

RAPPORT TP 2

I) Introduction

Ce TP 2 a pour but de nous familiariser avec l'utilisation de la documentation **man** de plus il va nous permettre de continuer notre apprentissage de la librairie **NCURSES**.

II) Réponses aux questions

Exercice 1 :

1. strcmp

(**strcmp**) La fonction **strcmp** sert à comparer deux chaînes de caractères en utilisant les valeurs numériques des codes ASCII, pour savoir si la première chaîne de caractère est inférieure, supérieure ou égale à la deuxième.

source : la commande "man strcmp"

2. Programme strcmp

Nous avons fait un petit programme qui demande deux chaînes de caractères et nous dit si elles sont identiques ou différentes en les comparant à l'aide de **strcmp**

3. strchr / strrchr

(**strchr**) La fonction **strchr** sert à rechercher et renvoyer la position de la première occurrence du caractère passé en second paramètre, dans la chaîne de caractère que l'on a entrée en premier paramètre.

source : la commande "man strchr"

(strchr) La fonction **strchr** elle sert à rechercher et renvoyer la position de la dernière occurrence du caractère passé en second paramètre, dans la chaîne de caractère que l'on a entré en premier paramètre.

source : la commande "man strchr"

4. Programme strchr / strchr

Nous avons fait un petit programme pour **strchr** qui change les "e" d'une chaîne de caractère en "z", et pour **strchr** un petit programme qui change la dernière occurrence de "e" de la chaîne de caractère en "z".

5. atoi

(atoi) La fonction **atoi** signifie "ASCII to integer", elle permet donc de convertir une chaîne de caractère de int en un entier

source : La commande "man atoi"

6. Programme atoi

Nous avons fait un petit programme qui demande une chaîne de caractère de int et renvoie cette chaîne de caractère sous forme d'un entier.

Exercice 2

1. Rectangle.c

Créer d'un rectangle rouge centré d'une dimension de 15 sur 3, grâce à une double boucle for.



2. CarreClic.c

Pour la création du carré on procède de la même façon que pour le rectangle ci-dessus.

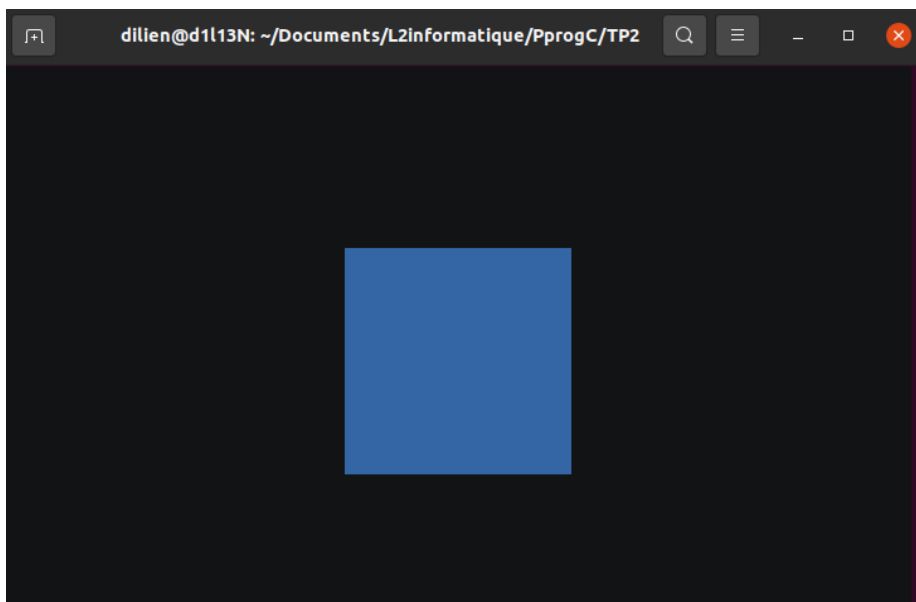
Il faut ensuite faire une détection de clic grâce au MEVENT qui permet de savoir quand et à quelle position l'utilisateur a cliqué sur la fenêtre.

