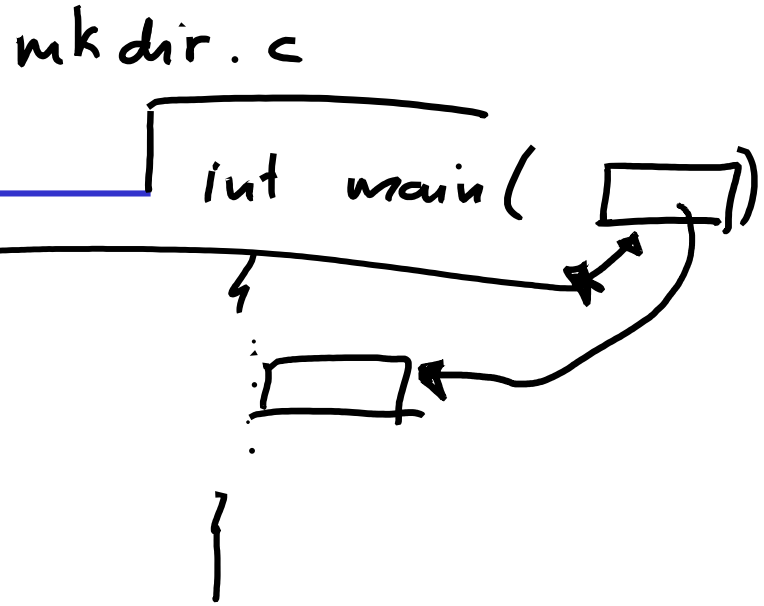
```
int main(int argc, char *argv[])
{
    ...
    return 0;
}
```

Argomenti della linea di comando

LINEA DI COMANDO




```
[sistop@hal9042:~] mkdir helloworld
[sistop@hal9042:~] cd helloworld
[sistop@hal9042:~/helloworld] gedit helloworld.c
[sistop@hal9042:~/helloworld] gcc helloworld.c -o
  helloworld
[sistop@hal9042:~/helloworld] ls
helloworld helloworld.c helloworld.o
[sistop@hal9042:~/helloworld] ./helloworld
Hello World!!
[sistop@hal9042:~/helloworld]
```

ARGOMENTI DELLA LINEA DI COMANDO

- Anche la funzione **`main`** può avere parametri. I parametri rappresentano gli eventuali argomenti passati al programma, quando viene messo in esecuzione:

```
./prog arg1 arg2 ... argN
```

- I parametri formali di **`main`**, differentemente dalle altre funzioni, sono sempre due, convenzionalmente chiamati **`argc`** e **`argv`**

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

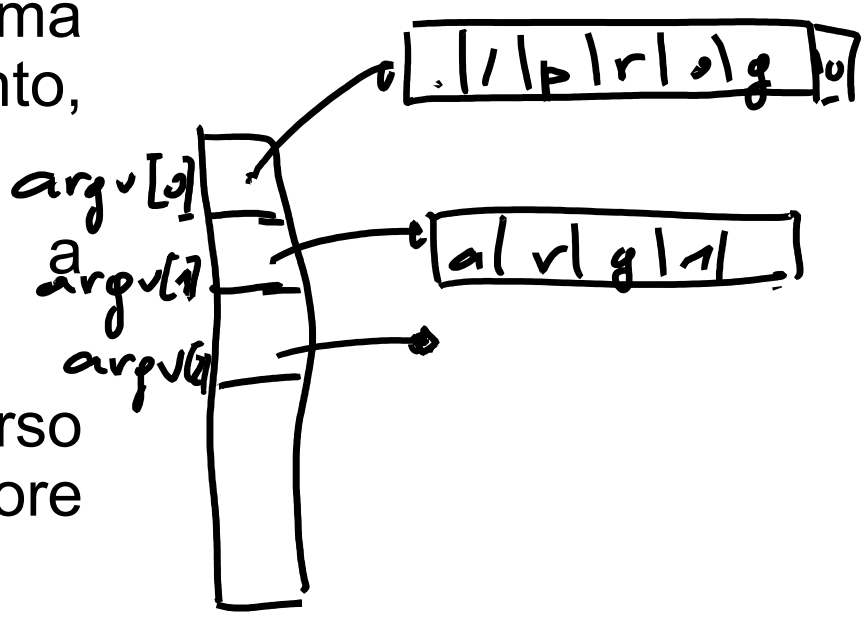
ARGOMENTI DELLA LINEA DI COMANDO

./prog arg1 arg2

`int argc`: è un parametro di tipo ***intero***. Rappresenta il numero degli argomenti effettivamente passati al programma nella linee di comando con cui si invoca la sua esecuzione. Anche il nome stesso del programma (nell'esempio, *prog*) è considerato un argomento, quindi argc vale sempre almeno 1.

`char **argv:` è un puntatore a un puntatore a carattere, ovvero un array di stringhe.

