



Università degli Studi di Verona

Dipartimento di Biotecnologie

Laurea in Biotecnologie

Corso di Informatica 2014/2015



Linux Ubuntu e il compilatore C

Dicembre 2014 - Sergio Marin Vargas

Caratteristiche di Linux

Nel 1991, uno studente finlandese, **Linus Torvalds**, per la sua tesi di laurea scrive un "mini-**unix** per PC", basandosi su software **GNU**, e nasce così **LINUX**.



Linux possiede molte caratteristiche che ne fanno un ottimo sistema operativo:

- ✓ robusto e affidabile
- ✓ ampio supporto di processori e periferiche
- ✓ networking avanzato
- ✓ strumenti di sviluppo e debugging
- ✓ interfaccia grafica
- ✓ ampia disponibilità di software
- ✓ gratuito e open source (licenza GNU)

Distribuzioni Linux

- Principali distribuzioni
 - Debian, Ubuntu
 - RedHat, Fedora
 - Novell SuSe, Open Suse
 - Mandriva
 - Slackware
- Altre distribuzioni
 - 1000+



Ubuntu

Ubuntu è una espressione in lingua **bantu** (Nazioni dell’Africa sub-Sahariana) che indica "benevolenza verso il prossimo, si ispira in una regola di vita, basata sulla compassione e il rispetto altrui".

- Ubuntu deriva da Debian
- Ubuntu sarà sempre
 - “free to download”
 - “free to use” and
 - “free to distribute to others”
- Desktop Edition
- Server Edition
- X86-32, x86-64, and Mac
- Ubuntu, Kubuntu, Xubuntu, ...