Une fois détecté il suffit de changer les couleurs grâce à un attron

Résultat :

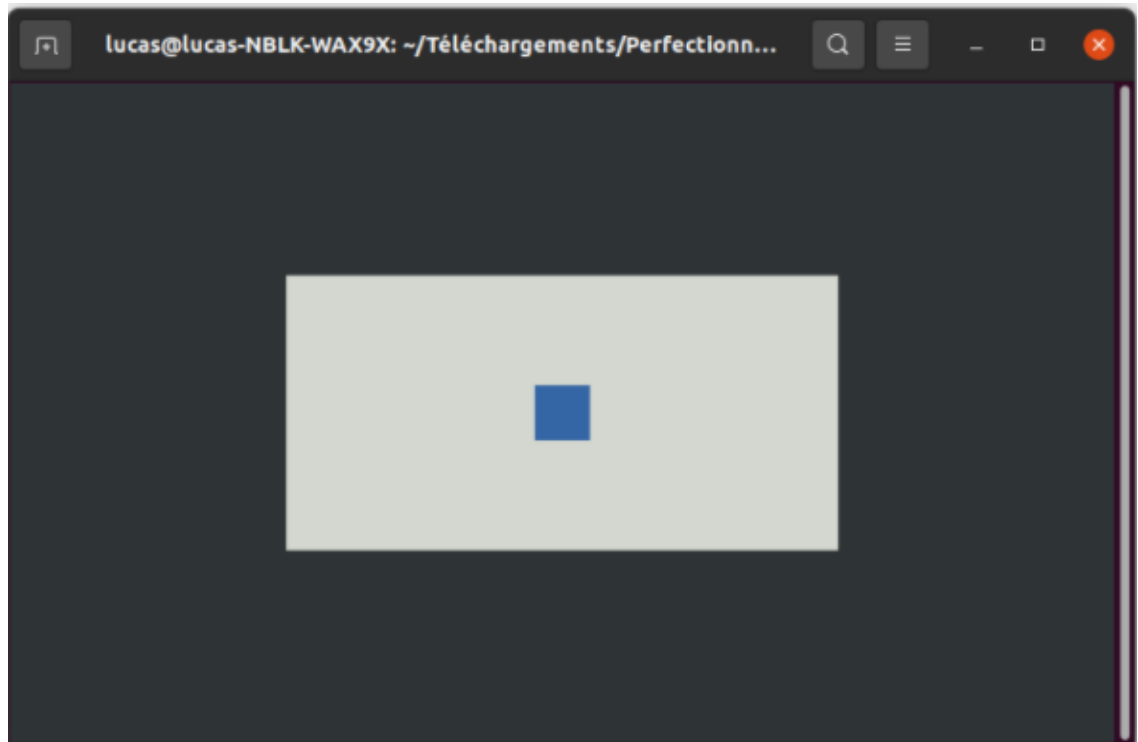


Après un clic sur le carré rouge :



3. Rebond.c

Carré bleu au centre d'un rectangle blanc qui rebondit sur les bordures :

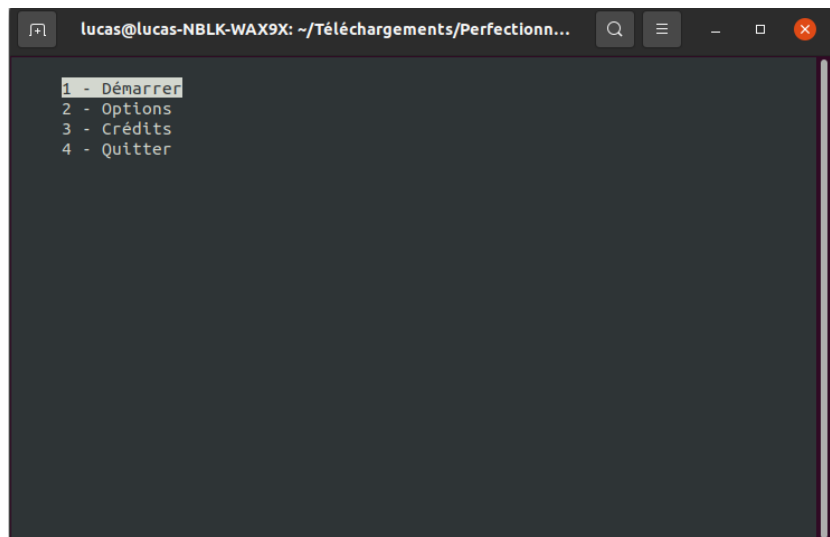


Le carré commencera sa route une fois la touche ENTRER actionnée. Pour arrêter le processus il suffit de réappuyer sur la touche ENTRER.

