Laboratorio di Programmazione 1

Docente: Dr Damiano Macedonio

Lezione 1 – 03/10/2013

Organizzazione del Corso

- 2 Moduli:
  - Teoria: prof. Fausto Spoto
  - Laboratorio: dr. Damiano Macedonio
    tutor: dott. Alessandro Menti
    dott. Vladan Mijatovic

- Orario Lezioni
  - Giovedì 15.30 - 18.30 (Laboratorio Delta)

- Orario Ricevimento
  - Martedì 16.30 – 17.30 Aula A,
    Su appuntamento concordato via e-mail:
    damiano.macedonio@univr.it

Materiale del Corso

- Testo di riferimento:
  - *Programmare in C* (Edizione 3)
    Stephen G. Kochan, Pearson 2011
- Sito del modulo di Laboratorio:
  - http://profs.sci.univr.it/~macedonio/labProg2014.html

Argomenti del Corso e Modalità d'Esame

- Obiettivo del corso:
  - *Fornire conoscenze di base per la scrittura di programmi C*.
    - Come si organizza un programma C.
    - Relazione tra il linguaggio C e la macchina sottostante.
    - Implementazione di semplici algoritmi e strutture dati.
- Argomenti trattati:
  - Approfondimento in laboratorio degli argomenti spiegati nella parte di teoria (1/2 settimane di distanza).
- Modalità d'esame:
  - Unificato con la parte di teoria.
  - Capacità di organizzare un algoritmo e le relative strutture dati e di tradurli in linguaggio C.

Breve Introduzione a Linux

GNU: UN PO' DI STORIA…

- **GNU** è un acronimo "ricorsivo" e significa GNU's Not UNIX. Proprio perché nasce con l'idea di sviluppare un S.O. stile UNIX ma libero, che permettesse di sviluppare liberamente favorendo così la collaborazione tra programmatori.
- Il Progetto GNU (1983, Richard Stallman), si basa sulla gestione dei diritti d'autore secondo la definizione di *software libero* (contrapposta a software proprietario).
- Fulcro del Progetto GNU è la licenza *GNU General Public License* (GNU GPL), che sancisce e protegge le libertà fondamentali che, secondo Stallman, permettono l'uso e lo sviluppo collettivo e naturale del software.
- Nel 1984 inizia lo sviluppo del Sistema GNU. In realtà il *kernel* di tale sistema Hurd è tuttora in lavorazione. Ma nel 1991 Linus Torvalds scrisse il kernel Linux e lo distribuì sotto licenza GNU GPL. I sistemi GNU con kernel Linux vengono ufficialmente chiamati GNU/Linux.
- Per poter gestire alcuni casi il Progetto GNU ha creato anche la *GNU Lesser General Public License* (GNU LGPL), che permette di integrare software libero all'interno di software proprietario.

Che Cos'è Linux?

