argomenti da riga di comando

argc ◇ argv

Cristiana Bolchini

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

    if(argc == NIN){
       val = atoi(argv[1]);
```

acquisizione da riga di comando

- soluzione che prevede di mettere sulla riga di comando, dopo il nome del programma eseguibile, i dati d'ingresso al programma
- sostituisce la soluzione in cui si acquisisce il dato nel flusso del programma, mediante costrutti scanf, gets, o analoghi
- l'esecuzione del programma non richiede così l'interazione con l'utente successivamente alla chiamata

```
./conta_car_instr sequenzadicaratteriditest e ↵
```

acquisizione da riga di comando

- viene rimossa l'interazione con l'utente
- se non si riceve l'argomento da riga di comando, il programma termina
  - tipicamente visualizzando il modo di uso così che l'utente possa capire di aver sbagliato la chiamata
- se l'argomento non rispetta vincoli non è possibile richiedere l'argomento
  - tipicamente il programma controlla che i valori siano compatibili e non procede nell'elaborazione se così non è, visualizzando il modo di uso del programma

esempio che conosciamo

```
gcc -o conta_car_instr contacaratteri.c
```

- questo caso è ancora più sofisticato perché è possibile cambiare l'ordine con cui forniamo gli argomenti .. ma questo è qualcosa di più avanzato
- ci limitiamo a mettere gli argomenti nell'ordine previsto

argc e argv

- il sottoprogramma main riceve in ingresso
  - un array di indirizzi di caratteri: `char * argv[]`
  - il numero di elementi presenti nell'array: `int argc`