Nous avons déjà créé vérifier si le carré ne sortait pas du rectangle, s'il venait à sortir, il nous suffisait de multiplier par -1 ses mouvements.

Pour les mouvements rien de plus simple on fait une boucle while avec un usleep, tout en incrémentant les coordonnées du carré puis en rafraichissant la page.

4. Menu.c



```
lucas@lucas-NBLK-WAX9X: ~/Téléchargements/Perfectionn...  
1 - Démarrer  
2 - Options  
3 - Crédits  
4 - Quitter
```

Pour bouger dans le menu l'utilisation des touches directionnelles est possible ou alors vous pouvez tout aussi bien utiliser la partie numérique du clavier pour se déplacer.

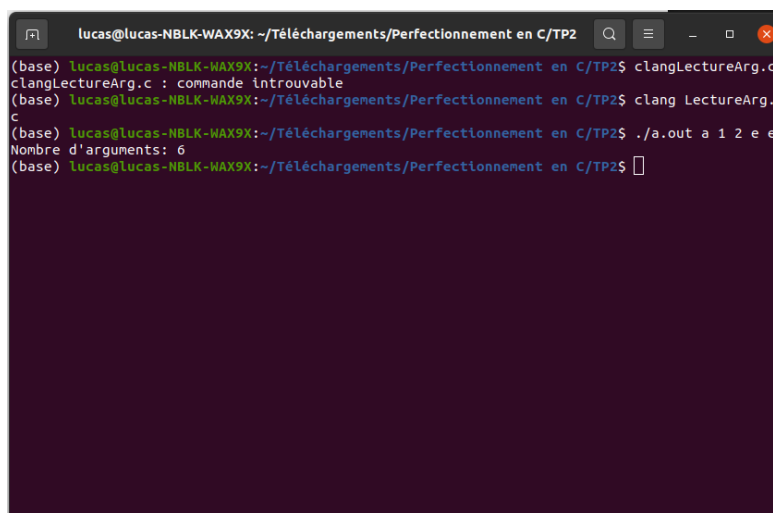
Pour faire fonctionner les flèches directionnelles, il nous a fallu récupérer sur quelle touche l'utilisateur a cliqué.

Grâce à cette touche le programme sait s'il faut

Exercice 3

1. LectureArg

Fonction qui compte le nombre d'arguments entrés :



```
lucas@lucas-NBLK-WAX9X: ~/Téléchargements/Perfectionnement en C/TP2  
(base) lucas@lucas-NBLK-WAX9X:~/Téléchargements/Perfectionnement en C/TP2$ clangLectureArg.c  
clangLectureArg.c : commande introuvable  
(base) lucas@lucas-NBLK-WAX9X:~/Téléchargements/Perfectionnement en C/TP2$ clang LectureArg.  
c  
(base) lucas@lucas-NBLK-WAX9X:~/Téléchargements/Perfectionnement en C/TP2$ ./a.out a 1 2 e e  
Nombre d'arguments: 6  
(base) lucas@lucas-NBLK-WAX9X:~/Téléchargements/Perfectionnement en C/TP2$
```

