RAPPORT TP1

Rapport de tp

Tables des matières :

Objectifs/Enjeux

Les objectifs de ce TP étaient de prendre de bonnes habitudes en programmation. Notamment ce familiariser avec l'utilisation de la **bibliothèque graphique MLV** mais aussi la documentation **man** qui nous permet de connaître précisément le rôle d'une fonction ou encore revoir l'utilisation des **paramètres d'une fonction** ou de programmer un **système de cryptage**.

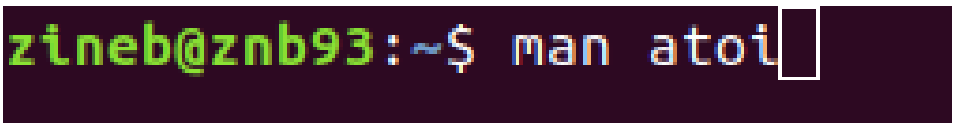
Moyens mis en œuvre

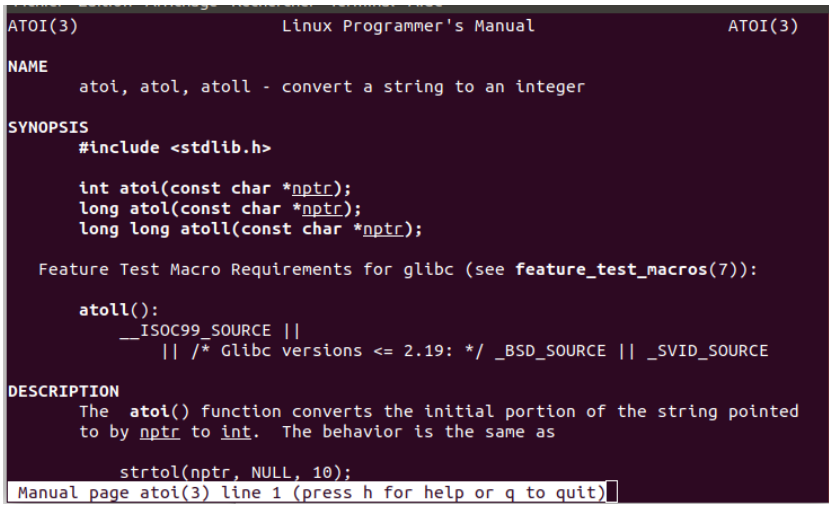
Pour le bon déroulement de ce TP, nous avons eu recours à différentes documentations :

1. La documentation de la bibliothèque graphique MLV ; http://www-igm.univ-mlv.fr/~boussica/mlv/index.html

2. L'utilisation d'un fichier Makefile ;

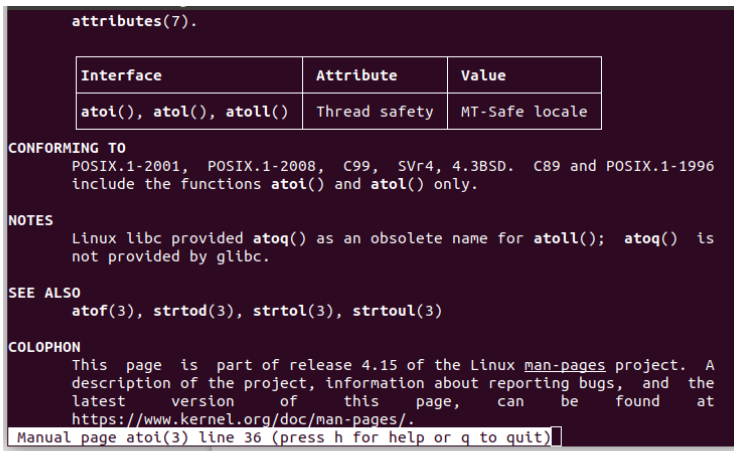
3. La documentation man que l'on utilise sur notre terminal.





```
ATOI(3)                    Linux Programmer's Manual                   ATOI(3)

NAME
       atoi, atol, atoll - convert a string to an integer

SYNOPSIS
       #include <stdlib.h>

       int atoi(const char *nptr);
       long atol(const char *nptr);
       long long atoll(const char *nptr);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       atoll():
           __ISOC99_SOURCE ||
               || /* Glibc versions <= 2.19: */ _BSD_SOURCE || _SVID_SOURCE

DESCRIPTION
       The  atoi() function converts the initial portion of the string pointed
       to by nptr to int.  The behavior is the same as

           strtol(nptr, NULL, 10);
 Manual page atoi(3) line 1 (press h for help or q to quit)
```

Documentation de la commande atoi grâce au man



```
       attributes(7).

       | Interface                | Attribute     | Value          |
       | atoi(), atol(), atoll()  | Thread safety | MT-Safe locale |

CONFORMING TO
       POSIX.1-2001,  POSIX.1-2008,  C99,  SVr4, 4.3BSD.  C89 and POSIX.1-1996
       include the functions atoi() and atol() only.

NOTES
       Linux libc provided atoq() as an obsolete name for atoll();  atoq()  is
       not provided by glibc.

SEE ALSO
       atof(3), strtod(3), strtol(3), strtoul(3)

COLOPHON
       This  page  is  part of release 4.15 of the Linux man-pages project.  A
       description of the project, information about reporting bugs,  and  the
       latest     version     of     this     page,     can     be     found     at
       https://www.kernel.org/doc/man-pages/.
 Manual page atoi(3) line 36 (press h for help or q to quit)
```

Questions du TP

Fichier : exo1.c

Question 1

*int strcoll(const char *s1, const char *s2);*

La fonction strcoll() compare deux chaînes de caractères (s1 et s2), renvoie :
- un entier négatif si s1 < s2;
- un entier nul si s1 = s2:
- un entier positif si s1 > s2;

Elle compare en fonction de la localisation en cours pour la catégorie LC_COLLATE (la mise en correspondance et le classement des chaînes de caractères).