ubuntu

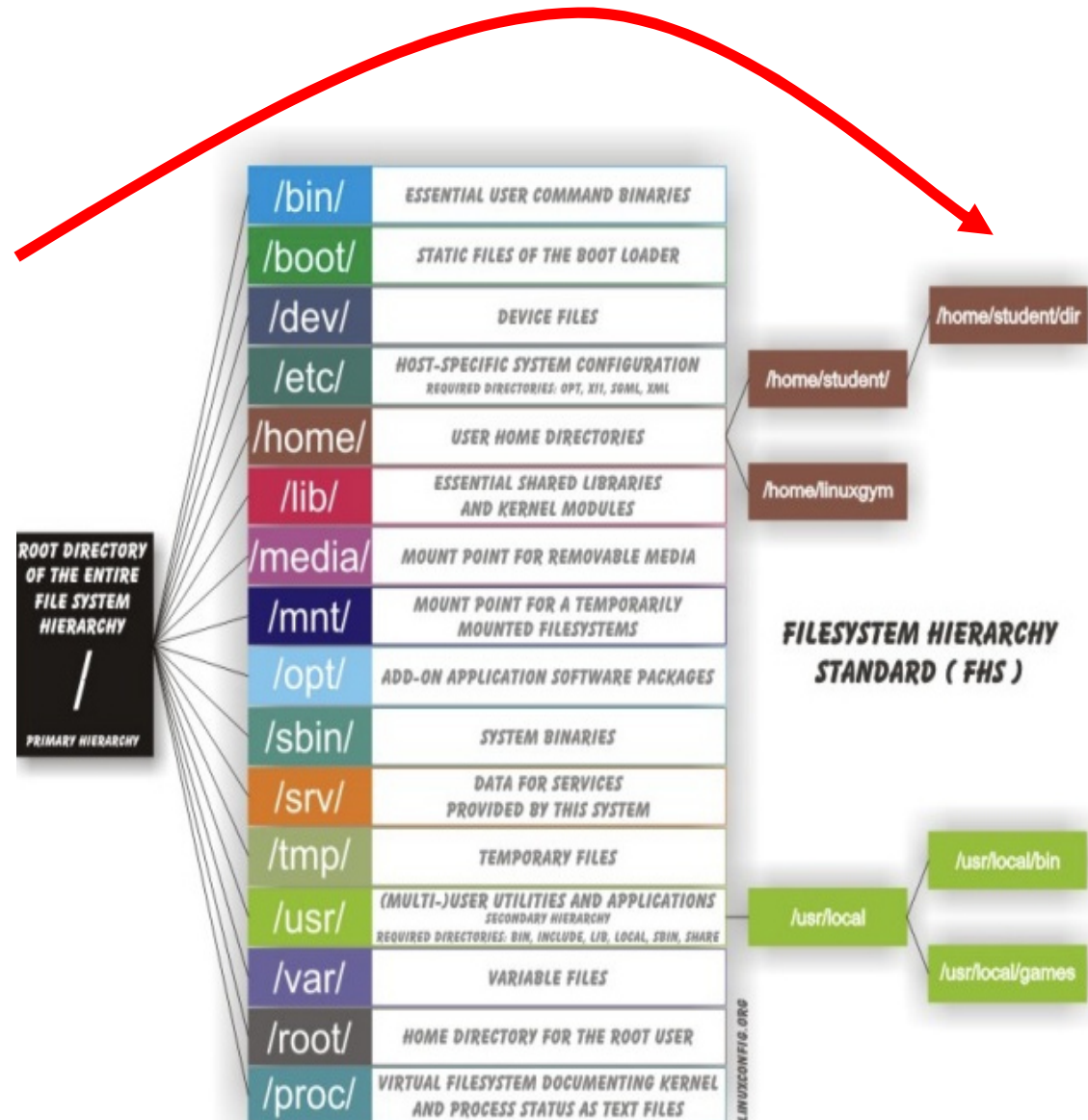
<http://www.ubuntu.com/>

Il filesystem di Linux

- Opera su diversi tipi file:
 - **normali**
Archivi di dati, testi, comandi, programmi sorgente, programmi eseguibili, immagini, ecc.
 - **directory**
Entità che raggruppa diversi files al suo interno.
 - **device**
Dispositivi hardware collegati, vengono visti come file speciali (stampanti, dischi esterni, chiavette, ecc).
 - **link**
Riferimento ad un altro file o directory. Le operazioni sul link si riflettono sull'oggetto collegato.

La vostra home dentro il filesystem

- Ora voi siete nella vostra home
- È una cartella in cui potete scrivere i vostri file, fare cartelle e mettere in ordine i vostri file di lavoro.



Struttura logica: pathnames

- Un file è individuabile attraverso il **nome** e le **sottodirectory del percorso dalla root** “/” questo percorso viene chiamato “**path**”.
- Le “**path**” possono essere **relative** (rispetto alla directory in cui si è posizionati cioè la directory corrente) o **absolute** (iniziano con “/” o “./”)

Esempio: path assoluta e path relativa (sul file “file.txt” che c’è nella cartella “/home/utente”)

- **Path assoluta**

- /home/utente/file.txt**

- (path assoluta posizionato in qualsiasi directory)

- ./file.txt**

- (path assoluta se sono posizionato su /home/utente)

- **Path relativa**

- utente/file.txt**

- (path relativa se sono posizionato su /home)

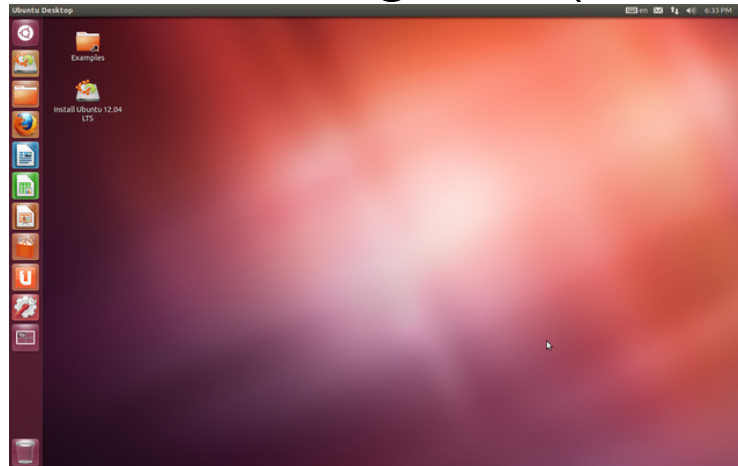
- file.txt**

- (path relativa se sono posizionato su /home/utente)

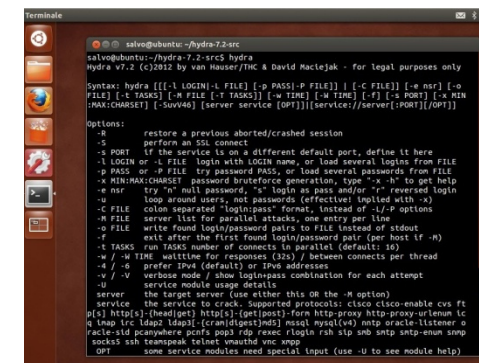
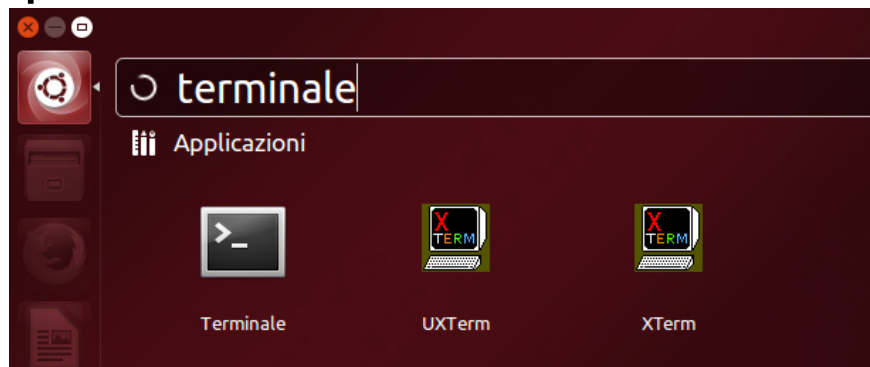
Interagire con il sistema

□ In Linux si può interagire con il sistema attraverso:

