed è subito sera
```


cancellazione, copia e spostamento di file

- è necessario avere i diritti di scrittura sul file per modificarlo o eliminarlo
- eliminazione di un file:

  `rm <nomefile>`
- copia di un file (e diritti):

  `cp <nomefile> <nuovofile>`
- spostamento di un file (e diritti):

  `mv <nomefile> <nuovofile>`

modifica dei bit di protezione

- il proprietario può modificare la proprietà di un file attraverso i comandi **chown / chgrp**
- è possibile cambiare i permessi dei propri file attraverso il comando chmod:

```
chmod mode <nomefile>
```

mode: **[ugoa][[+-=][rwxXstugoa...]...][,...]**

oppure: formato ottale dei bit di protezione

esempio chmod

```
$ ls -l file1.c
-rw-rw—r--  1   anna   staff ... file1.c

$ chmod 0666 file1.c
$ ls -l file1.c
-rw-rw-rw-  1   anna   staff ... file1.c

$ chmod a-w,u=rw file1.c
$ ls -l file1.c
-rw-r--r--  1   anna   staff ... file1.c
```

esempi modifica file (diritti)

```
ptorroni@lab3-linux:~$ ls –l quasimodo
-rw-r-—r-- 1 ptorroni staff ... quasimodo
ptorroni@lab3-linux:~$ chmod 0400 quasimodo
ptorroni@lab3-linux:~$ mv quasimodo subito
ptorroni@lab3-linux:~$ rm subito
rm: remove ‘subito’, overriding mode 0400? n
```


esempi

```
ptorroni@lab3-linux:~$ cp quasimodo sera
ptorroni@lab3-linux:~$ ls
quasimodo   sera

ptorroni@lab3-linux:~$ mv quasimodo poesia
ptorroni@lab3-linux:~$ ls
poesia    sera

ptorroni@lab3-linux:~$ rm poesia
ptorroni@lab3-linux:~$ ls
sera
```

Uso dei comandi per la gestione del file system:

- Verificare il contenuto della propria home directory
- Spostarsi nella directory superiore e visualizzarne il percorso
- Verificare i diritti di accesso associati ai file presenti
- Creare il file "pippo" nella propria **home** (ad esempio con: echo ciao! > pippo)
- leggere il contenuto del file pippo

- creare una nuova directory di nome "esercizi" all'interno della home
- entrare nella directory "esercizi"
- spostare il file "pippo" nella directory esercizi
- copiare il file "pippo" in un'altro di nome "replica"
- verificare proprietario e diritti del file "replica"
- rendere il file "replica" a sola lettura per tutti

Creazione ed esecuzione di comandi in Linux

compilatore C

file comandi (scripting)

Compilatore C: `gcc`

- File sorgente: deve avere come suffisso `.c` (Es: `HelloWorld.c`)
- compilazione di `HelloWorld.c` :
- `gcc HelloWorld.c`
- viene prodotto un eseguibile, che per default è chiamato `a.out`

gcc

- senza opzioni, gcc preprocessa, compila, assembla e linka producendo `a.out`
  - `gcc HelloWorld.c # produce a.out`
- Se voglio produrre un eseguibile che si chiami in un modo diverso da a.out, specifico l'opzione –o (output)
  - `gcc –o hw HelloWorld.c # produce hw`
- Se voglio arrestarmi dopo la produzione del modulo oggetto, specifico –c (output per default: .o)
  - `gcc –c HelloWorld.c # produce HelloWorld.o`

✎ Esercizio: Hello World! :

- utilizzare un editor per scrivere un programma C dal titolo `HelloWorld.c`, implementando il famoso programma di test.
- compilare `HelloWorld.c`, producendo l'eseguibile `hw`.
- eseguire il file `hw`

Nota: per eseguirlo, includere il path!! (es. `./hw`)

comando ps

- Un processo utente in genere viene attivato a partire da un comando. Di esso prende il nome. Ad es., dopo aver mandato in esecuzione il comando `hw`, verrà visualizzato un processo dal nome `hw`. Tramite `ps` si può vedere la lista dei processi

```
ptorroni@lab3-linux:~$ ps
  PID   TTY STAT TIME COMMAND
 4837      p2 S    0:00 -bash
 6945      p2 S    0:00 sleep 5s
 6948      p2 R    0:00 ps
```


Esecuzione in background

- I processi possono essere attivi in foreground o in background
- È possibile attivare un processo in background, tramite la seguente sintassi: `<nome_comando> &`
- È possibile chiamare in foreground un processo attualmente in background, tramite il comando `fg`
- Il CTRL+Z interrompe momentaneamente un processo; restituisce tra parentesi quadre il numero del “job” relativo al processo interrotto
- Dato un processo attualmente in foreground, è possibile:
  - interromperlo, tramite CTRL-Z
  - riprenderlo, in background o foreground (`bg / fg <job#>`)

terminazione forzata di un processo

- È possibile ‘terminare forzatamente’ un processo tramite il comando `kill`
- `kill -9 <PID>` provoca l’invio di un segnale SIGKILL al processo identificato dal PID
  - Esempio: `kill -9 6944`
- per conoscere il PID di un determinato processo, si può utilizzare il comando ps

monitor dei processi: `top`

```
  4:20pm  up 3 days,  6:48,  1 user,  load average: 0.00, 0.00, 0.00
84 processes: 83 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:  0.7% user,  0.3% system,  0.0% nice, 99.0% idle
Mem:  127840K av, 122660K used,   5180K free,  37088K shrd,  10024K buff
Swap: 130404K av,     16K used, 130388K free                100040K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
28704 ptorroni  12   0   748  748   568 R       0  0.9  0.5   0:02 top
28208 root       2   0  1084 1084   732 S       0  0.1  0.8   0:00 sshd
    1 root       0   0   432  432   356 S       0  0.0  0.3   0:01 init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kflushd
    3 root     -12 -12     0    0     0 SW<     0  0.0  0.0   0:00 kswapd
    4 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 md_thread
    5 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 md_thread
  495 root       0   0  4660 4660  1612 S       0  0.0  3.6   0:41 X
  621 root       0   0   320  320   264 S       0  0.0  0.2   0:00 mingetty
28748 ptorroni   8   0   280  280   236 S       0  0.0  0.2   0:00 sleep
   31 root       5   0   376  376   324 S       0  0.0  0.2   0:00 kerneld
  194 root       0   0   596  596   496 S       0  0.0  0.4   0:01 syslogd
  203 root       0   0   536  536   336 S       0  0.0  0.4   0:00 klogd
  214 daemon     0   0   416  416   340 S       0  0.0  0.3   0:00 atd
  225 root       1   0   484  472   440 S       0  0.0  0.3   0:00 crond
```

- background e kill :

- modificare HelloWorld.c in modo che stampi "hello world!" un numero infinito di volte.
- compilare `HelloWorld.c`, producendo l’eseguibile `hw`.
- eseguire il file `hw` in background.
- verificare la lista dei processi ed il loro stato
- terminare l'esecuzione del processo che esegue hw.