```
./testarg sequenza 123 x
```

- tutti i caratteri presenti sulla riga di comando vengono memorizzati in un unico array di caratteri, gestito a stringa (terminatore in fondo)

.	/	t	e	s	t	a	r	g		s	e	q	u	e	n	z	a		1	2	3		x	\0

e mettendo un terminatore al posto di ogni spazio

.	/	t	e	s	t	a	r	g	\0	s	e	q	u	e	n	z	a	\0	1	2	3	\0	x	\0

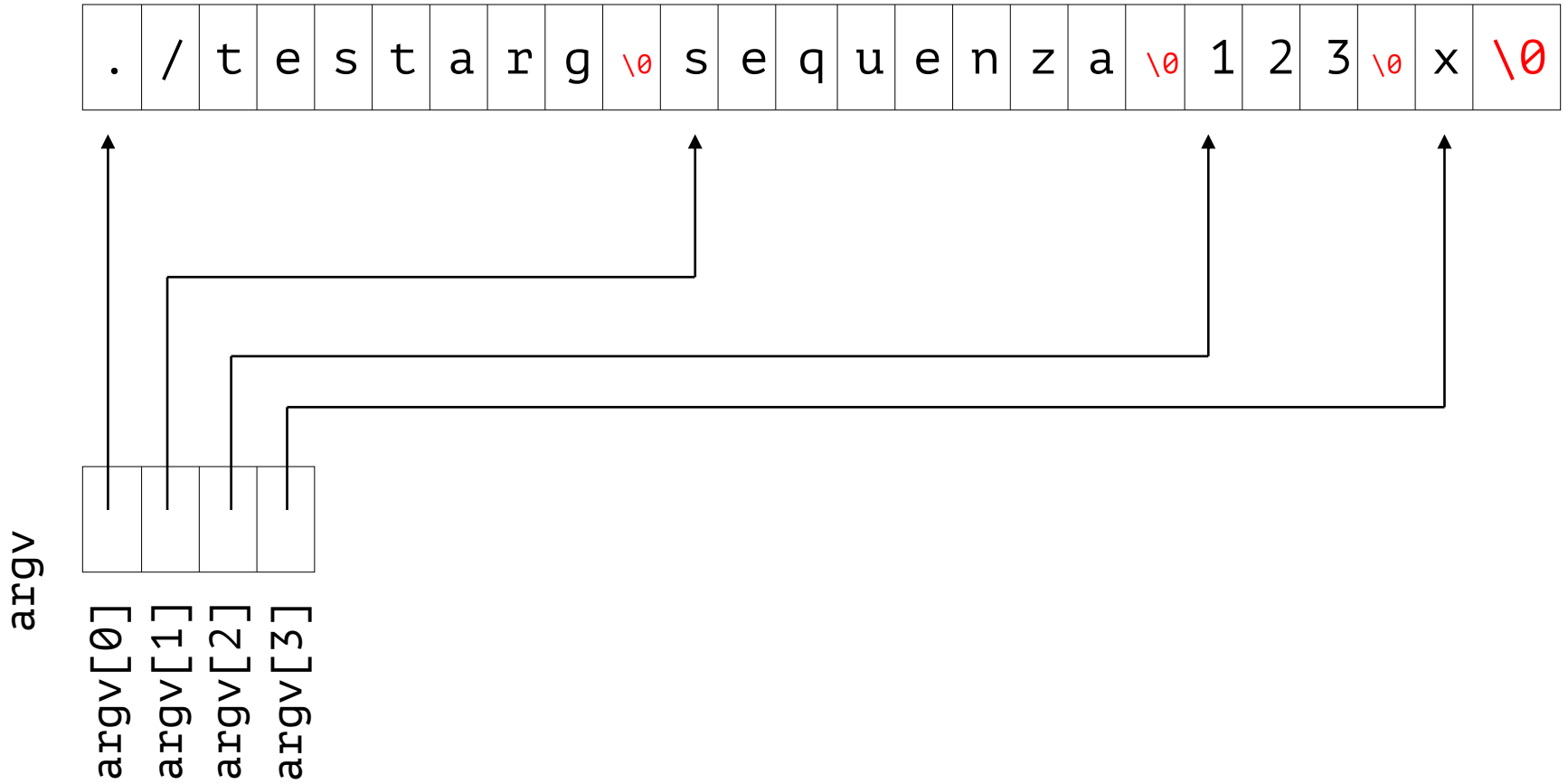
C.Bolchini

argc e argv

.	/	t	e	s	t	a	r	g	\0	s	e	q	u	e	n	z	a	\0	1	2	3	\0	x	\0

- argc: 4
  incluso il nome del programma stesso
  (che c'e' sempre, per cui argc >= 1 sempre)
  e mettendo un terminatore al posto di ogni spazio
- argv[]: 4 indirizzi
  un array di 4 riferimenti a caratteri (gestiti a stringa, per cui con '\0' alla fine)

argv



argv

- tutti gli argomenti acquisiti sulla riga di comando sono stringhe

  nel caso si tratti di **valori numerici** è necessario calcolare il corrispondente valore numerico

  sottoprogrammi di libreria (libreria: stdlib.h)

  int atoi(char []);

  riceve in ingresso una stringa (che si presuppone contenere caratteri numerici) e restituisce l'intero corrispondente

  float atof(char []);

  riceve in ingresso una stringa (che si presuppone contenere caratteri numerici) e restituisce il valore reale corrispondente

argv

- tutti gli argomenti acquisiti sulla riga di comando sono stringhe

  nel caso si tratti di **singoli caratteri** è necessario riferirsi al carattere

  - è il primo carattere di una stringa cui fa riferimento l'argomento

    argv[i][0]

  - è il contenuto all'indirizzo cui fa riferimento l'argomento

    ***(argv[i])**

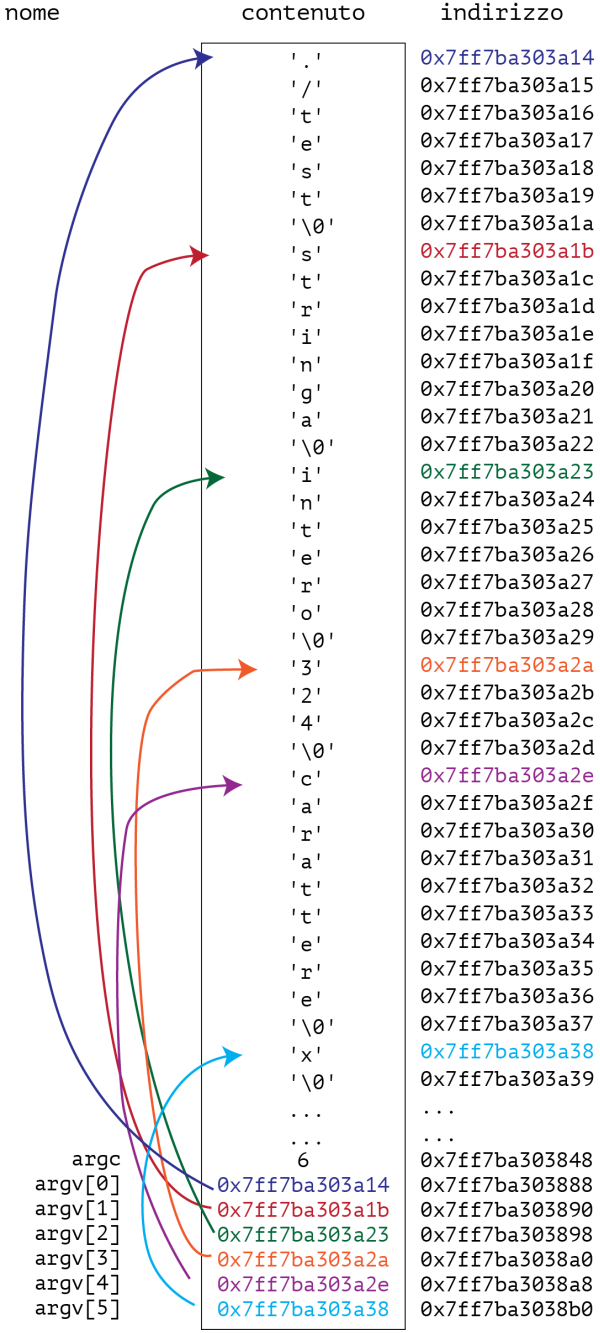
C.Bolchini

