

Prima Esercitazione

Linux shell e
linguaggio C

Linux: desktop e shell

Dall'interfaccia grafica di linux è possibile

- aprire programmi con interfaccia grafica, es:
un browser web:
 - Applications -> Web Browser
 - **caricare <https://iol.unibo.it/>**
- aprire altre finestre di shell (non-login shell):
 - Applications -> Terminal emulator
 - **exit** uscire dalla non-login shell

il comando **pwd**

Dopo il login, l'utente può cominciare a operare all'interno di uno specifico directory (la sua **home**).

- Visualizzare il directory corrente con **pwd** (print working directory)

```
dloreti@cloudpS:~$ pwd  
/home/dloreti
```

il comando **ls**

provare il comando **ls [-opzioni...]** [file...]

- listare il direttorio corrente:

ls -l (list directory - as a List)

ls -la (list directory - as a List -All)

ls a* ...

- creare qualche file (vuoto):

“> a1.txt” “> a2.txt” e riprovare “**ls a***”

* corrisponde a qualunque stringa, anche vuota.
E' un metacarattere!

- Provare: **ls -l /home** ...cosa fa?

bit di protezione: lettura, scrittura, esecuzione

Ad esempio, il file:

| | U | G | O |
|-----------|-------|-------|-------|
| pippo.txt | 1 1 1 | 1 0 1 | 0 0 1 |
| | r w x | r - x | - - x |

- è leggibile, scrivibile, eseguibile per il proprietario
- è leggibile e eseguibile per gli utenti dello stesso gruppo
- solo eseguibile per tutti gli altri

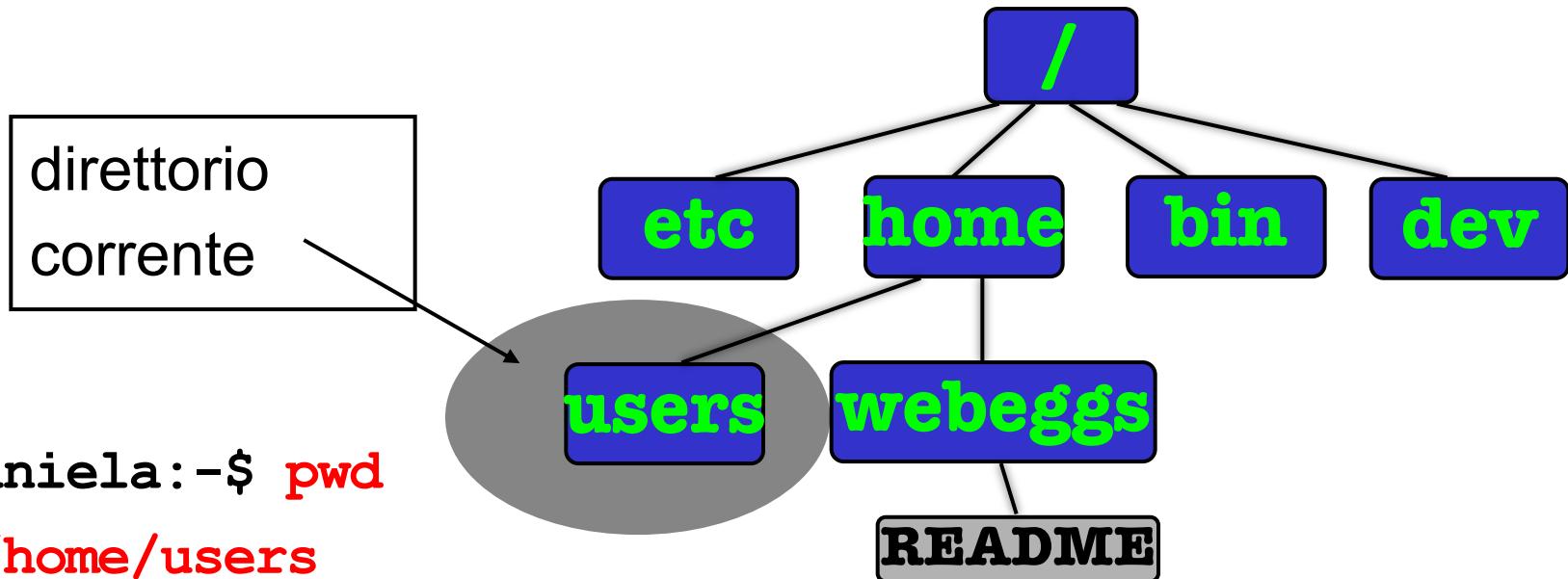
formato ottale: 111 => 7; 101 => 5; ... -rwxr-x--x => 0751