* argv[]

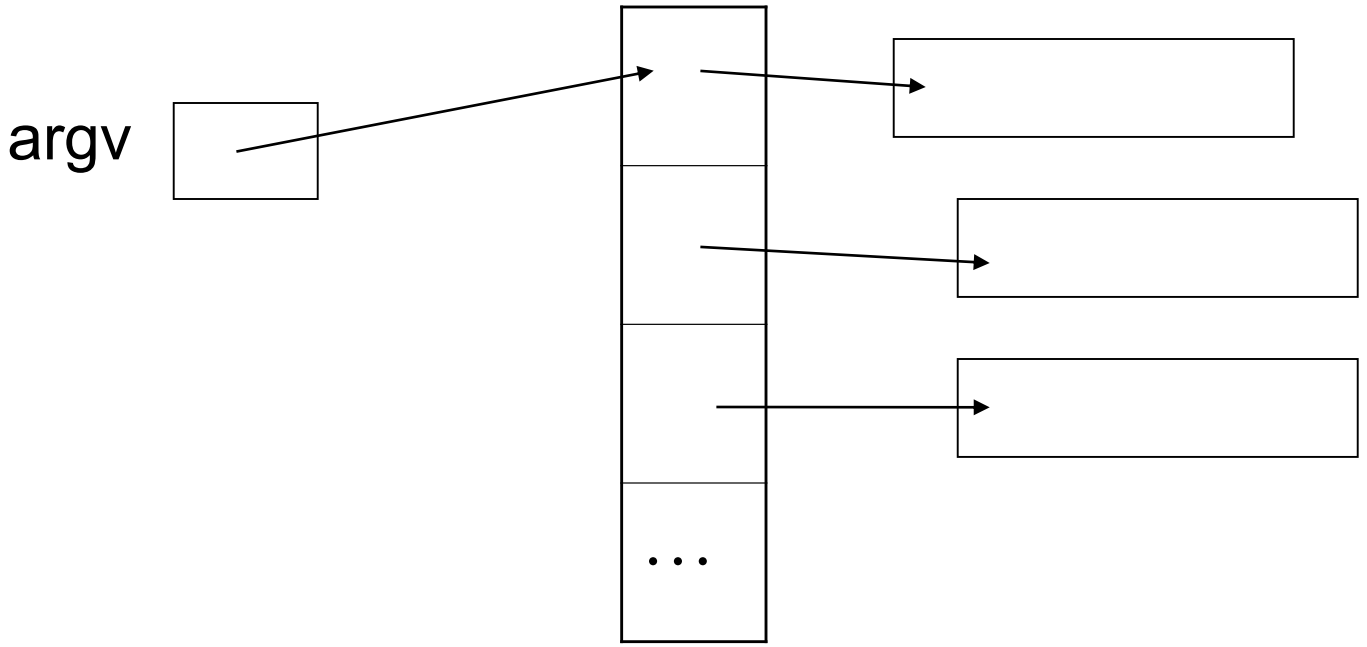


Ciascuna stringa nel vettore contiene un diverso argomento. Gli argomenti sono memorizzati nel vettore nell'ordine con cui sono dati dall'utente.

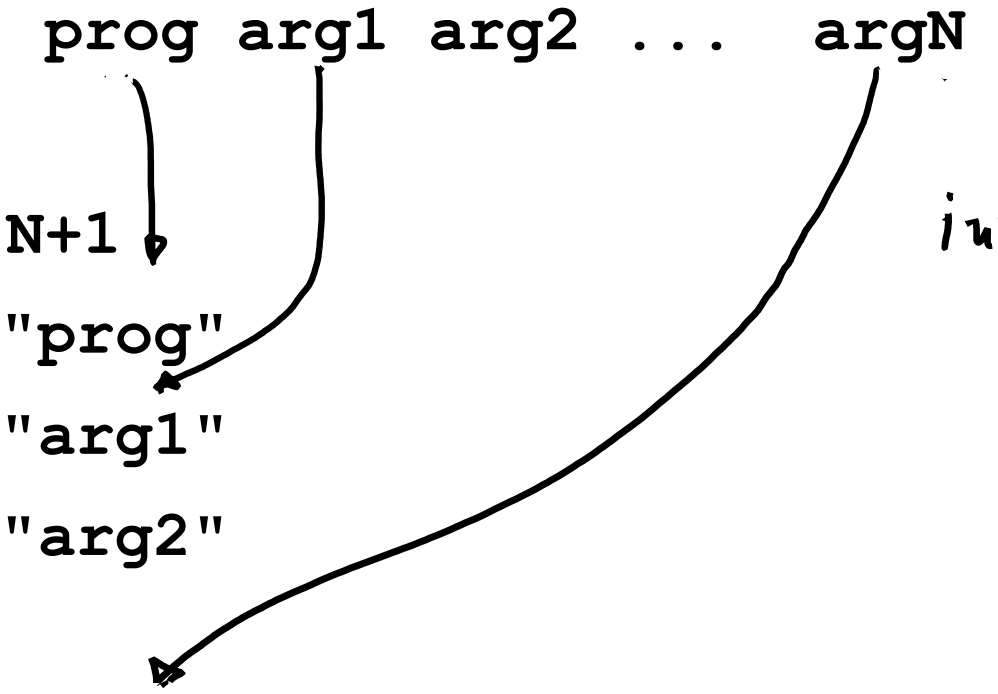
`argv[0]` contiene il nome del programma stesso.