2. Calc.c

Fonction qui exécute un calcul à partir d'arguments :

```
dilien@d1113N:~/Documents/L2informatique/PprogC/TP2$ ./nccomp EX03/Calc
dilien@d1113N:~/Documents/L2informatique/PprogC/TP2$ ./EX03/Calc
Vous n'avez pas le bon nombre d'arguments.
dilien@d1113N:~/Documents/L2informatique/PprogC/TP2$ ./EX03/Calc + 10 10
Addition : 10 + 10 = 20
dilien@d1113N:~/Documents/L2informatique/PprogC/TP2$ ./EX03/Calc + 0 10
Addition : 0 + 10 = 10
dilien@d1113N:~/Documents/L2informatique/PprogC/TP2$ ./EX03/Calc + zdaom 10
21ème siècle et les calculs de str ne fonctionnent toujours pas !
dilien@d1113N:~/Documents/L2informatique/PprogC/TP2$ ./EX03/Calc * 0 10
Vous n'avez pas le bon nombre d'arguments.
dilien@d1113N:~/Documents/L2informatique/PprogC/TP2$ ./EX03/Calc x 0 10
Multiplication : 0 x 10 = 0
dilien@d1113N:~/Documents/L2informatique/PprogC/TP2$ ./EX03/Calc / 0 10
Division : 0 / 10 = 0
dilien@d1113N:~/Documents/L2informatique/PprogC/TP2$
```

Le programme Calc.c exécute une opération selon le premier argument entré par l'utilisateur. les opérations possibles :

- multiplication (x)
- division (/)
- addition (+)
- soustraction (-)

Ce programme prend en compte quand un utilisateur ne rentre pas un entier.

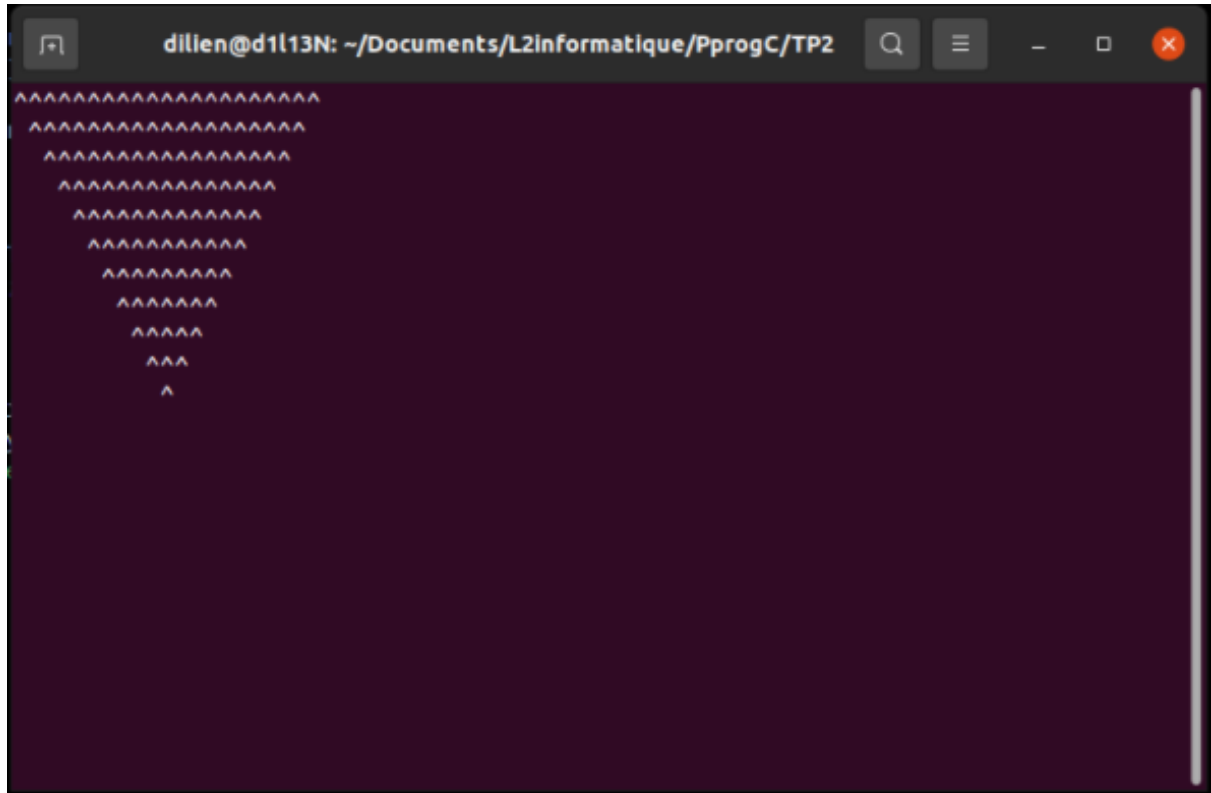
Nous avons utilisé la fonction **strtol** qui nous a permis de différencier les entiers des caractères.