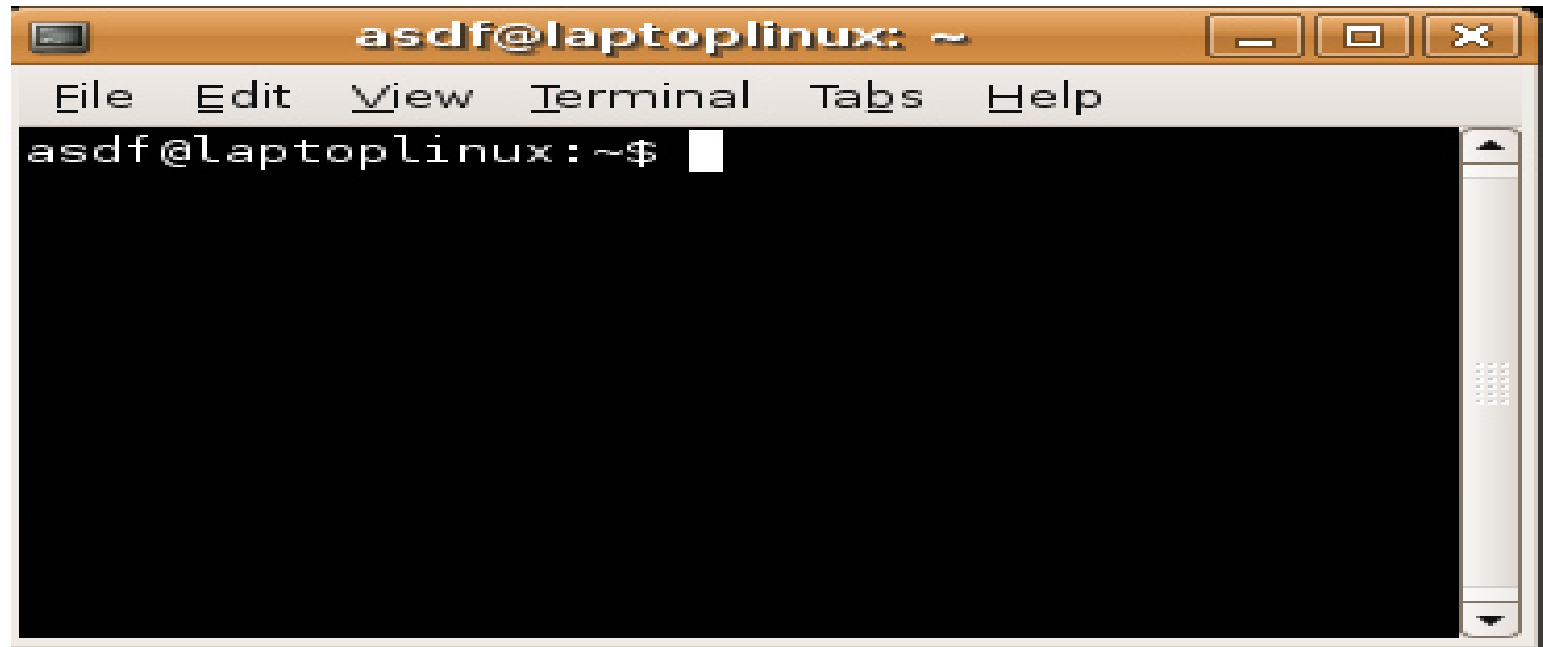
❖ Un'interfaccia grafica (simile a windows)



❖ Un'interfaccia a riga di comando chiamata **shell**, alla quale si accede attraverso il **terminale**

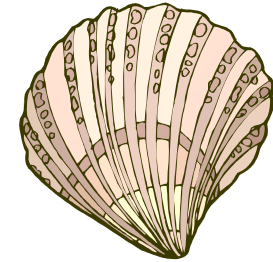


Il terminale



- ❑ Il terminale è un ambiente dove si possono scrivere i comandi e leggere le risposte.
- ❑ Dentro il terminale I vostri comandi sono interpretati dalla **shell**, cioè un programma che traduce i vostri comandi in linguaggio macchina.

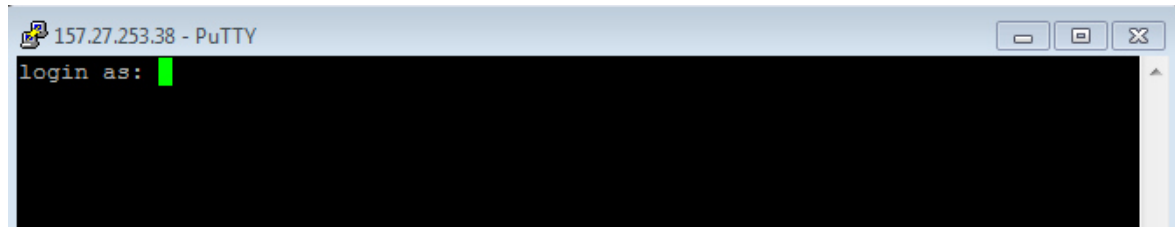
La Shell



- In Linux si può interagire con il sistema attraverso un programma chiamato **shell** il quale viene invocato automaticamente al *Login*.
- Esistono diverse *shell* (**sh**, **bash**, **tcsh**, ...)
 - la maggior parte dei comandi ha la stessa sintassi nelle diverse shell, la più utilizzata resta la “**sh**”
 - la scelta di una *shell* è essenzialmente questione di gusti
- In Linux i comandi sono dati sotto la *shell* che li interpreta, esegue e scrive (se necessario) il risultato sul terminale.
- L'attenzione della shell è espressa dal *prompt* , carattere che può essere deciso dall'utente.
- Esempi di *prompt* :

```
>  
utente@server:~$
```