argv

`char ** argv` (o `char * argv[]`) ha questa forma:



ARGOMENTI DELLA LINEA DI COMANDO



Quindi

`argc` vale `N+1`

`argv[0]="prog"`

`argv[1]="arg1"`

`argv[2]="arg2"`

`...`

`argv[N]="argN"`

Per convenzione, `argv[argc]` vale `NULL`.

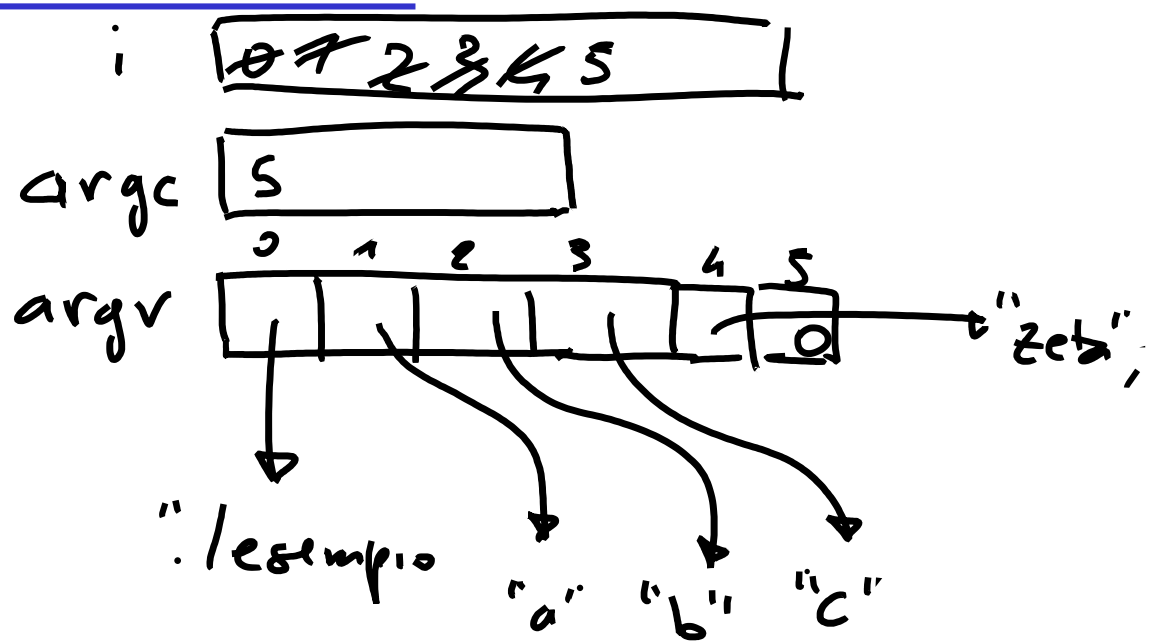
N+1

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

printf("%s\n",argv[0]);

}
```

ESEMPIO

Programma che stampa i suoi argomenti

```
#include <stdio.h>
/* programma esempio.exe */
main(int argc, char *argv[])
{
  int i;

  for(i=0; i<argc; i++)
      printf("%s ",argv[i]);
  return 0;
}
```



Invocazione

```
./esempio a b c zeta
```

Cosa stampa?

./esempio a b c zeta

Esercizio

Si scriva un programma C che prende dalla linea di comando un sequenza di nomi di file di testo:

argv[0] argv[1] argv[2]

```
concatena fileout.txt file1.txt file2.txt
  … fileN.txt
```

argv[argc - 1]

il programma deve creare il file **`fileout.txt`** ottenuto concatenando i file **`file1.txt`** … **`fileN.txt`**

Per fare questo, si scriva una procedura o funzione che prende come parametri un puntatore ad un FILE di output (che deve essere stato aperto) ed il nome di un file di input e ricopia il secondo sul primo

```
void ricopia(FILE * f_out, char nome_in[]);
```

Valore di ritorno

Il **`main`** è una funzione che a default restituisce un **`int`**.

Il valore di ritorno può essere letto dal sistema operativo.

In DOS/Windows c'è una variabile **`%ERRORLEVEL%`**

In Linux la variabile si chiama $?

Essa contiene il valore restituito dall'ultima operazione; può essere visualizzata col comando **`echo`**.

Lo stesso valore può anche essere restituito come parametro della funzione **`exit`**.

```
#include <stdio.h>
int main(int argc, char *argv[])
{ return 5;
}
```

exit(5);

```
$ ./a.out
$ echo $?
5
```