



ASTEK



Colle

opt_get

Responsable Astek astek_resp@epitech.eu

Abstract:



Table des matières



.1 Consignes

- Le projet devra être fait en binôme.
- Vous n'avez pas le droit à votre libmy.
- Vous n'avez pas le droit à internet.
- Vous avez le droit d'utiliser les fonctions `malloc`, `free`, `exit`, `write`, `read`, `open`, `close`, `srand`, `rand` et `time`
- Vos exercices doivent être à la norme.
- Binaire et exemple non contractuels (le sujet fait foi).
- Les variables globales sont interdites pour ce projet.
- Les variables statiques sont interdites pour ce projet.
- Vous devez écrire toutes les déclarations de fonctions et de types dans un fichier nommé `opt_get.h` situé à la racine du projet.
- Votre programme sera testé avec nos propres fichiers `main`,
vous ne devez donc pas rendre de fichier ou de fonction `main`.
- Travaillez en local !
C'est-à-dire que pour chaque exercice vous devez le compiler sur votre compte linux.
Ceci dans le simple but de ne pas surcharger les serveurs car vous êtes nombreux.
- Il n'y a pas de rendu sur les serveurs :
`présentez-vous à la soutenance avec votre machine`



.2 Sujet



Si un comportement est indéfini, à vous de le définir, mais restez cohérent ! Et relisez bien le sujet avant de conclure qu'un comportement est indéfini !

- Vous devez coder la fonction `opt_get` (parseur de ligne de commandes) et sa liste de fonctions utilitaires (`opt_isset`, `opt_pos`, `opt_args` et `opt_free`).
- `t_opt* opt_get(int ac, char **av, const char* format);`
 - Cette fonction analyse l'adéquation de la ligne de commande `av` de taille `ac` avec le pattern `format`.
 - Le paramètre `av` de `opt_get` sera le même que le paramètre `av` de votre main. Vous ignorerez donc la première case de `av`.
 - Si la ligne de commande diffère avec `format`, `opt_get` renvoie `NULL`.
 - Si le format est invalide : comportement indéterminé.
 - Si la ligne de commande concorde avec `format`, `opt_get` renvoie un `t_opt*`.
 - `format` est une chaîne de caractères structurée de la façon suivante :
 - Les espaces seront ignorés.
 - Les doublons seront ignorés (`const char *format = "-l -a -l"` sera interprété comme `const char *format = "-l -a"`), aussi bien dans la ligne de commandes que dans `format`.
 - Une option sera toujours préfixée par un `-` ou par un `--`.
 - Une option dans `format` qui est préfixée par un `-` comportera au maximum un caractère (par exemple `const char *format = "-l -a"`).
 - Une option préfixée par un `-` pourra être utilisée dans une ligne de commandes en syntaxe compressée (si `const char *format = "-l -a"`, une ligne de commande de type `-la` ou `-al` sera valide).
 - Une option préfixée par un `--` ne pourra pas être utilisée dans une ligne de commandes en syntaxe compressée.
 - Une option entourée de `[]` sera optionnel (par exemple `const char *format = "-l [-a]"` : `-a` sera optionnelle).
 - Des `[]` ne pourront entourer qu'une et une seule option (`const char *format = "[-l -a]"` est invalide, en revanche `const char *format = "[-l] [-a]"` est valide).



- Les options pourront prendre des arguments, les arguments ne pourront pas être précédés par un - ou par un -- (par exemple `const char *format = "-f FILE1 FILE2"`).
- Une option qui prend des arguments ne pourra pas être utilisée en syntaxe compressée dans la ligne de commandes.
- Les arguments supporteront les [], cependant de la même manière que pour les options, les [] ne pourront entourer qu'un et un seul argument (par exemple `const char *format = "-f [FILE1] [FILE2]"` est valide alors que `const char *format = "-f [FILE1 FILE2]"` est invalide)



`const char *format = "[-f FILE1]"` signifie que -f est facultatif, mais que si -f est présent, FILE1 est obligatoire.

- Un argument nommé "..." sera interprété comment une liste de taille indéfinie d'arguments facultatifs.



`const char *format = "-f FILE1 ..."` signifie par exemple que l'on souhaite avoir au moins un argument à l'option "-f".



L'ordre des options dans format n'est pas contractuel. Si `const char *format = "-l -a"`, une ligne de commande de type `"-a -l"` ou `"-al"` sera valide.