Login



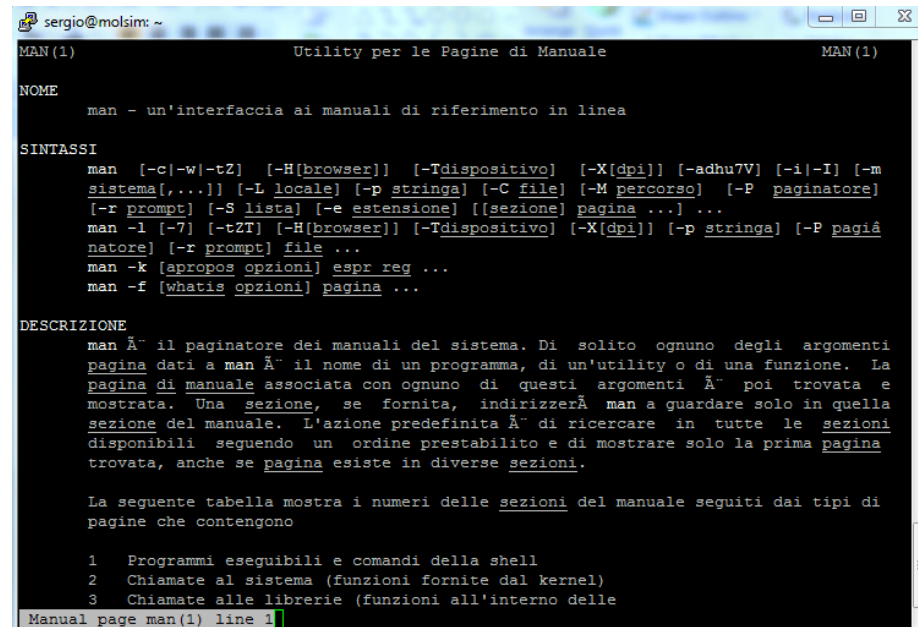
- Una volta partito il sistema l'utente esegue la procedura di Login.
- Se si è installato un ambiente grafico di desktop, la procedura è gestita dall'ambiente stesso.
- Il *Login* richiede sempre uno *username* e una *password*.
- All'utente viene assegnato una zona di disco identificata come home directory (/home/utente/)
- Alla fine della procedura di *Login* l'utente "si trova" dentro il **terminale**, in una particolare **shell**, posizionato nella sua **home directory** cioè in una zona di disco in cui ha tutti i privilegi per creare nuovi *files* o *directories*.

Comandi principali: man

In Linux il comando di *help* non si chiama **help** !
Per avere informazioni su un qualsiasi comando si usa **man** che mostra l'uso del comando e delle sue opzioni in modo formattato.

Ovviamente per illustrare il funzionamento del comando **man stesso**, si può fare anche:

```
man man
```



```
sergio@molsim: ~
MAN(1) Utility per le Pagine di Manuale MAN(1)
NOME
man - un'interfaccia ai manuali di riferimento in linea

SINTASSI
man [-c|-w|-tZ] [-H[browser]] [-Tdispositivo] [-X[dpi]] [-adhu7V] [-i|-I] [-m
sistema[,...]] [-L locale] [-p stringa] [-C file] [-M percorso] [-P paginatore]
[-r prompt] [-S lista] [-e estensione] [[sezione] pagina ...] ...
man -l [-7] [-tZ] [-H[browser]] [-Tdispositivo] [-X[dpi]] [-p stringa] [-P paginatore] [-r prompt] file ...
man -k [apropos opzioni] espr reg ...
man -f [whatis opzioni] pagina ...

DESCRIZIONE
man Ã" il paginatore dei manuali del sistema. Di solito ognuno degli argomenti
pagina dati a man Ã" il nome di un programma, di un'utility o di una funzione. La
pagina di manuale associata con ognuno di questi argomenti Ã" poi trovata e
mostrata. Una sezione, se fornita, indirizzerÃ" man a guardare solo in quella
sezione del manuale. L'azione predefinita Ã" di ricercare in tutte le sezioni
disponibili seguendo un ordine prestabilito e di mostrare solo la prima pagina
trovata, anche se pagina esiste in diverse sezioni.

La seguente tabella mostra i numeri delle sezioni del manuale seguiti dai tipi di
pagine che contengono

1 Programmi eseguibili e comandi della shell
2 Chiamate al sistema (funzioni fornite dal kernel)
3 Chiamate alle librerie (funzioni all'interno delle
Manual page man(1) line 1
```

Comandi principali: ls

- Permette di elencare il contenuto (in files) di una cartella.

