# Arguments d'un programme

*L2 SPI, Bruno Jacob*

## 1 Arguments

Quand on appelle un programme dans un environnement UNIX ou MS-DOS, on compose une ligne de commandes de la forme :

```
nom-du-programme argument1 argument2 ...
```

L'exécution d'un programme commence par la fonction `main` . Tous se passe alors comme si le Système d'Exploitation avait appelé la fonction `main` comme une fonction ordinaire. Voici la signature habituelle de cette fonction :

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

avec

**argc** (pour *argument counter*) : nombre d'arguments de la ligne de commande y compris le nom du programme

**argv** (pour *argument value*) : liste des arguments de la ligne de commande. Ceux ci sont considérés comme des chaînes de caractères.

**env** (pour *environment*) : liste des variables d'environnement et leur valeur.

Vous pouvez bien sur changer les noms des paramètres `argc` , `argv` et `env` . En général, si on ne se sert pas des variables d'environnement, alors on ne déclare pas le praramètre `env` .

Exemple :

Si d'une part, dans le programme `toto` on veut capturer les arguments de la ligne de commande alors

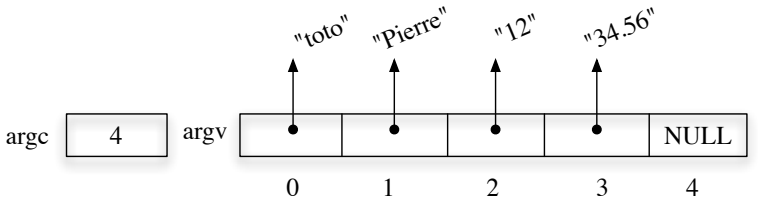
- on doit utiliser `stdio.h`
- la signature de sa fonction `main` dans le fichier source `toto.c` sera

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

et si, d'autre part, le programme est activé par la ligne de commandes

```
toto Pierre 12 34.56
```

alors les arguments de la ligne de commandes sont transmis aux paramètres de la fonction `main` de la manière suivante :




- La place mémoire du tableau `argv` est réservée
- le $1^o$ argument ( `argv[0]` ) est toujours rempli et affecté par le nom de l'exécutable (ou du programme ou de la commande)
- les autres éléments pointent sur les arguments. La place mémoire que prend la chaîne de caractères pour stocker chaque argument est réservée
- en principe, le dernier élément de `argv` (après le dernier argument donc) est à `NULL` .

`arguments_main.c , arguments_main_map.c`

# 2 Options courtes

Les options sont des arguments optionnels de la ligne de commandes. Ils peuvent apparaître à n'importe quelle place ou pas du tout. Généralement, pour distinguer les options des arguments obligatoires on les fait précéde par "-" ou "–". Les options peuvent être simplement un indicateur (ou flag) ou avoir une valeur.

Exemple :
si on tape la ligne de commande

```
nom-du-programme -w arg1 -x=toto arg2 -y=10 arg3 -z
```

alors on aura :

**liste des arguments** : `arg1` , `arg2` et `arg3`

**liste des options** : `w` sans valeur, `x` avec la valeur "toto" , `y` avec la valeur "10" et `z` sans valeur

Pour capturer ces options et leurs valeurs, sans les confondre avec les arguments, il faut analyser la chaîne de caractères des arguments du `main` . Dans ce cas on fait généralement appelle à la fonction `getopt` .

## 2.1 Synopsis

```
#include <unistd.h>

extern int optarg ;
extern int optind;
extern int optopt;
extern int opterr;
extern int optreset;

int getopt(int argc, char * const argv[], const char *optstring);
```

- `argc` et `argv` sont les mêmes que ceux du `main` : ce sont le nombre et la liste d'arguments qui sont transmis à la fonction `main()` lors du lancement du programme.
- `optsring` contient les noms des options (1 caractère)

## 2.2 Description

La fonction `getopt()` analyse les arguments de la ligne de commande. Tout élément de `argv` qui commence par "-" est considéré comme une option. Les caractères à la suite du "-" initial sont les caractères de l'option. Si `getopt()` est appelée à plusieurs reprises, elle renverra successivement chaque caractère de chaque option.