Le contenu de `t_opt` n'est pas précisé volontairement, c'est à vous de le définir.

- `t_bool opt_isset(t_opt *options, const char *option)`



- Cette fonction renvoie 1 si `option` est dans `options` et 0 le cas échéant.
- Le prototypage de `t_bool` (true = 1 et false = 0) est donné :

```
typedef int t_bool;
```



Si `option` contenait des arguments, on ne retiendra que le nom de l'option. Par exemple pour `const char *format = "[-f FILE1 ...]"`, le test à effectuer pour tester la présence de `-f` sera `opt_isset(options, "-f")`.

- `t_pos opt_pos(t_opt *options, const char *option)`

- Cette fonction renvoie la position de `option` par rapport à la ligne de commandes.
- En cas d'erreur `opt_pos` renverra -1.
- Le type de `t_pos` est donné par :

```
typedef int t_pos;
```



>macommand `-f tutu toto -ua -t`, supposons que le format donné à `opt_get` concorde, `opt_pos(options, "-f")` renvoie 0, `opt_pos(options, "-a")` renvoi 2, `opt_pos(options, "-u")` renvoie 1, `opt_pos(options, "-t")` renvoie 3 et `opt_pos(options, "-z")` renvoie -1

- `const char **opt_args(t_opt *options, const char *option)`

- Cette fonction renvoie un tableau contenant les arguments relatifs à `option`.
- Le tableau renvoyé sera terminé par un pointeur NULL.
- En cas d'erreur, `opt_args` renverra NULL.



>macommand `-f tutu toto`, supposons que le format donné à `opt_get` concorde, `opt_args(options, "-f")` renverra { "tutu" , "toto", NULL }



- `void opt_free(t_opt* options)`
 - Cette fonction libérera la mémoire allouée par `options`.
 - En cas d'erreur, cette fonction ne fait rien.

.3 Bonus

- Gestion de la séquence spéciale `--` dans la ligne de commandes, cette séquence permet de considérer tout le reste de la ligne de commande comme des arguments.



```
>macommande -f -- -l -a : ceci sera interprété comme l'option -f avec  
les arguments -l et -a
```

- Gestion des séquences d'arguments multiples dans la ligne de commande.



```
>macommand -f a -f b : ceci sera interprété comme l'option -f avec  
les arguments a et b
```

- Gestion des `regex` dans `format`.



```
man regex
```

- Gestion des erreurs de `format`



.4 Exemple

```
int main()
{
    {
        const char*format = "-f -a";
        char *av[] = {"a.out", "-af"};
        t_opt* res = opt_get(2, av, format);
        // res != NULL car av concorde avec format
    }

    {
        const char*format = "-f ARG -u -a";
        char *av[] = {"a.out", "-u", "-f", "main.c", "-a"};
        t_opt* res = opt_get(5, av, format);
        // res != NULL car av concorde avec format
    }

    {
        const char*format = "-f ARG -u -a";
        char *av[] = {"a.out", "-u"};
        t_opt* res = opt_get(2, av, format);
        // res == NULL car av ne concorde pas avec format
        // les options -f ARG et -a etants obligatoires
    }

    {
        const char*format = "-f [-a]";
        char *av[] = {"a.out", "-f"};
        t_opt* res = opt_get(2, av, format);
        // res != NULL car av concorde avec format
        // l'option -a etant optionnelle
    }

    {
        const char*format = "-f [-a]";
        char *av[] = {"a.out", "-a"};
        t_opt* res = opt_get(2, av, format);
        // res == NULL car av ne concorde pas avec format
        // l'option -f etant obligatoire
    }

    {
        const char*format = "-f [-a] [-u ARG]";
        char *av[] = {"a.out", "-f", "-u"};
        t_opt* res = opt_get(3, av, format);
    }
}
```




```
// res == NULL car av ne concorde pas avec format
// Un argument etant obligatoire en presence de l'option -u
}

{
    const char*format = "-a ...";
    char *av[] = {"a.out", "-a", "file1", "file2", "file3"};
    t_opt* res = opt_get(5, av, format);
    // res != NULL car av concorde avec format
}

{
    const char*format = "-a ...";
    char *av[] = {"a.out", "-a"};
    t_opt* res = opt_get(2, av, format);
    // res != NULL car av concorde avec format
}
}
```