Opzioni:

- -l (informazioni estese)
- -a (visualizza file nascosti, cioè iniziati con il .)
- -R (visualizza sottocartelle)
- -t (ordina la lista di file secondo l'ora dell'ultima modifica)

- Esempio:

```
$ ls -laR
```

- Carattere jolly “*”:

```
$ ls -l *.py
```

```
sergio@molsim:~/mutprotdb$ ls -laR
.:
totale 11073520
drwxr-xr-x  3 sergio sergio      4096 2014-10-08 16:09 .
drwxr-xr-x 32 sergio sergio      4096 2014-09-29 12:36 ..
-rw-r--r--  1 sergio sergio     3379 2014-07-17 16:41 01-CreateTables.sql
-rw-r--r--  1 sergio sergio      931 2014-07-17 10:32 02-AlterTables.sql
drwxr-xr-x  2 sergio sergio      4096 2014-10-08 16:09 dati
-rwxr-xr-x  1 sergio sergio      519 2014-05-12 16:40 db-connection.py
-rwxr-xr-x  1 sergio sergio      498 2014-05-15 12:11 db-select.py
-rwxr-xr-x  1 sergio sergio     1060 2014-05-04 14:30 elab-great-xml.py
-rwxr-xr-x  1 sergio sergio     1280 2014-05-14 19:02 elab-uniprot.py
-rwxr-xr-x  1 sergio sergio    10494 2014-07-18 15:01 elab-uniprot-xml.py
-rwxr-xr-x  1 sergio sergio     1942 2014-05-03 01:46 leggi1-xml.py
-rwxr-xr-x  1 sergio sergio     1302 2014-05-02 18:14 leggi-xml.py
-rw-r--r--  1 sergio sergio    8613569 2014-07-19 23:51 mutagenesis.log
-rw-r--r--  1 sergio sergio     74484 2014-07-16 12:29 P12023.xml
-rwxr-xr-x  1 sergio sergio       78 2014-07-16 12:07 prova-split.py
-rw-r--r--  1 sergio sergio    11444 2014-05-02 15:12 prova.xml
-rwxr-xr-x  1 sergio sergio       435 2014-05-02 14:53 scrivi-xml.py
-rw-r--r--  1 sergio sergio    5576522688 2014-04-30 16:27 uniprot_sprot-20140430.xml
-rw-r--r--  1 sergio sergio    5753978017 2014-07-17 15:47 uniprot_sprot.xml

./dati:
totale 8
drwxr-xr-x  2 sergio sergio      4096 2014-10-08 16:09 .
drwxr-xr-x  3 sergio sergio      4096 2014-10-08 16:09 ..
sergio@molsim:~/mutprotdb$
```


Comandi principali: less, more e cat

- Permette di visualizzare un file (**di testo**) con la gestione della paginazione

```
less nomefile
```

- Permette di visualizzare un file (**di testo**) con solo la gestione dell'avanzamento di pagina

```
more nomefile
```

- Permette di visualizzare un file (**di testo**) senza la gestione della paginazione

```
cat nomefile
```

Comandi principali: cd e pwd

- Permette di cambiare la cartella corrente

```
cd altra_cartella      (relativo)
cd /path/altra_cartella (assoluto)
```

Opzioni:

- cartella corrente: .
- cartella superiore: ..
- home directory: ~ (oppure lasciare in bianco)

- **Esempio:**

```
$ cd ..      (va alla cartella sup.)
$ cd ./utente/cartella (se esiste)
$ cd        (va alla home)
```

- Visualizzare il path assoluto della cartella corrente

```
pwd
```

I permessi nei file linux

Permessi

```
$ ls -l          Proprietario Gruppo Dimensione Data ultima modifica
totale 44
drwxr--r--  2 fred fred    48 25 mag 22:11 drafts
-r-xr-xr-x  1 fred fred   8460 25 mag 22:12 edit
-rw-r--r--  1 fred fred  30405 25 mag 22:12 edition-32
lrwxrwxrwx  1 fred fred     4 25 mag 22:16 editor -> edit
```

- La prima lettera indica se è una **directory** (d) , un **link simbolico** (l) o un **file** (-) mentre le restati 9 (a gruppi di 3) definiscono i permessi per **proprietario**, **gruppo** e **altri utenti**.
- I tipi di permesso possono essere:
 - **lettura** (r)
 - **scrittura** (w)
 - **esecuzione** (x)
- Per cambiare il proprietario di un file si usa il comando **chown**, è necessario essere proprietario del file o “super user” (**sudo**).

```
$ chown proprietario.gruppo nomefile
```

- Per cambiare i permessi di un file si usa il comando **chmod**, è necessario essere proprietario del file o “super user” (**sudo**).

```
$ chmod 760 nomefile
$ chmod 666 nomefile
$ chmod 777 nomefile
$ chmod +r nomefile
$ chmod -w nomefile
$ chmod -x nomefile
```

Cifra	Rappresentazione simbolica	Permessi
0	---	nessun permesso
1	--x	esecuzione
2	-w-	scrittura
3	-wx	scrittura ed esecuzione
4	r--	lettura
5	r-x	lettura ed esecuzione
6	rw-	lettura e scrittura
7	rwx	lettura, scrittura, esecuzione

Comandi principali: mkdir e rmdir

- Creare nuove cartelle

```
mkdir nome_cartella
```

- Esempio:

```
$ mkdir nuovacartella1 nuovacartella2
```

- Eliminare una cartella (solo se vuota)

```
rmdir nome_cartella
```

Comandi principali: cp, mv e rm

- Copiare file e cartelle

```
cp [opzioni...] sorgente.. destinazione
```

- Spostare o rinominare file e cartelle

```
mv [opzioni...] sorgente.. destinazione
```