Question 3

*char *strchr(const char *s, int c);*

La fonction strchr() prend en argument une chaîne de caractère et la valeur ASCII du caractère recherché dans la chaîne. Renvoie un pointeur sur le caractère correspondant, ou NULL si le caractère n'a pas été trouvé.

Question 5

*int atoi(const char *nptr);*

La fonction atoi a comme argument une chaîne de caractère.
Si la chaîne de caractère est composé de chiffres/nombres en début de chaîne, celle-ci sera convertie en un entier qui aura pour valeur les nombres/chiffres trouvés en début de chaîne.

Fichier : exo4.c

Question 1

En calculant l'expression '-9 % 5' on obtient comme résultat 1, cela nous pose problème car dans le code ASCII cela ne correspond pas à un caractère.

Fonctions principales

int compare(char *s1, char *s2) ;

La fonction prend en argument s1 et s2 (deux chaînes de caractères), permet de comparer ces deux chaînes pour retourner un entier qui indiquera si la première chaîne est plus grande, plus petite ou égale à la deuxième.

char* recherche(char *s, int c) ;

La fonction prend en paramètre une chaîne de caractère s et d'un entier c. Cette fonction permet de rechercher la première occurrence de l'entier c que l'on appel en paramètre. On retourne un pointeur sur le caractère qui correspond à notre entier. La fonction retourne la chaîne de caractère à partir du pointeur correspondant.

int convert(char *s) ;

Prend en paramètre une chaîne de caractère. Celle-ci est parcourue pour trouver des caractères qui correspondent à des chiffres, pour ensuite, les convertir en type entier. La conversion marchera que si les caractères sont situés en début de chaîne, on retourne donc l'entier. Si les bons caractères ne se trouvent pas en début de chaîne, la fonction renvoie 0.

Fichier rectangle.c

Ce programme affiche un rectangle bleu dans une fenêtre de longueur 500 et de largeur 500. On ferme la fenêtre une fois que l'utilisateur clic dans la fenêtre.

Fichier rectangleClic.c

Ce programme affiche un rectangle rouge dans une fenêtre de longueur 700 et de largeur 700. Si l'utilisateur clic dans le rectangle alors il change de couleur, en passant de rouge à bleu et inversement. Si l'utilisateur clic en dehors du rectangle alors la fenêtre est fermée.

void move_Rectangle(int dest_x, int dest_y, int *x_rec, int *y_rec) ;

Fonction qui prend en argument les positions du rectangle ainsi que sa destination voulue. Permet de déplacer de rectangle du nord-est au sud-ouest de manière rebondi jusqu'à que l'utilisateur appuie sur la touche entrée (qui se trouve près des touches numériques) pour mettre fin au programme.

Fichier LectureArg.c

Programme qui compte le nombre d'argument que l'on entre dans le terminale et les affiche.

Fichier Calc.c

Programme qui prends les arguments et les opérateurs puis effectue le calcul correspondant.

int bon_modulo(int a, int b) ;

Cette fonction prend en paramètre deux entier a et b et elle renvoie 0 si l'un des deux entiers est négatif ou elle renvoie le résultat du calcul qui effectue le modulo de deux nombre a et b s'ils sont positifs.

char* crypt(int cle, char *mot) ;

Cette fonction prend en paramètre la clé de cryptage et le mot que l'on veut modifier. Elle renvoie le nouveu mot crypter. On crypt le mot que l'on donne en paramètre en fonction de la clé de cryptage donnée.

char* decrypt(int cle, char *mot) ;

Cette fonction est pratiquement la même que la fonction crypt(), mais elle fait l'inverse de celle-ci. Elle prend les même paramètre que cette fonction et renvoie aussi un le nouveau mot. On des-incrémente par la clé de cryptage, pour retrouver le mot de départ.

Exécution du programme

Pour compiler le programme il suffit de lancer la commande make suivie de l'exécution à l'aide de « ./NOM_PROGRAMME » avec arguments si besoin.

Conclusion

Grâce à ce TP, nous avons progressé sur l'utilisation du man ,mais encore, nous nous sommes familiarisé avec la bibliothèque graphique MLV.

La difficulté a été rencontré lors de la fonction move_Rectangle, quand on devait assigner le statut « relâché » du bouton ENTRÉE, on devait récupérer le type MLV du bouton pour pouvoir faire notre condition. Le type pour le bouton n'était simple à trouver car on a du parcourir plusieurs fichier .h dans la librairie MLV.