altri esempi: 110 => 6; 010 => 2; 011 => 3; 100 => 4

creare il file **file.txt** con un editor (es: Applications -> Accessories > Mousepad) e salvarlo nella home. Che diritti ha?



nomi relativi / assoluti: *esempio*



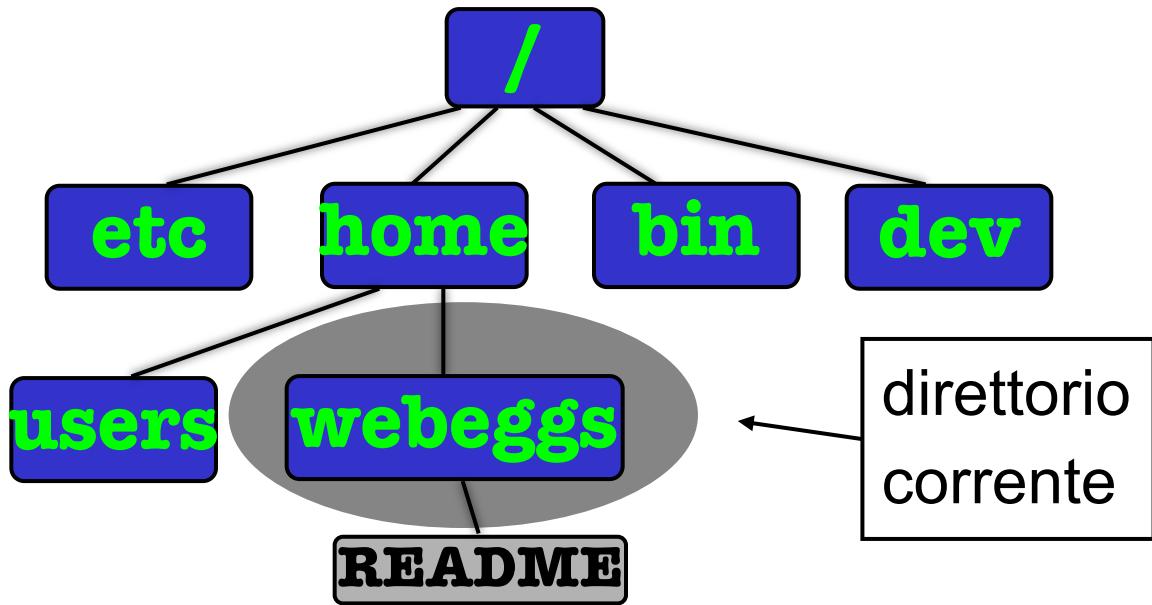
daniela:~\$ ls -l /home/webeggs/README (1) ASS.

daniela:~\$ ls -l ../webeggs/README (2) REL.

daniela:~\$ ls -l ../../home/webeggs/README (3) REL.