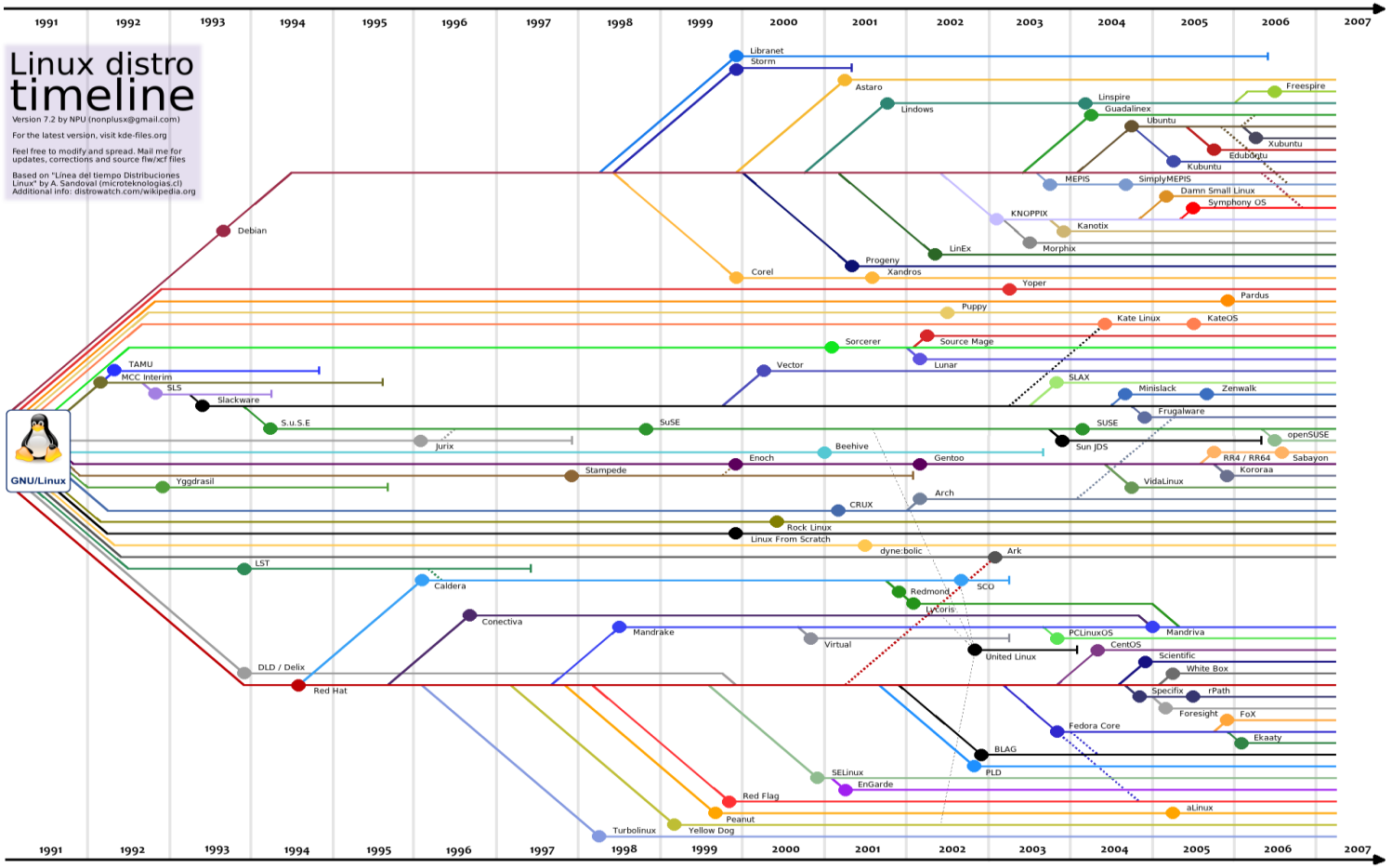
- Linux è una famiglia di sistemi operativi open-source che utilizzano un kernel comune.

- Molte importanti società hanno sviluppato o sviluppano un sistema Linux: IBM, HP, Red Hat, Novell, Google, ecc...

- Le varie versioni di Linux sono chiamate *distribuzioni*.
  - Condividono il Kernel Linux, anche se in versioni diverse e spesso personalizzate.
  - Si differenziano per i pacchetti disponibili, il sistema di gestione del software, per i servizi di manutenzione e assistenza offerti.

- Alcuni esempi:
  - Ubuntu, Fedora, Debian, Slackware, SuSE, CentOS, Red Hat Enterprise Linux, Gentoo Linux, Knoppix, Android, ecc...

Che Cos'è Linux?



Per una versione up-to-date e very impressive (!) :
http://it.wikipedia.org/wiki/Linux

UBUNTU

- Ubuntu è una distribuzione Linux basata su Debian.
  - Obiettivo principale: facilità di utilizzo.
  - Ambiente desktop grafico:
    - Fino alla versione 11.04: GNOME Panel.
    - Dalla versione 11.04: Unity.
  - Versione corrente: 12.04.

- Ubuntu può essere:
  - Installata sul computer tramite CD o dispositivo USB
    - Partizione fisica o su una virtual machine
  - Installata su un dispositivo USB
    - Installazione portabile che può essere utilizzata su qualunche computer in grado di effetture il boot da USB
  - Eseguita direttamente dal Live CD
  - Installata su una partizione Windows tramite WUBI (Windows-based UBuntu Installer)
    - Viene creata un'immagine su disco all'interno della quale viene installata Ubuntu.
    - Ubuntu vede tale immagine come una partizione reale.
    - Viene aggiunta una voce sul menu di boot per l'avvio di Ubuntu.

