argc, argv, getopt

acquisizione e gestione di argomenti e opzioni da linea di comando

acquisizione dati da riga di comando

esempio che conosciamo:

```
gcc –o nomeexe nomesrc.c
```

funziona anche:

```
gcc nomesrc.c –o nomeexe
```

significato:

```
-o _nome_del_file_da_creare
```

dall'help (gcc –help)

```
-o <file>                 Write output to <file>
```

1. argomenti in ingresso al programma

si scrivono i dati di ingresso del programma direttamente sulla riga di comando

il programma non interagisce con l'utente ma riceve le informazioni necessarie all'avvio (no `scanf`/`gets`)

esempi:

```
./estrai ./dati.txt ./risultati.txt

./fattoriale 7
```

2. argomenti in ingresso: argv e argc

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

…
numero

   return 0;
}
```

numero di argomenti in ingresso

gli argomenti in ingresso

…

un array di dati

3. argv

`argv` : serie di *stringhe* (sequenza di caratteri, delimitata da spazio, tab, a-capo, …)

il primo argomento è SEMPRE il nome del programma (completo di percorso …) > è la prima cosa che si scrive!

`./fattoriale` `7`

`./estrai` `dati.txt` `risultati.txt`

4. `argc`

`argc`: il numero di argomenti

SEMPRE >= 1

./fattoriale 7 `argc : 2`

./estrai dati.txt risultati.txt `argc : 3`

5. esempio

```
#include <stdio.h>

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

   for(i = 0; i < argc; i++)
        printf("arg %d: %s\n", i, argv[i]);
    printf("totale: %d argomenti\n", argc);

   return 0;
}
```

6. esempio in esecuzione

```
> ./show numero argomenti in ingresso
arg 0: ./show
arg 1: numero
arg 2: argomenti
arg 3: in
arg 4: ingresso
totale: 5 argomenti
>
```

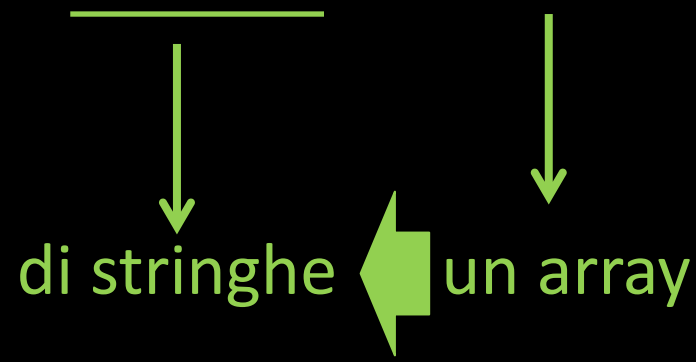
7. argv

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

   int i;

   for(i = 0; i < argc; i++)
        printf("arg %d: %s\n", i, argv[i]);
    printf("totale: %d argomenti\n", argc);

   return 0;
}
```



8. argv

```
> ./show numero argomenti in ingresso

argv[0] ~ "./show"
argv[1] ~    "numero"
argv[2] ~    "argomenti"
argv[3] ~    "in"
argv[4] ~    "argomenti"

   > ./fattoriale 7
argv[0] ~ "./fattoriale"
argv[1] ~    "7"
```

stringa!

8. argv

gli argomenti sono acquisiti come stringhe, se hanno una natura diversa devono essere convertiti

```
> ./fattoriale 7
   argv[0] ~ "./fattoriale"
   argv[1] ~    "7"
```

```
int n;
n = atoi(argv[1]);
```

9. argv

gli argomenti sono stringhe: c'è memoria allocata

```
> ./estrai dati.txt risultati.txt

argv[0] ~ "./estrai"
argv[1] ~    "dati.txt"
argv[2] ~    "risultati.txt"

char * nomesrc, * nomedst;
nomesrc = argv[1];
nomedst = argv[2];
```

10. sottoprogrammi "tipici"

```
/* mostra errore se parametro mancante */
void showProgMissingArg(char *);

/* mostra come chiamare il programma */
void showProgUsage(char *);
```

11. sottoprogrammi "tipici"

```
void showProgMissingArg(char * progname)
{
      printf("%s: argomento mancante\n", progname);
      return;
}

void showProgUsage(char * progname)
{
      printf("Uso: %s VALORE_INTERO_NON_NEGATIVO\n”,
progname);
      return;
}
```