(1) (2) e (3) sono equivalenti

nomi relativi / assoluti: *esempio*



```
daniela:~$ pwd  
/home/webeggs
```

```
daniela:~$ ls -l README (1) REL.
```

```
daniela:~$ ls -l ../webeggs/README (2) REL.
```

```
daniela:~$ ls -l ./README (3) REL.
```

```
daniela:~$ ls -l /home/webeggs/README (4) ASS.
```

(1), (2), (3) e (4) sono equivalenti

comandi per la gestione del file system

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

il comando cd: change directory

È possibile ‘**spostarsi**’ da un direttorio attraverso 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**

il comando cd

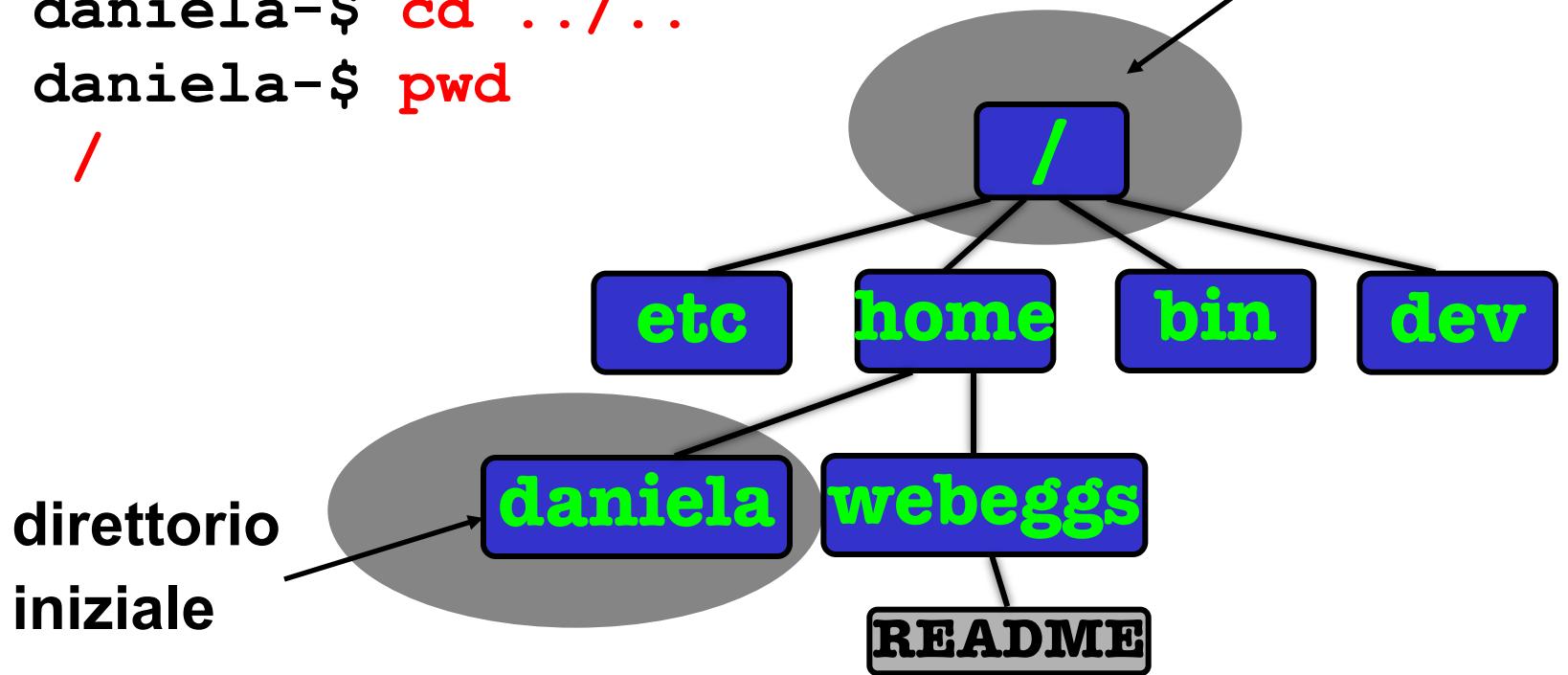
Esempio:

```
daniela-$ pwd  
/home/daniela
```

```
daniela-$ cd ../../..
```

```
daniela-$ pwd  
/
```