- **Eliminare file** (opzioni: -i per modalità interattiva, -r ricorsiva, elimina le cartelle **senza passare per il cestino!!!**)

```
rm [opzioni...] file
```

- **Eliminare cartelle** (Cancella la cartella e tutto il suo contenuto. Attenzione ad utilizzarla, **cancella tutto !!!**)

```
rm -r nome_cartella
```


Creare e modificare files di testo

- Il programma per editare “**file di testo**”, quindi anche “**sorgenti di programma**” si chiama **gedit**, questo viene lanciato dal terminale.

```
gedit nome-file
```

- Gedit è un editor semplice, alcune comandi dell'editor sono:
 - Per selezionare una parte del testo, utilizzate “shift” e le “freccie”.
 - Per selezionare tutto il testo utilizzate Ctrl-A.
 - Ctrl-C permette di “copiare” il testo selezionato.
 - Ctrl-X permette di “tagliare” il testo selezionato.
 - Ctrl-V permette di “incollare” quello che avete “copiato” con Ctrl-C o “tagliato” con Ctrl-X.

Editare il primo programma in C

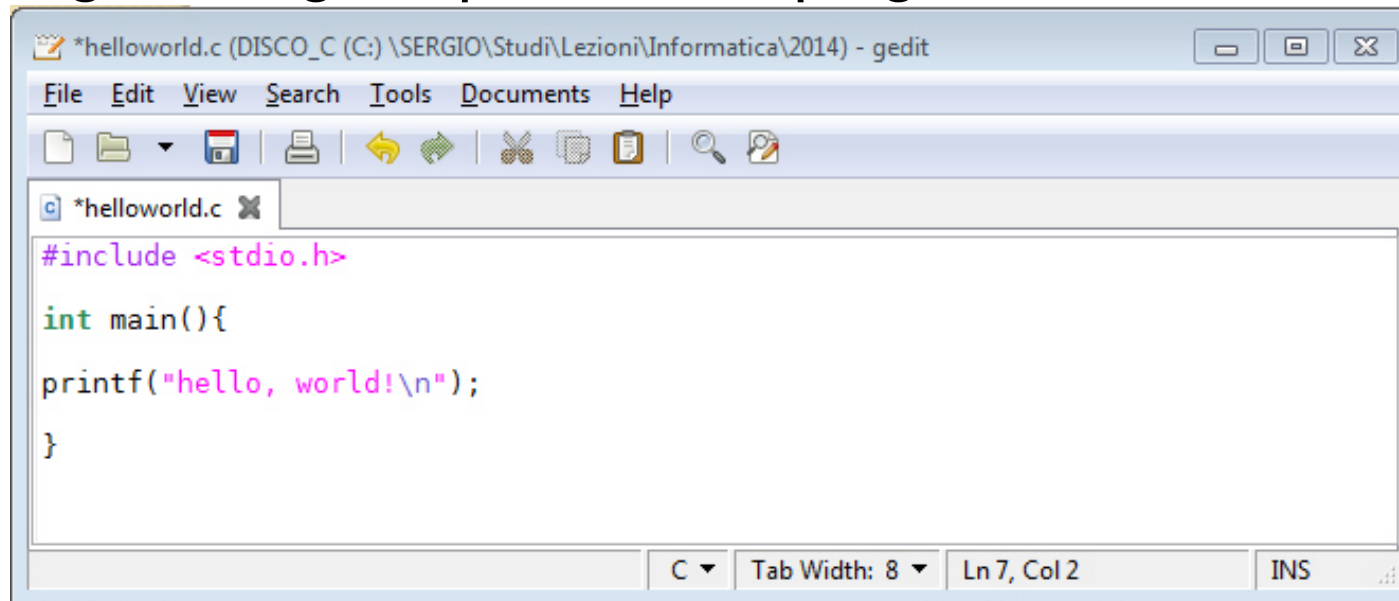
- Aprite il terminale e posizionatevi nella vostra “home”

```
cd /home/nome-utente
```