`optstring` est une chaîne contenant l'ensemble des caractères d'option autorisés. Si un de ces caractères est suivi par un deux-point (":") alors l'option nécessite un argument supplémentaire. Celui-ci sera placé dans `optarg` .

Remarques

1. les options doivent se trouver entre le nom du programme et les arguments
2. les indices des arguments dans `argv` ne bougent pas

### 2.3 Valeurs renvoyées

`getopt()` renvoie
- le caractère de l'option si une option est trouvée
- `-1` si toutes les options de la ligne de commande ont été analysées
- `"?"` si un caractère d'option ne se trouve pas dans `optstring` . La variable externe `optopt` contient alors ce caractère inconnu
- si une valeur manque à une option
  - `":"` si premier caractère de `optstring` est ":"
  - `"?"` sinon

`options_courtes.c`

## 3 Options longues

Les options avec `getopt` sont ici réduites à un seul caractère, et on peut trouver que ce n'est pas très explicite ou lisible, on peut alors utiliser les fonctions `getopt_long()` et `getopt_long_only()` .

### 3.1 Synopsis

```
#include <getopt.h>

extern char *optarg;
extern int optind;
extern int optopt;
extern int opterr;
extern int optreset;

int getopt_long(int argc, char * const *argv, const char *optstring,
                const struct option *longopts, int *longindex);

int getopt_long_only(int argc, char * const *argv, const char *optstring,
                     const struct option *longopts, int *longindex);
```

### 3.2 Description

La fonction `getopt_long()` fonctionne comme `getopt()` sauf qu'elle accepte également des noms longs d'option, commençant par deux tirets. Les noms longs d'option peuvent être abrégés si l'abréviation est unique. Une

option longue peut prendre un argument, de la forme `--arg=param` ou `--arg param` .

`longopts` est un pointeur sur le premier élément d'un tableau de structures struct option déclarées dans `<getopt.h>` ainsi :

```
struct option {
    const char *name;
    int         has_arg;
    int        *flag;
    int         val;
};
```

La signification des différents champs est la suivante :

- `name` est le nom de l'option longue.
- `has_arg` vaut :

  `no_argument` (ou 0) si l'option ne prend pas d'argument

  `required_argument` (ou 1) si l'option prend un argument

  `optional_argument` (ou 2) si l'option prend un argument optionnel
- `flag` spécifie la manière de renvoyer les résultats pour une option longue ( en général à `NULL` )
- `val` est la valeur à renvoyer, ou à charger dans la variable pointée par `flag` .

Le dernier élément de la table doit être rempli avec des zéros.

Si `longindex` n'est pas NULL, il pointe vers une variable qui est définie avec l'index de l'option longue correspondant à longopts.

`getopt_long_only()` fonctionne comme `getopt_long()` , mais "-" tout comme "–" indiquent une option longue. Si une option commençant par "-" (et non "–") ne correspond pas à une option longue, mais correspond à une option courte, elle est analysée en tant qu'option courte.

Remarques

1. les options peuvent se trouver n'importe où dans la ligne de commandes
2. les indices des arguments dans `argv` débutent à 0
3. ces fonctions sont dans les objets `getopt.o` et `getopt1.o` . Ils sont souvent déjà installés mais si ce n'est pas le cas on peut trouver facilement les sources et les compiler.

## 3.3 Valeurs renvoyées

`getopt_long()` et `getopt_long_only()` renvoient également le caractère d'option courte s'ils en trouvent une. Pour les options longues, ils renvoient `val` si `flag` vaut `NULL` , et 0 sinon. Les erreurs et la fin des options sont gérées comme avec `getopt()` , en renvoyant de surcroît `"?"` pour une correspondance ambiguë, ou un paramètre en trop.

`options_longues.c`