Il File System: Obiettivi

- Gestire in modo efficiente la memoria di massa.
- Presentare all'utente l'organizzazione logica dei dati (ad es. in file e cartelle) e le operazioni che è possibile compiere su di essi.
- Fornire all'utente e ai programmi applicativi alcuni servizi di base:
  - La creazione/cancellazione di file e cartelle
  - La manipolazione di file e cartelle esistenti
  - La copia e lo spostamento di dati su supporti diversi
  - L'associazione tra file e dispositivi di memorizzazione secondaria (memorie di massa)
  - La gestione di collegamenti (link o alias) tra file e cartelle.

    Def. Un collegamento è un riferimento ad un oggetto (file o cartella) presente nel file system.

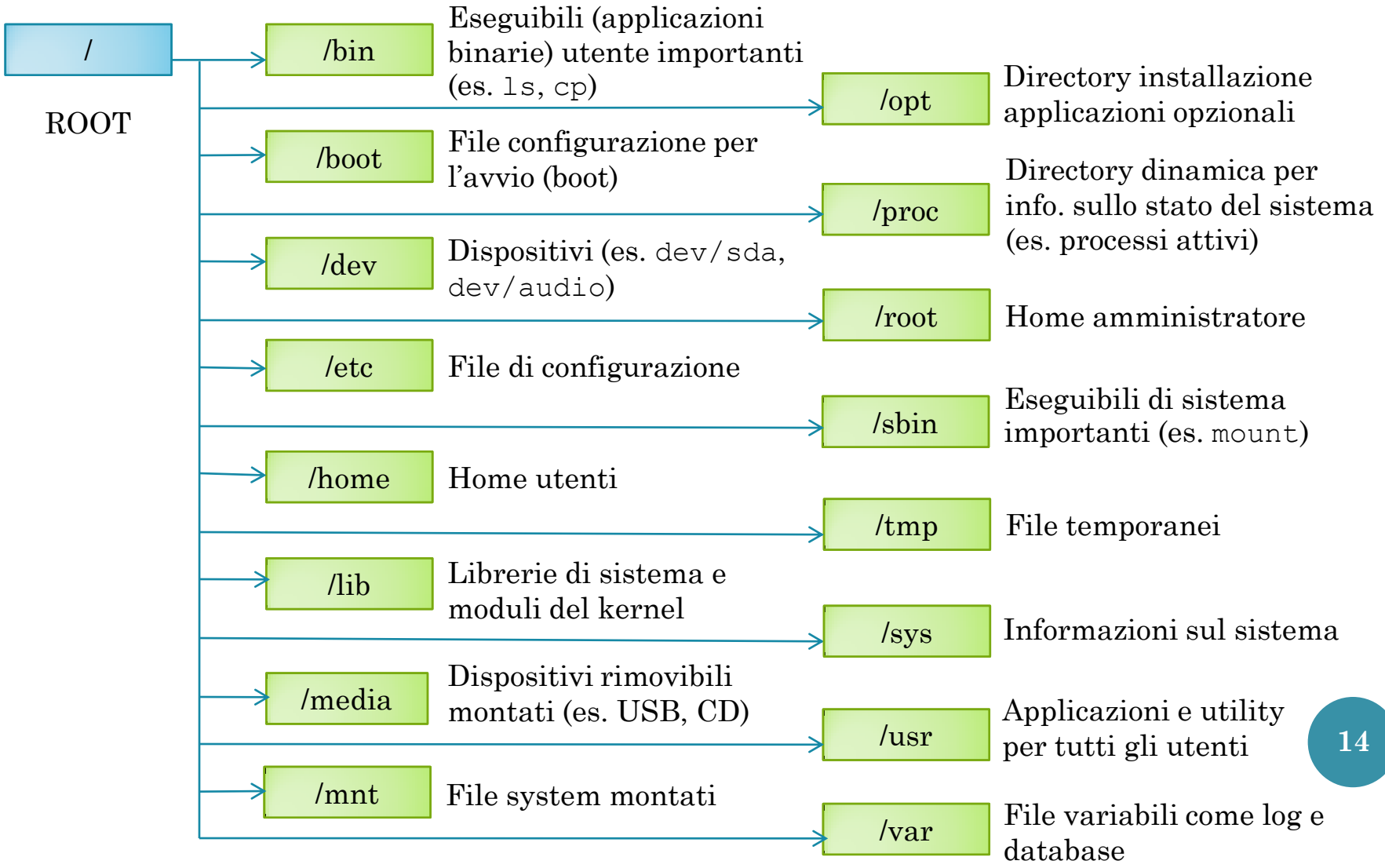
Il File System: Organizzazione

- I dati vengono organizzati in file
  - Un file è un contenitore logico di informazioni (dati o istruzioni)