12. sottoprogrammi "tipici"

```
> ./fattoriale
./fattoriale: argomento mancante
Uso: ./fattoriale VALORE_INTERO_NON_NEGATIVO

> ./fattoriale 5
120
```

eventalmente sistemare per togliere "./" ...

sottoprogramma che riceve in ingresso una stringa che contiene il nome di un programma, comprensivo di percorso e restituisce una nuova stringa che contiene solo il nome dell'eseguibile

13. argomenti in ordine qualsiasi / opzioni

```
gcc -o nomeexe nomesrc.c

gcc nomesrc.c -o nomeexe
```

l'ordine può essere qualsiasi purchè ci sia un modo per capire quale argomento significa cosa: utilizzo delle opzioni

- opzione brevi: `-o`
- opzione estesa/lunga: `--output`

13. opzioni: esempio

```
> ./frequency -h
Usage: frequency [options] INPUTFILE
 -h  --help            Displays this information
 -a  --alpha           Use only alphanumerical
 -t  --threshold VALUE  The threshold for results
 -o  --output FILENAME Write output to file
```

14. opzioni: esempio

```
> ./frequency -t 23 -o ./ris.txt ./dati.txt

int main (int argc, char* argv[]){
  int thresh;
  int blnAlpha; /* 1: only alpha, 0: all */
  char * nomesrc, * nomedst;
  char * nomeexe = argv[0];
  …
```

15. opzioni

gestione manuale

- analisi di argv
  - coppie opzione / valore
    - » `-t 23`

```
argv[i] ~ "-t"
argv[i+1] ~ "23"
```

16. opzioni

utilizzo di sottoprogrammi di libreria :

```
getopt
getopt_long
```

richiede la definizione delle opzioni supportate

- opzione (breve e estesa)
- facoltativa / obbligatoria

analizza gli argomenti e identifica le coppie

permette un agile accesso alle opzioni e ai corrispondenti valori

17. getopt

```
int main (int argc, char* argv[]){
    ...
    /* stringa che specifica opzioni brevi valide  */
    const char* short_options = "hat:o:";
    /* che descrive le opzioni lunghe  */
    const struct option long_options[] = {
       { "help",      0, NULL, 'h' },
       { "alpha",     0, NULL, 'a' },
       { "threshold", 1, NULL, 't' },
       { "output",    1, NULL, 'o' },
       { NULL,        0, NULL, 0}  /*req end of array */
    };
```

18. getopt_long

```
...
  while ((next_option = getopt_long (argc, argv,
            short_options, long_options, NULL) != -1) {
      /* analisi dei casi */
      if(next_option == 'h'){
            /* visualizza help */
            print_usage(stdout, nomeexe, 0);
      } else if (next_option == 't') {
            /* c'è un valore dopo l'opzione, lo si usa
*/
            thresh = atoi(optarg);
      } else if (next_option == 'a') {
         /* non c'è valore */
            blnAlpha = 1;
```

19. getopt_long

```
...
    } else if (next_option == 'o') {
    /* c'è un valore dopo l'opzione, lo si usa */
          nomedst = optarg;
    } else if (next_option == '?') {
    /* opzione non prevista */
          print_usage(stderr, nomeexe, 1);
    } else {
    /* situazione inattesa */
          abort();  /* return -1; */
    }
```

20. getopt

```
...
  while ((next_option = getopt(argc, argv,
           short_options) != -1) {
      /* analisi dei casi */
      if(next_option == 'h'){
            /* visualizza help */
            print_usage(stdout, 0);
      } else if (next_option == 't') {
            /* c'è un valore dopo l'opzione, lo si usa */
            thresh = atoi(optarg);
      ...
```

```
while ((next_option = getopt(argc, argv,"hat:o:") != -1)
{
```

21. print_usage

```
void print_usage (FILE* fp, char * pname, int exit_code)
{
      fprintf (fp, "Usage: %s [options] INPUTFILE\n",
pname);
      fprintf (fp,
   "  -h  --help              Displays this information\n"
   "  -a  --alpha             Use only alphanumerical\n"
   "  -t  --threshold VALUE   The threshold for results\n"
   "  -o  --output FILENAME   Write output to file\n");
      exit (exit_code);
}
```