Projet de Programmation

Benoit Donnet

Année Académique 2023 - 2024

Agenda

Partie 1: Techniques Avancées de C

- Chapitre 1: Définition de Type
- Chapitre 2: Généricité
- Chapitre 3: Arguments d'un Programme

Agenda

- Chapitre 3: Arguments d'un Programme
  - Variables d'Environnement
  - Options Courtes
  - Options Longues

Agenda

- Chapitre 3: Arguments d'un Programme
  - Variables d'Environnement
    - ✓ Principe
    - ✓ `main()`
    - ✓ `getenv()`
  - Options Courtes
  - Options Longues

Principe

- **Variable d'environnement**?
  - valeur dynamique chargée en mémoire pouvant être utilisée par plusieurs processus fonctionnant simultanément
- Utilité?
  - résumé d'informations nécessaires
  - exemple
    - ✓ sur la plupart des OS, les emplacements de certaines librairies (ou exécutables) se trouvent à des endroits différents
    - ✓ une variable d'environnement peut déterminer où se trouvent ces librairies (ou exécutables)

Principe (2)

- Exemple
  - chemin d'accès vers les librairies/programmes

```
$>echo $PATH
/usr/local/bin:/usr/local/share/python:/Users/
benoit/.local/bin:/opt/local/bin:/opt/local/sbin:/opt/
local/bin:/opt/local/sbin:/opt/local/bin/:/Users/
benoit/.local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/
local/bin:/opt/X11/bin:/usr/texbin
```

- Il existe plein de variables d'environnement
  - `$HOME`
  - `$PWD`
  - `$LANG`
  - `$LD_LIBRARY_PATH`
  - ...

main()

- Il est possible, depuis un programme C, d'accéder aux variables d'environnement
- Comment?

```
int main(int argc, char *argv[], char *env[]){
  //le programme
}//fin programme
```

tableau contenant toutes les variables d'environnement

main() (2)

- Exemple

```
#include <stdio.h>

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

  for(int i=0; env[i]!=NULL; i++)
    printf("%s\n", env[i]);

  return 0;
}//fin programme
```

main() (3)

- Compilation et exécution

```
$>gcc -o main var_env.c
$>./main
LC_MONETARY=fr_BE.utf-8
TERM_PROGRAM=iTerm.app
TERM=xterm-color
SHELL=/bin/bash
TMPDIR=/var/folders/0n/l7r6bq2j77dg4rf4q5trmyxm0000gn/T/
Apple_PubSub_Socket_Render=/tmp/launch-HQ9zhO/Render
LC_NUMERIC=fr_BE.utf-8
OLDPWD=/Users/benoit/Enseignement/ULg/INFO0030/Slides
...
```


getenv()

- On peut, bien entendu, accéder à une variable particulière
  - `char *getenv(const char *name);`
- Retourne une chaîne de caractères
  - correspondant à la valeur de la variable d'environnement passée en paramètre
  - `NULL` si la variable n'existe pas

getenv() (2)

- Exemple

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]){
  char *login = getenv("USER");

  if(login!=NULL)
    printf("Hello %s!\n", login);

  return 0;
}//fin programme
```


Agenda

- Chapitre 3: Arguments d'un Programme
  - Variables d'Environnement
  - Options Courtes
    - ✓ Principe
    - ✓ `getopt()`
    - ✓ Règles
    - ✓ Utilisation
  - Options Longues

Principe

- Si un argument (du programme) commence par `-`, c'est une option
- Il existe deux types d'options
  - **option courte**
    - ✓ un seul caractère
  - **option longue**
    - ✓ chaîne de caractères
- Si on a `-xyz`, alors `x`, `y` et `z` désignent des options courtes

Principe (2)

- Exemple (option courte)

```
$>gcc -o main args.c
$>./main -hi
```

getopt()

- Fonction définie dans `unistd.h`
- Objectifs?
  - analyser les arguments d'un programme
  - ignorer leur ordre
- Mal porté sous Windows

getopt() (2)

- Format de `getopt()`

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

Arguments du programme

Format attendu des arguments

Le caractère lu

- Si un caractère d'option est lu, `getopt()` le renvoie
- Sinon
  - `EOF` si plus d'options
  - gestion d'erreur (cfr. slide 20)

Règles

- Une variable particulière doit être déclarée et utilisée pour définir les options (courtes) autorisées
  - `optstring`
- Format
  - `x:`
    - ✓ indique que l'option x a un paramètre
  - `x::`
    - ✓ indique que l'option x a un paramètre *optionnel*

Règles (2)

- Exemple

```
int main(int argc, char **argv){
  /*
   * -h -> help
   * -i input
   * -o [output]
   */
  char *optstring = "hi:o::";

  //code...

  return 0;
}//fin programme
```

Règles (3)

- Le paramètre d'une option doit être introduit
  - soit par un espace
    - ✓ `-i xxx`
      - l'argument est `xxx`
  - soit par rien
    - ✓ `-ixxx`
      - l'argument est `xxx`
    - ✓ `-i=xxx`
      - l'argument est `=xxx`
- Dans le cas d'un paramètre optionnel
  - seule la forme `-ixxx` est acceptée

Règles (4)

- Si `optstring` commence par `:`, `getopt` renverra
  - `?` en cas d'option inconnue
  - `:` en cas d'argument manquant
- La variable externe `int optopt` contient le caractères d'option courant
- La variable externe `char *optarg` contient l'argument de l'option courante

Utilisation