  - Ogni file è identificato da un identificatore o filename (`nome.estensione`), dalla periferica (`drive`) e dal percorso (`path`) sulla periferica, da varie altre informazioni (data di creazione e di ultima modifica, dimensione, diritti di accesso al contenuto del file, ecc...)
  - I file possono essere raggruppati in più contenitori logici, cartelle o directory, e sottocartelle o sottodirectory, organizzati secondo una struttura gerarchica ad albero
  - I collegamenti (o link, alias) permettono di creare riferimenti ad altri oggetti (file e directory) nel file system. Permettono di accedere ad un oggetto da più punti dell'albero.

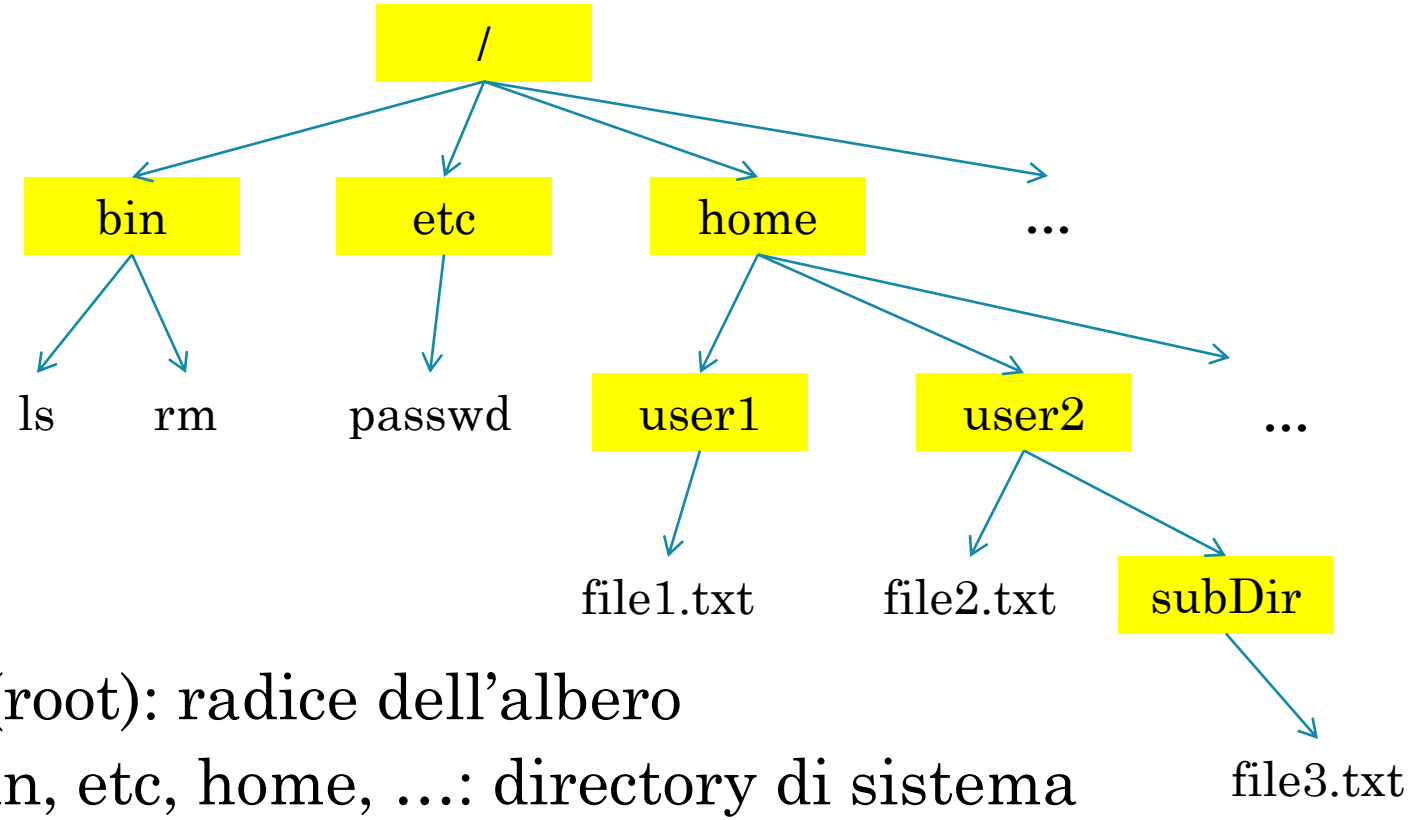
Il File System

- In Linux tutto è un file
  - Dischi, terminali, schede video, ecc…
- Unico albero in cui vengono montati tutti i dispositivi.
- Il file system ha origine nella directory /, detta root.
- Caratteri e directory speciali:
  - ~ home directory
  - . directory corrente
  - .. directory superiore