nuovo direttorio corrente



il comando cd

Esempio:

```
daniela-$ pwd
```

/dev

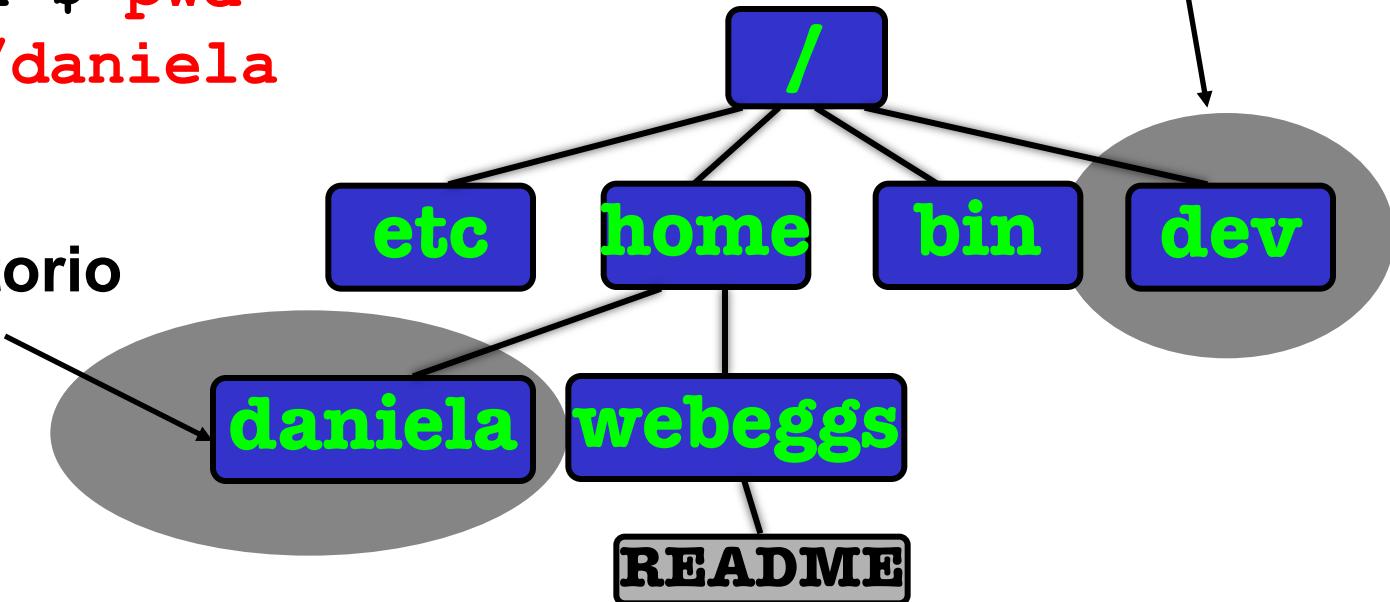
```
daniela-$ cd
```

```
daniela-$ pwd
```