- Create il vostro primo programma in “C” con il nome helloworld.c

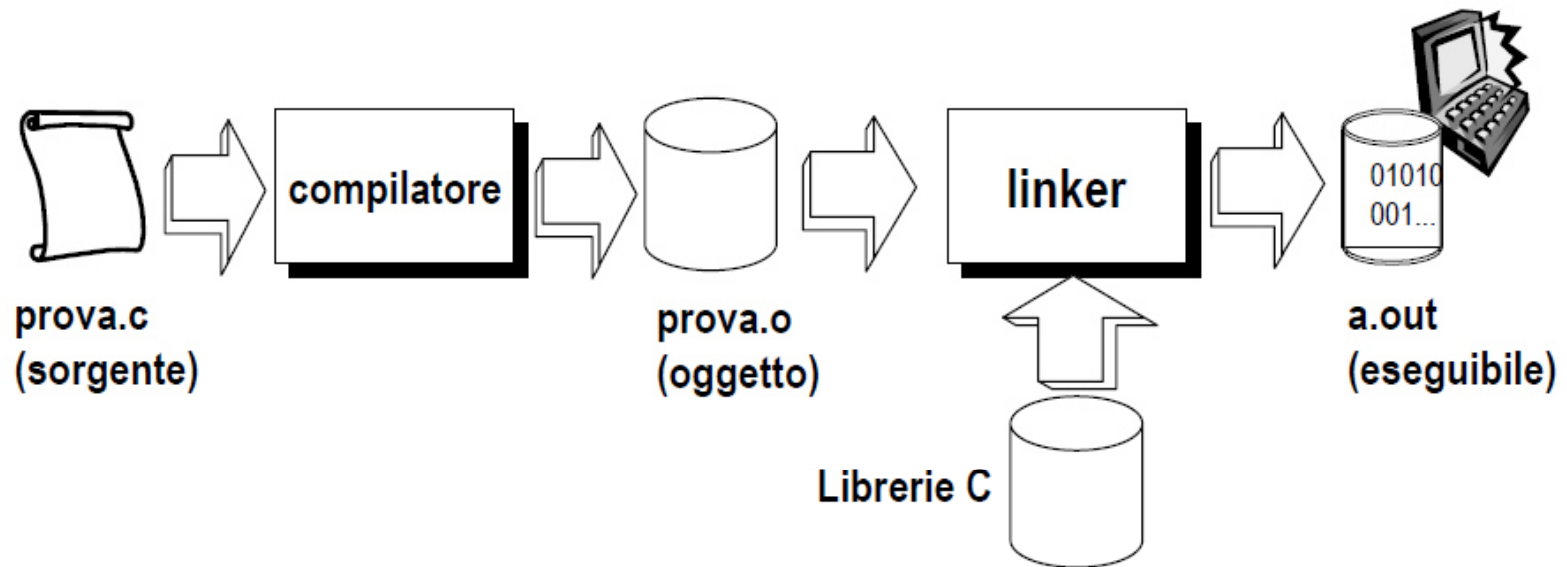
```
gedit helloworld.c
```

- Una volta dentro l’editor “**gedit**” scrivete esattamente le seguente righe e poi salvate il programma.



```
*helloworld.c (DISCO_C (C:) \SERGIO\Studi\Lezioni\Informatica\2014) - gedit
File Edit View Search Tools Documents Help
*helloworld.c
#include <stdio.h>
int main(){
printf("hello, world!\n");
}
C Tab Width: 8 Ln 7, Col 2 INS
```

Il compilatore C



Dentro la **shell**, si può compilare un programma sorgente “prova.c”, scritto in linguaggio C, tramite i seguenti comandi:

- **gcc prova.c**
Compila “prova.c” e genera un eseguibile col nome “**a.out**”
- **gcc -o prova prova.c**
Compila “prova.c” e genera un eseguibile con il nome “**prova**”

Compilare helloworld.c

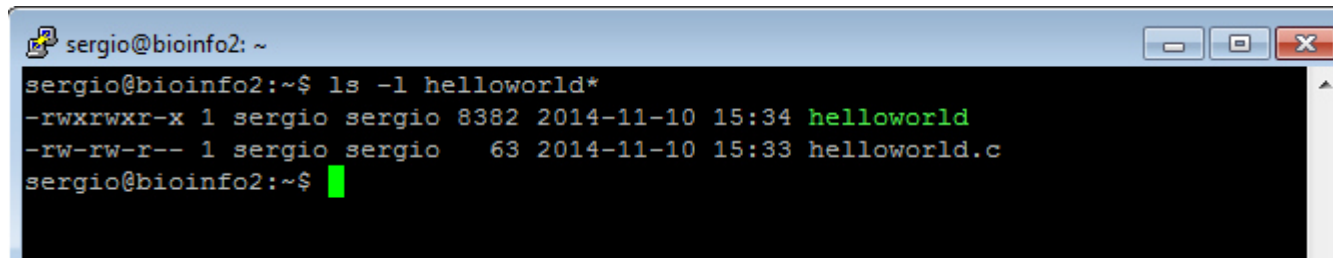
- Aprite il terminale e posizionatevi nella vostra “home”

```
cd /home/nome-utente
```

- Compilare il programma “**helloworld.c**”, dando all’eseguibile il nome “*helloworld*”

```
gcc -o helloworld helloworld.c
```

- Il compilatore vi creerà un eseguibile “helloworld”, con già le autorizzazioni di esecuzione.

A terminal window titled "sergio@bioinfo2: ~" showing the output of the command "ls -l helloworld*". The output lists two files: "helloworld" with permissions "-rwxrwxr-x" and "helloworld.c" with permissions "-rw-rw-r--".

```
sergio@bioinfo2:~$ ls -l helloworld*  
-rwxrwxr-x 1 sergio sergio 8382 2014-11-10 15:34 helloworld  
-rw-rw-r-- 1 sergio sergio  63 2014-11-10 15:33 helloworld.c  
sergio@bioinfo2:~$
```