Il File System



Struttura Logica



- / (root): radice dell'albero
- bin, etc, home, …: directory di sistema
- user1, user2, …: directory e file utente
- ls, rm, passwd: eseguibili (comandi)

Pathnames (Percorsi)

- Un file è individuabile attraverso il nome e le sottodirectory del percorso dalla root /

  Esempio: `/home/user1/file1.txt`

- I cammini possono essere relativi (rispetto a directory di lavoro) o assoluti

  Esempio: cammino assoluto e cammino relativo rispetto alla directory `user2`:

  `/home/user2/subDir/file3.txt`

  `subDir/file3.txt`

La Shell Bash: Comandi

- *Bash* (Bourne-Again Shell) è un clone evoluto della shell standard di unix (/bin/sh) chiamata anche Bourne shell dal nome del suo autore originario Stephen Bourne.

- La shell è un interprete dei comandi.

```
$ nome-comando <opzioni e/o argomenti>
```

- `man <nome-comando>`: informazioni sulla sintassi, il funzionamento e le opzioni disponibili. (si naviga con le frecce **UP/DOWN** e si esce con tasto **Q**)

- Autocompletamento con tasto **TAB**.

La Shell Bash: Canali Flusso

- Esistono 3 canali di flusso principali per i processi:
  - *stdin*: canale di input.
  - *stdout*: canale di output.
  - *stderr*: canale di errore.
- I flussi possono essere rediretti su un file:
  - `< nome-file`: reindirizza *stdin*.
  - `> nome-file`: reindirizza *stdout*.
  - `2> nome-file`: reindirizza *stderr*.
  - `&> nome-file`: reindirizza sia *stdout* che *stderr*.
  - `>> nome-file`: aggiunge al file se esiste già.

```
$ echo "hello" > f.txt
$ echo "world" >> f.txt
$ cat f.txt
hello
world
```

La Shell Bash: File e Directory

- `touch <nome-file>`
  - Crea un nuovo file vuoto, oppure cambia data di accesso di un file esistente.
- `cat <nome-file>`
  - Visualizza uno o più file sullo *stdout*, utile per visualizzare file piccoli.
  - Utile per unire più file.
    - `cat file1.txt file2.txt > out.txt`
- `split <nome-file>`
  - Divide un file in pezzi più piccoli
  - `split -b N` produce pezzi da `N` byte
- `less <nome-file>`
  - Visualizzatore di file usato dal comando `man`.
- `tail <nome-file>`
  - Visualizza le ultime righe di un file, utile per visualizzare i file di log.
  - `tail -n` visualizza le ultime *n* righe.
  - `tail -f` aspetta nuovi caratteri dal file (utile se vogliamo seguire la scrittura di un file da parte di un nostro programma).

La Shell Bash: File e Directory

- `cp <nome-file-origine> <nome-file-destinazione>`
  - Copiare un file o directory.
  - `cp -R` copia ricorsiva (di una directory e le sue sottodirectory)

- `mv <nome-file-origine> <nome-file-destinazione>`
  - Sposta e/o rinomina un file.
  - `mv -i` chiede conferma prima della sovrascrittura

- `rm <nome-file>`
  - Cancella un file o una directory.
  - `rm -R` cancellazione ricorsiva delle sottodirectory
  - `rm -i` chiede conferma ad ogni file
  - `rm -f` forza la rimozione

- `pwd`
  - Stampa la directory corrente.

La Shell Bash: File e Directory

- `cd <percorso>`
  - Cambia la directory corrente.
- `mkdir <nome-dir>`
  - Crea una nuova directory.
- `ls`
  - Visualizza il contenuto di una directory.
  - `ls -l`: visualizza anche informazioni.
  - `ls -R`: mostra ricorsivamente le sottodirectory.
  - `ls -a`: mostra che file nascosti (`.nomefile`).
  - `ls -h`: dimensioni human-readable.
- `df`
  - Mostra lo spazio libero su un dispositivo.
    - `df -h` human-readable.
- `du`
  - Mostra lo spazio usato da un file.
    - `du -h` human-readable.