/home/daniela

nuovo directory
corrente

direttorio
iniziale

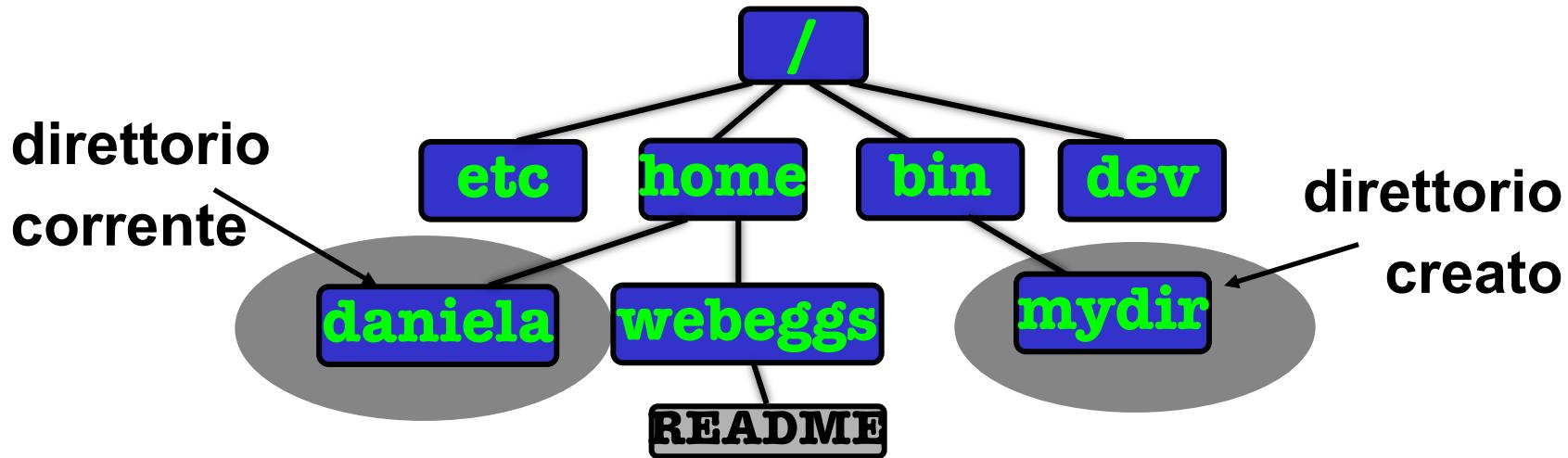


modifica del file system: directory

- **Creazione** di un direttorio: `mkdir <nomedir>`
- **Eliminazione** : `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** sul direttorio stesso;
 - che la directory sia **vuota**

Esempio di mkdir/rmdir

creo un nuovo directory utilizzando il percorso relativo o assoluto:



```
:~$ mkdir /bin/mydir      occorre avere w su bin  
:~$ mkdir ../../bin/mydir
```

```
:~$ rmdir /bin/mydir      occorre avere w su mydir
```

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
 - provare: cat file.txt
- **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)

cancellazione, copia e spostamento di file

- **copia** di un file (e diritti):

`cp <nomefile> <nuovofile>`

- **spostamento** di un file (e diritti):

`mv <nomefile> <nuovofile>`

- **eliminazione** di un file:

`rm <nomefile>`



esempi

```
:~$ cp file.txt sera (copia il file)
```

```
:~$ ls  
file.txt sera
```

```
:~$ mv file.txt poesia (sposta/rinomina il file)
```

```
:~$ ls  
poesia sera
```

```
:~$ rm poesia  
:~$ ls  
sera
```

NB: Per eliminare un file occorre avere diritto di scrittura sulla directory che lo contiene

modifica dei bit di protezione

- **chown / chgrp** permettono di modificare la **proprietà** di un file (occorre essere root)
- è possibile **cambiare i permessi** dei propri file attraverso il comando chmod:

chmod <mode> <nomefile>

[ugoa] [[+ - =] [rwxXstugoa...] ...] [, ...]

oppure: formato ottale dei bit di protezione



esempio chmod

```
:~$ ls -l sera
-rw-rw-r-- 1 daniela staff ... sera

:~$ chmod 0666 sera ([6]8 = [110]2)
:~$ ls -l sera
-rw-rw-rw- 1 daniela staff ... sera

:~$ chmod a-w,u=rw sera
:~$ ls -l sera
-rw-r--r-- 1 daniela staff ... sera
```



esempi modifica file (diritti)

```
$ ls -l sera  
-rw-r--r-- 1      loreti staff ... sera  
$ chmod 0400 sera          ([4]8 = [100]2)  
$ mv sera subito  
$ ls -l subito      (mv sposta il file e i suoi diritti)  
-r----- 1      loreti staff ... subito  
$ rm subito  
rm: remove 'subito' ,overriding mode 0400[y/n]?
```

Questa domanda è una feature di **rm**, **NON** un fatto di permessi!