Eseguire programmi

- In windows gli eseguibile sono “.exe”, in linux un eseguibile si riconosce perche il terzo attributo del file è “x”, questi normalmente vengono colorati col colore verde:

```
$ ls -l  
$ chmod 777 nome-programma
```

```
sergio@molsim:~/mutprotdb$ ls -l  
totale 11073508  
-rw-r--r-- 1 sergio sergio      3379 2014-07-17 16:41 01-CreateTables.sql  
-rw-r--r-- 1 sergio sergio       931 2014-07-17 10:32 02-AlterTables.sql  
-rwxr-xr-x 1 sergio sergio       519 2014-05-12 16:40 db-connection.py  
-rwxr-xr-x 1 sergio sergio       498 2014-05-15 12:11 db-select.py  
-rwxr-xr-x 1 sergio sergio      1060 2014-05-04 14:30 elab-great-xml.py  
-rwxr-xr-x 1 sergio sergio       1280 2014-05-14 19:02 elab-uniprot.py  
-rwxr-xr-x 1 sergio sergio      10494 2014-07-18 15:01 elab-uniprot-xml.py
```

- Per che un programma possa essere eseguito, questo deve trovarsi in una delle cartelle indicate dalla variabile di ambiente **PATH**, oppure deve essere lanciato con la **path assoluta**:

```
$ echo $PATH (Per visualizzare i percorsi eseguibili)
```

Se il programma si trova in una delle cartelle di \$PATH:

```
$ nomeprogramma
```

Se il programma si trova nella cartella corrente ma non nella \$PATH:

```
$ ./nomeprogramma
```

Se il programma non si trova ne nella cartella corrente ne nella \$PATH:

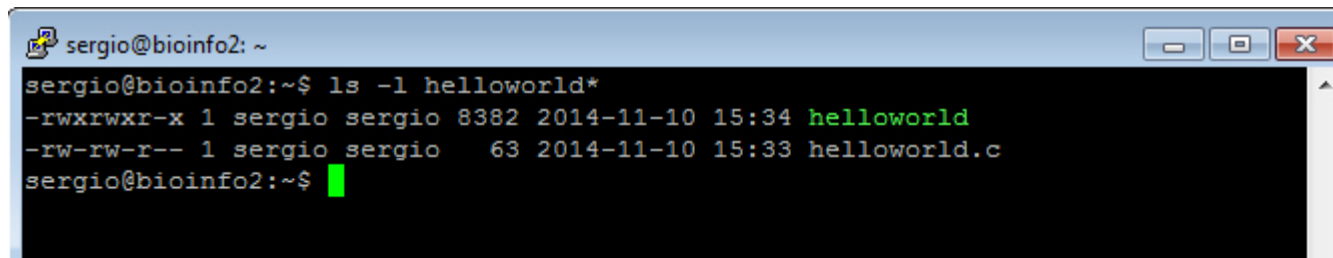
```
$ /path_assoluta/nomeprogramma
```


Eseguire helloworld

- Aprite il terminale e posizionatevi nella vostra “home”

```
cd /home/nome-utente
```

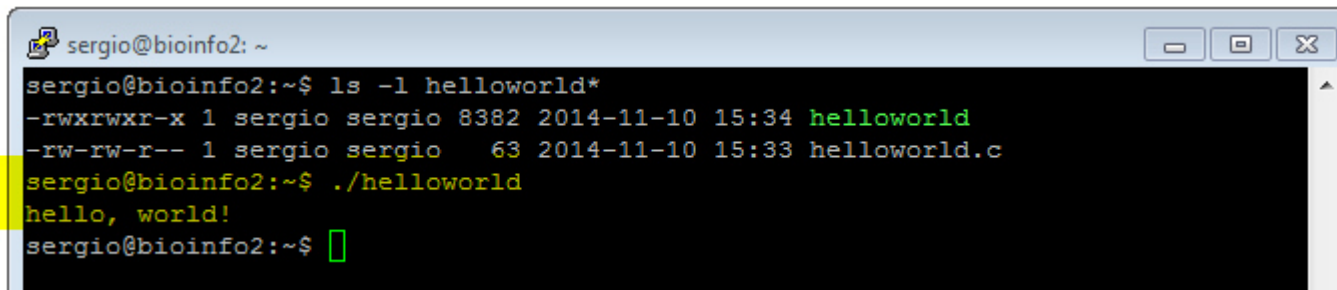
- Se avete già compilato in precedenza il programma, facendo “**ls -l**” troverete il vostro programma “helloworld”



```
sergio@bioinfo2: ~  
sergio@bioinfo2:~$ ls -l helloworld*  
-rwxrwxr-x 1 sergio sergio 8382 2014-11-10 15:34 helloworld  
-rw-rw-r-- 1 sergio sergio 63 2014-11-10 15:33 helloworld.c  
sergio@bioinfo2:~$
```

- Per eseguirlo lanciare il seguente comando dentro il terminale:

```
./helloworld
```



```
sergio@bioinfo2: ~  
sergio@bioinfo2:~$ ls -l helloworld*  
-rwxrwxr-x 1 sergio sergio 8382 2014-11-10 15:34 helloworld  
-rw-rw-r-- 1 sergio sergio 63 2014-11-10 15:33 helloworld.c  
sergio@bioinfo2:~$ ./helloworld  
hello, world!  
sergio@bioinfo2:~$
```



**Grazie
per la vostra
attenzione**