La Shell Bash: File e Directory

- `ln -s <source> <target>`
  - Crea link simbolico tra file, cioè un alias per raggiungerli da percorsi diversi.
  - Utile ad esempio per includere delle librerie comuni ad un progetto.

- `tar`
  - `tar -c`: creazione di un archivio (`.tar`).
  - `tar -x`: estrazione di un archivio.
  - `-f <nome-file>`: leggi/scrivi su file anzichè su *stdin/stdout*.
  - `-z`: archivio compresso (`tar.gz`).
  - `-v`: elenca file decompressi.

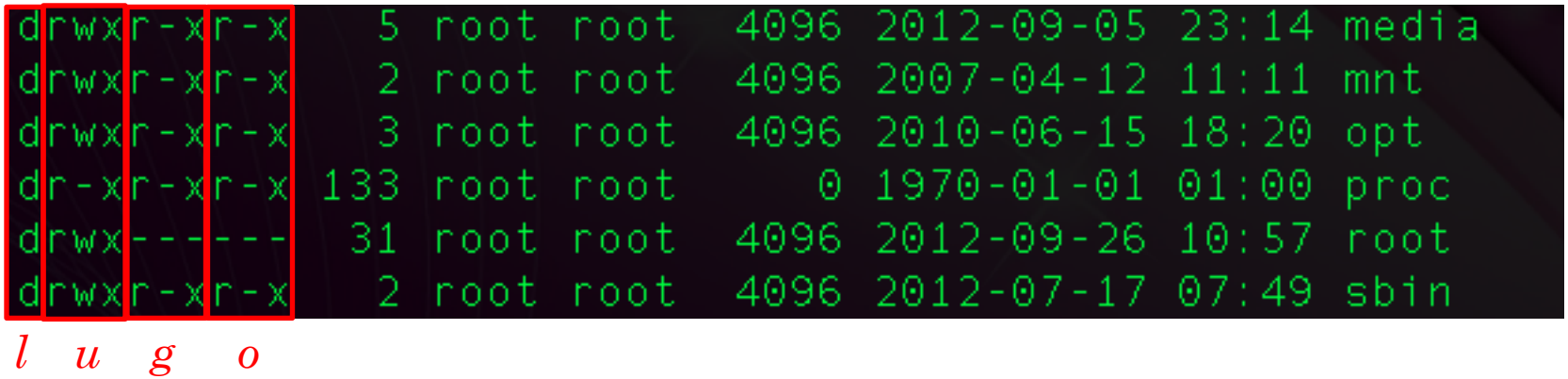
- `find`
  - Cerca un file
  - `-name <nome-file>` cerca un file per nome
  - `-iname <nome-file>` cerca un file per nome no case-sensitive
  - `-type f/d` cerca solo file/directory
  - `-size +/-dim` dimensione maggiore o minore di dim
  - `-ctime +/- num` file modificati da più/meno num giorni

La Shell Bash: Permessi su File

- Ad ogni file viene associato un:
  - *owner* – proprietario del file.
  - *group* – un gruppo di appartenenza.
- È possibile specificare permessi di lettura, scrittura ed esecuzione.
- Un permesso è rappresentato con 10 caratteri suddivisi in 4 campi:
  - *l*: tipo di file
  - *u*: permessi associati al proprietario del file.
  - *g*: permessi associati ai membri del gruppo.
  - *o*: permessi associati ad altri utenti.

La Shell Bash: Permessi su File

- Il primo carattere *(l)* specifica il tipo di file
  (es. – significa normale, `d` significa directory, `l` significa link, ecc...).
- Un permesso *u*,*g* ed *o* è formato da 3 caratteri che specificano i permessi di lettura (*r*), scrittura (*w*) ed esecuzione (*x*).
  - 1° carattere: – lettura non permessa, `r` lettura permessa.
  - 2° carattere: - scrittura non permessa, w scrittura permessa.
  - 3° carattere: - esecuzione non permessa, x esecuzione permessa.



- Per le directory il significato di *r*, *w*, *x* è:
  - *r*: è permesso leggere il contenuto della directory.
  - *w*: è permesso modificare il contenuto delle directory.
  - *x*: è permesso usare pathname che contengono la directory.