```c
#include <stdio.h>
#include <stdlib.h> /* per il sottoprogramma atoi */

int main(int argc, char * argv[])
{
  int i, j;
  char * str_in_ingresso;
  /* variabili per accedere agli argomenti con nomi più espressivi */
  /* per le stringhe non serve dichiarare spazio ... c'è già in argv */
  char * seq; /* riferimento alla stringa in ingresso */
  int val;    /* valore intero dalla stringa in ingresso */
  char c;     /* carattere dalla stringa in ingresso */

  /* visualizzo il contenuto in memoria */
  for(i = 0; i < argc; i++){
    /* la prossima stringa */
    str_in_ingresso = argv[i];
    /* per ogni elemento della stringa sulla riga di comando */
    for(j = 0; str_in_ingresso[j] != '\0'; j++)
      /* visualizza il suo indirizzo in memoria e il carattere contenuto */
      printf("%p %c\n", &str_in_ingresso[j], str_in_ingresso[j]);
    /* visualizzo io il terminatore */
    printf("%p \\0\n", &str_in_ingresso[j]);
  }
  /* omesso controllo cardinalità */
  seq = argv[1];
  str_in_ingresso = argv[3]; /* riferito al 4 argomento che e' un intero */
  val = atoi(str_in_ingresso);
  str_in_ingresso = argv[5]; /* riferito al 6 argomento che e' un carattere */
  c = str_in_ingresso[0];
  printf("%s %d %c\n", seq, val, c);
  /* alternativa senza passare per la variabile di supporto str_in_ingresso */
  val = atoi(argv[3]);
  c = *(argv[5]); /* il contenuto nel primo elemento della stringa */
  printf("%s %d %c\n", seq, val, c);
  c = argv[5][0]; /* non è un array bidimensionale ma fa il calcolo dell'indirizzo */
  printf("%s %d %c\n", seq, val, c);
  /* visualizzo in memoria argc argv ... posizione e contenuto  */
  printf("%p %d\n", &argc, argc);
  for(i = 0; i < argc; i++)
    printf("%p %p\n", &argv[i], argv[i]); /* contenuto è un indirizzo  */
  return 0;
}
```

```
%> ./test stringa intero 324 carattere x
```



argv

variabili di supporto con nomi evocativi

```
int main(int argc, char * argv)
{
   char * st_in          /* stringa in ingresso */
   char * val_str        /* stringa con l'intero */
   int val;              /* il valore intero corrispondente */
   char * car_str;       /* stringa con il carattere */
   char c;               /* il carattere */

   if(argc == 4){ /* se ci sono tutti gli argomenti */
         st_in = argv[1]; /* st_in si riferisce alla stringa */
         val_str = argv[2];
         val = atoi(val_str);
         car_str = argv[3];
         car = *car_str;  /* oppure car = car_str[0]; */
         /* elaborazione */
         . . .
   } else
         printf("parametri non corretti\n");
}
```

```
./testarg sequenza 123 x
```

l'uso delle variabili di supporto aumenta leggibilità, soprattutto nel momento in cui vengono utilizzate nel main per l'elaborazione.

argv

versione senza variabili di supporto

```
int main(int argc, char * argv)
{
   int val;       /* il valore intero corrispondente */
   char c;        /* il carattere */

   if(argc == 4){ /* se ci sono tutti gli argomenti */
         val = atoi(argv[2]);
         car = *argv[3];  /* oppure car = argv[3][0]; */
         /* elaborazione */
         . . .
   } else
         printf("parametri non corretti\n");
}
```

```
./testarg sequenza 123 x
```

anche le variabili **val** e **car** sono non strettamente necessarie.

flusso principale

```
int main(int argc, char * argv[])
{
  if(argc == _numero_atteso_){
    /* utilizzo degli argomenti */
    /* e flusso principale */

  } else {
     printf("argomenti non coerenti\n");
     /* visualizzo come usare il programma */
     usage();
  }
}
```

sottoprogramma che mostra l'help

modo d'uso di un programma ed opzioni

```
void usage(char progname[])
{
   printf("utilizzo: %s seq car", progname);
   printf("\n");
   printf("seq : la sequenza di caratteri da analizzare\n");
   printf("car : il carattere da cercare/contare\n");
}
```

fattoriale

scrivere un programma che acquisito un valore intero da riga di comando calcola e visualizza il suo fattoriale

```
int fattoriale(int);
int main(int argc, char * argv[])
{
  int val, ris;

  if(argc == 2){
       val = atoi(argv[1]);
       ris = fattoriale(val);
       printf("%d! = %d\n", val, ris);
  } else
       printf("argomenti non coerenti\n");
  return 0;
}
```

```
./fattoriale 5
```

anagramma

scrivere un programma che acquisite due stringhe visualizza 1 se una è l'anagramma dell'altra 0 altrimenti (due stringhe identiche non sono una l'anagramma dell'altra)

```
int anagramma(char [], char []);
int main(int argc, char * argv[])
{
  char * seq1, *seq2;
  int ris;

  if(argc == 3){
       seq1 = argv[1];
       seq2 = argv[2];
       ris = anagramma(seq1, seq2);
       printf("%d\n", ris);
  } else
       printf("argomenti non coerenti\n");
  return 0;
}
```

```
./anagramma prato porta
```

- ✓ non è necessario definire quale sarà la dimensione massima delle stringhe.