Il permesso di cancellare un file dipende dai permessi sulla directory che lo contiene

cancellazione, copia e spostamento di file: diritti

- **rm <nomefile>**
 - Per eliminare un file occorre avere diritto di scrittura sulla directory che lo contiene
- **cp <nomefile> <nuovofile>**
 - Per copiare un file occorre avere diritto di lettura sul file e di scrittura sulla directory di destinazione
- **mv <nomefile> <nuovofile>**
 - Per spostare un file occorre avere diritto di lettura sul file, di scrittura sulla directory di provenienza e su quella di destinazione

Programmare in linguaggio C

editor, compilatore, parametri

Editor

- Esistono vari editor offerti dal sistema:
 - vi: editor standard UNIX (1976)
 - Vim: (vi improved) versione estesa
presente in tutte le distribuzioni unix
 - **Kate**(Applications->Accessories->Kate),
Kwrite
 - Gedit
 - Emacs
 - ...

Compilazione sorgenti - C

- un file.c non è direttamente eseguibile dal processore. Occorre tradurlo in linguaggio macchina compilandolo.
- Comando **gcc <file>**:
 - compila **<file>** producendo il **file eseguibile a.out**
 - per dare nome diverso al file prodotto: opzione **-o**
- Esempio: **gcc file.c -o fex**

- occorre rendere fex eseguibile:

chmod u+x fex

- Esecuzione: **./fex <parametri>**

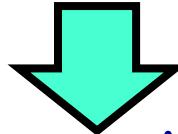
Compilazione sorgenti - C

file.c

(file sorgente C)

```
main(int argc, char *argv[])
{ ... }
```

```
:~$ gcc file.c -o fex
```



fex (file in linguaggio macchina)

```
@<<Q?tdR?td` `` ??/lib64/ld-linux-x86-
64.so.2GNUGNU?>;D??c\?S?%D???Dgp....
```

```
:~$ ls -l fex
```

```
-rw-r--r-- 1 dloreti staff ... fex
```



occorre rendere fex eseguibile: **chmod u+x fex**

Esecuzione: **./fex <parametri>**

Parametri della linea di comando: *argc, argv*

```
main (int argc, char *argv[]){ }
```

- **int argc:** rappresenta il numero degli argomenti effettivamente passati al programma; anche il nome stesso del programma (nell'esempio, fex) e` considerato un argomento, quindi argc vale sempre almeno 1.
- **char **argv:** vettore di stringhe, ciascuna delle quali contiene un diverso argomento. Per convenzione, **argv[0]** contiene il **nome del programma stesso**.

Esempio

```
// sorgente di file.c
main(int argc, char *argv[])
{ ... }
```

- invoco l'eseguibile generato coi seguenti parametri:

- : \$./fex roma 1 x

- quindi:

argc = 4

argv[0] = "./fex"

argv[1] = "roma"

argv[2] = "1"

argv[3] = "x"

NB: gli argomenti sono passati tutti come stringhe

Esercitazione 1- Obiettivi

Programmazione C

- Gestione dei parametri in ingresso
 - argomenti **argc** ed **argv**
 - controllo di correttezza dei parametri in ingresso
- Gestione delle stringhe
 - libreria string.h
- usare **gcc** e **chmod**

Esercizio 1 - La Lotteria (1/3)

Vogliamo realizzare un programma che realizzi una lotteria organizzata tra studenti di unibo.

A questo scopo si progetti un programma C con un'interfaccia del tipo

\$./lotteria stud1 stud2 stud3... studM W

Dove:

- **stud1,..studM** sono M stringhe ($M > 0$), ognuna delle quali rappresenta l'identificatore unico di uno studente; in particolare, si assuma che il formato di ogni identificatore sia:
- **<campus><matricola>** (es. B00000930345) in cui:
 - **<campus>** è una stringa di 2 caratteri alfabetici che indicano il campus ove ha sede il corso dello studente (BO per bologna, CE per cesena, FO per forlì, RA per ravenna)
 - **<matricola>** è una stringa di 10 caratteri numerici corrispondenti al numero di matricola dello studente
- **W** è un intero positivo minore o uguale a M, che rappresenta il numero del biglietto vincitore della lotteria.

Esercizio 1 - La Lotteria (2/3)

Inizialmente, il programma deve effettuare opportuni controlli sugli argomenti, ovvero:

- controllare che sia stato passato **almeno** l'identificatore di **uno studente**;
- controllare che gli identificatori di studenti passati al programma siano **conformi alle caratteristiche** sopra indicate.
- Verificare che W sia positivo e minore o uguale a M.