La Shell Bash: Permessi su File

- `chown <nuovo_utente> <nome_file>`
  - Cambia il proprietario di un file o di una directory.
- `chgrp <nuovo_gruppo> <nome_file>`
  - Cambia il gruppo di un file o una directory.
- `chmod <permessi> <nome_file>`
  - Cambiare i permessi di un file o di una directory.

Cambiare i Permessi: Esempi

Il comando `chmod` permette di cambiare i permessi

- con operatore di assegnamento (=)

  Esempio:

```
$ chmod u=rwx miofile
$ chmod go= miofile
$ chmod a=rx miofile
```

- con operatori di aggiunta (+) e eliminazione (–)

  Esempio:

```
$ chmod go-rx miofile
$ chmod a+rx miofile
```

n.b. “a”: all (tutti)

La Shell Bash: Caratteri Speciali

- Caratteri speciali:
  - `*` qualsiasi stringa (nulla inclusa)
  - `?` carattere qualsiasi
  - `[...]` carattere nell'intervallo
  - `[^...]` carattere non nell'intervallo
  - "" nomi con spazi

```
$ touch {g,p,od}ino
$ ls
gino pino odino
$ ls *ino
gino pino odino
$ ls ?ino
gino pino
$ ls [a-m]ino
gino
$ ls [^a-m]ino
pino
$ touch "ubuntu 12"
```

- Gli alias permettono di velocizzare la digitazione di comandi frequenti, es. aggiungendo opzioni di default.

```
$ alias ll='ls -l'
```

La Shell Bash: Variabili d'Ambiente

- A cosa servono?
  - Per impostare opzioni per la shell.
  - Per impostare opzioni predefinite per alcuni programmi.
- Visualizzare valore di una variabile:
  - `echo $<nome-var>`
- Impostare il valore di una variabile:
  - `export <nome-var>=<valore>.`

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin
$ export PATH=~/mybin:$PATH
$ echo $PATH
/home/mace/mybin:/usr/local/bin:/usr/bin:/bin
```

La Shell Bash: Processi

- Un *processo* è un programma in esecuzione.
- Ad ogni processo è associato un identificatore univoco: `PID`.
- Un processo può essere *attivo* o *sospeso*.
- Un processo può essere eseguito in:
  - *foreground*: esecuzione in primo piano che blocca la shell.
  - *background*: esecuzione in sfondo, lascia libera la shell.
    - Un programma può essere lanciato in background usando il carattere `&`.
- Mentre un processo è in esecuzione in foreground si può alterarne l'esecuzione con:
  - `CTRL-\` si chiede al processo di uscire.
  - `CTRL-C` viene ucciso il processo.
  - `CTRL-Z` il processo viene sospeso e viene restituita la shell.

La Shell Bash: Processi

- I processi sospesi o in background possono essere visualizzati con il comando `jobs`.
- Un processo sospeso può essere riattivato con i comandi:
  - `bg <number>`: viene mandato in background.
  - `fg <number>`: viene mandato in foreground.

  (dove `<number>` è l'identificativo assegnato da `jobs`) .

- I processi attivi possono essere visualizzati con il comando `ps`
  - `ps -u nome` mostra processi di `nome`
  - `ps -e` mostra tutti i processi
  - `ps ax` mostra tutti i processi con linea di comando
- Invio di un segnale ad un processo

  `kill -<signal> <pid>`.
  - es. `kill -9 <pid>` uccide il processo.

La Shell Bash: Pipeline e Sequenze

- Lo *stdout* di un processo può essere connesso allo *stdin* di un altro utilizzando il carattere | e formando una *pipeline*

```
$ ps -e | grep bash
7191 pts/3 00:00:00 bash
8582 pts/5 00:00:00 bash
8732 pts/5 00:00:00 bash
```

- I comandi possono essere combinati in sequenza:
  - `com1;com2` `com2` è eseguito dopo `com1`
  - `com1&&com2` `com2` è eseguito dopo `com1`, solo se `com1` non ha dato errori

Alcuni Riferimenti

- Sito ufficiale:
  - http://www.gnu.org/software/bash/bash.html
    http://www.gnu.org/software/bash/manual/
- The Linux Documentation Project
  - http://tldp.org/LDP/Bash-Beginners-Guide/html/index.html
  - http://tldp.org/LDP/abs/html/index.html