Enfin, nous avons privilégié l'utilisation de **exit** après un **fprintf** dans le fichier de message d'erreur.

3. Triangle.c

Fonction qui trace un triangle inversé d'un hauteur n , et un symbole x données par l'utilisateur.

Exemple avec une taille **10** et un symbole **^**



```
dillien@d1l13N: ~/Documents/L2Informatique/PprogC/TP2
^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^
  ^^^^^^^^^^^^^^^^^
    ^^^^^^^^^^^^^^
      ^^^^^^^^^^^
        ^^^^^^^^^
          ^^^^^^
            ^^^^^
              ^^^
                ^
```

Pour tracer ce triangle, rien de plus simple. Il nous a suffit d'initialiser sa base de taille $2 * \text{hauteur} + 1$.

Puis dans une boucle for nous avons réduit cette hauteur de 2 à chaque passage ainsi que la position d'écriture qui est incrémenté de 1 en x et y.

Exercice 4

1. Expression

Lorsque l'on fait l'expression " $-9 \% 5$ " cela nous donne une valeur négative et cela est gênant car au lieu de nous donner quelque chose de positif, ce qui fait buger le cryptage du message ou encore son décryptage.

3. Crypt.c

Fonction qui crypte une chaîne de caractère avec une clef qui sont toutes les deux données par l'utilisateur.

Pour cet exercice nous avons incrémenté chacune des lettres (ascii) de **clef**. Le soucis avec cette technique est si nous dépassons z. Car dans notre on veut que la lettre a suive la lettre z.

Pour cela il nous a suffit de faire la même formule mais avec cette fois-ci une soustraction de 25, pour retirer ce surplus.

```
dillen@dill13N:~/Documents/L2Informatique/PprogC/TP2$ ./EX04/Crypt 3 Les TP de pr
ogrammmation sont super !
Mot non crypté : Les TP de programmation sont super !
Mot crypté : Ohv WS gh surjudppdwlrq vrqw vxshu !
```

4. Decrypt..c

Fonction qui décrypte une chaîne de caractère avec une clef qui sont toutes les deux données par l'utilisateur.


```
dilien@d1l13N:~/Documents/L2informatique/PprogC/TP2$ ./EX04/Decrypt 3 Ohv WS gh  
surjudppdwlrq vrqw vxshu !  
Mot Crypté : Ohv WS gh surjudppdwlrq vrqw vxshu !  
Mot décrypté : Les TP de programmation sont super !
```

IV) Conclusion

1) Difficultés rencontrées

Dans ce **TP 2** nous n'avons pas rencontré de difficultés particulières.

2) Ce que le sujet nous a apporté

Ce **TP 2** nous a servi à apprendre à utiliser la documentation **man** qui est utile lorsque l'on utilise des nouvelles fonctions, de plus il a servi à approfondir nos connaissances sur la librairie **NCURSES**.

ANNEXE 1

Lors de la première exécution du **programme** il faut entrer la **commande** :

chmod u+x nccomp

Cette **commande** permet que le **nccomp** soit reconnu.

Pour compiler le **projet** il faut exécuter la **commande** :

*./nccomp EXO*X*/nom_du_fichier*

Où le **X** est le numéro de l'exercice ici, 1, 2, 3 ou 4 et *nom_du_fichier* est le nom du programme.c que l'on souhaite exécuter (écrire sans le .c)

ANNEXE 2

Pour utiliser le **projet** il suffit d'exécuter la **commande** suivante :

*./EXO*X*/nom_du_fichier*

Où le **X** est le numéro de l'exercice, ici 1, 2, 3 ou 4 et *nom_du_fichier* est le nom du programme.c que l'on souhaite exécuter (écrire sans le .c)