Esercizio 1 - La Lotteria (3/3)

Se i controlli sugli argomenti danno esito positivo, il programma dovrà associare ad ogni studente S_i un numero N_i determinato in **modo casuale e compreso tra 1 e M**. Tale numero N_i rappresenta il **biglietto** della lotteria assegnato allo studente S_i . Realizzare l'assegnazione in modo tale da escludere che due studenti abbiano lo stesso biglietto.

Terminata l'assegnazione dei biglietti, il programma dovrà **individuare lo studente vincitore**, cioè quello il cui biglietto ha lo stesso valore di W e stamparne a video l'identificatore.

Esercizio 1 – suggerimenti

- convertire una stringa in int:

```
int atoi(char *string) di <stdlib.h>
```

- per generare un numero random tra 1 e M usare la funzione **rand()** di **<stdlib.h>**

```
int r = rand() % M + 1;
```

- per controllare che un carattere sia una cifra numerica potete usare la funzione **isdigit()** della libreria **<ctype.h>**

```
char c='5';
```

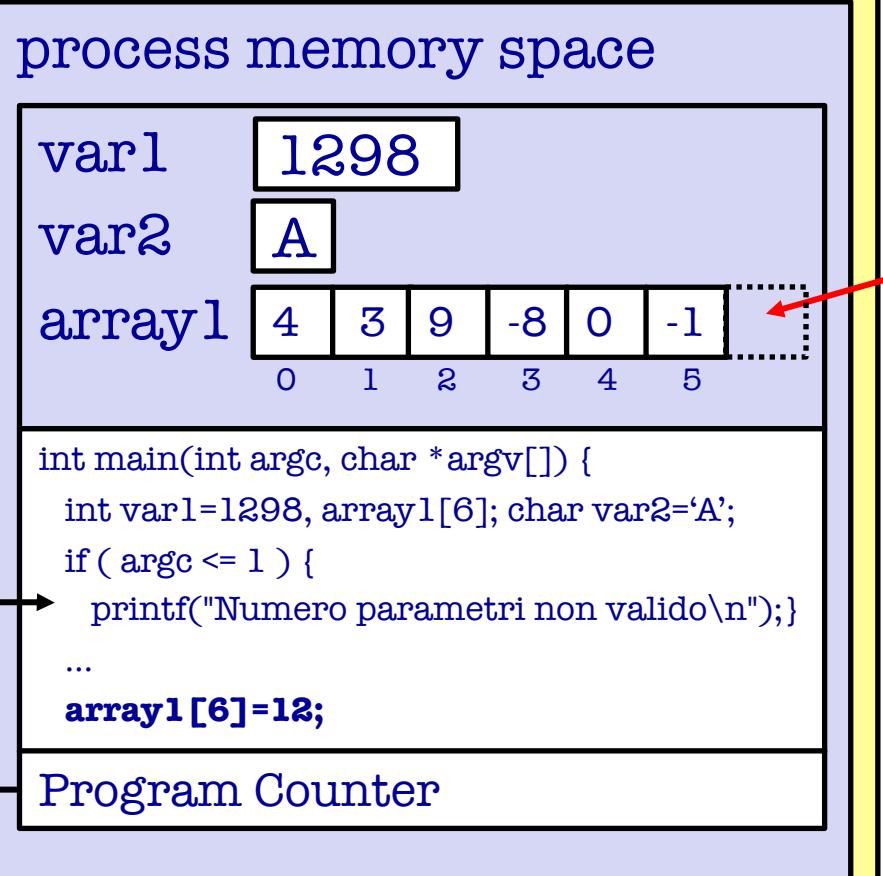
```
int d = isdigit(c);
```

```
//d=1 se c è un digit
```

```
//d=0 altrimenti
```

Segmentation fault

RAM



frequente con array e puntatori

il processo sta cercando di accedere a un'area di memoria per la quale non ha il permesso

→ il SO interviene e termina il processo