```c
#include <stdio.h>
#include <unistd.h>

int main(int argc, char **argv){
  /*
   * -h -> help
   * -i input
   * -o [output]
   */
  const char *optstring = ":hi:o::";
  int val;

  //à suivre

}//fin programme
```


Utilisation (2)

```c
int main(int argc, char **argv){
  //cfr. slide précédent
  while((val=getopt(argc, argv, optstring))!=EOF){
     switch(val){
          case 'h':
               printf("help\n"); break;
          case 'i':
               printf("input: %s\n", optarg); break;
          case 'o':
               printf("output: %s\n", optarg); break;
          case '?':
               printf("unknown option: %c\n", optopt); break;
          case ':':
               printf("missing arg: %c\n", optopt); break;
     }//fin switch
  }//fin while

  return 0;
}//fin programme
```

Utilisation (3)

- Utilisation

```
$>gcc -o main args.c
$>./main
$>./main -hi
help
missing arg: i
$>./main -olulu -h -i toto
output: lulu
help
input: toto
$>./main -olulu -h -i -y -z -oh
output lulu
help
input -y
Unknown option: z!
output h
```


Agenda

- Chapitre 3: Arguments d'un Programme
  - Variables d'Environnement
  - Options Courtes
  - Options Longues
    - ✓ Principe
    - ✓ `getopt_long()`
    - ✓ Règles
    - ✓ Utilisation

Principe

- Les options longues sont introduites par `--`
  - chaîne de caractères désignant l'option
- Les arguments (optionnels) peuvent être introduits
  - soit avec un =
    - ✓ `--input=xyz`
  - soit avec un espace
    - ✓ `--input xyz`
- Exemple

```
$>gcc -o main args_long.c
$>./main --help --input=toto.txt
```


`getopt_long()`

- Fonction définie dans `getopt_h`
- Objectifs?
  - analyser les arguments longs d'un programme
  - permettre aussi de prendre en compte les arguments courts
- Mal porté sous Windows

getopt_long() (2)

- Format de `getopt_long()`

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

Arguments du programme
Format attendu des arguments courts
Structure décrivant les arguments longs
Indice de l'option dans le tableau
Le caractère d'option lu

getopt_long() (3)

- `getopt_long()` renvoie le caractère d'option
- Si `longindex` ≠ `NULL` et si une option longue est trouvée
  - Alors `*longindex` vaut l'indice de l'option dans le tableau

Règles

- Structure décrivant les options

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

- `has_arg` peut prendre 3 valeurs
  - `0=no_argument`
  - `1=required_argument`
  - `2=optional_argument`
- Si `flag` vaut `NULL`
  - Alors `getopt_long()` renvoie `val`
  - Sinon `getopt_long()` renvoie 0 et `*flag` vaut `val`
- `val` est la valeur à renvoyer

Règles (2)

- `getopt_long()` accepte aussi les options courtes

```
const char *optstring=":hi:o::";
const struct option lopts[] = {
  {"help", no_argument, NULL, 'h'},
  {"input", required_argument, NULL, 'i'},
  {"output", optional_argument, NULL, 'o'},
  {NULL, no_argument, NULL, 0},
};
```

Utilisation

```c
#include <stdio.h>
#include <getopt.h>

int main(int argc, char **argv){
  //options courtes identiques à getopt()
  const char *optstring = ":hi:o::";
  const struct option lopts[] = {
    {"help", no_argument, NULL, 'h'},
    {"input", required_argument, NULL, 'i'},
    {"output", optional_argument, NULL, 'o'},
    {NULL, no_argument, NULL, 0},
   };

  //à suivre
}//fin programme
```


Utilisation (2)

```c
int main(int argc, char **argv){
  //cfr. slide précédent
  int val, index = -1;

  while(EOF!=(val = getopt_long(argc, argv, optstring,
  lopts, &index))){
    char msg[64];

    if(index == -1)
      sprintf(msg, "option courte -%c : ", val);
    else
      sprintf(msg, "option longue --%s : ",
      lopts[index].name);

  //à suivre
}//fin programme
```

Utilisation (3)

```
int main(int argc, char **argv){
  //cfr. slide précédent
    switch(val){
      case 'h':
        printf("%s\n", msg); break;
     case 'o':
        printf("%s arg=%s\n", msg, optarg); break;
     case 'i':
        printf("%s arg=%s\n", msg, optarg); break;
     case ':':
        printf("missing arg: %c\n", optopt); break;
     case '?':
        printf("unknown option: %c\n", optopt);
    }//fin switch
    index = -1;
   }//fin while

 return 0;
}//fin programme
```


Utilisation (4)

- Utilisation

```
$>gcc -o main args_long.c
$>./main
$>./main -hi
option courte -h
missing arg: i
$>./main --input
missing arg: i
$>./main --input toto.txt -ilulu.txt
option longue --input : arg=toto.txt
option courte -i : arg=lulu.txt
```