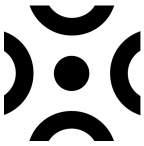
Perfectionnement à la programmation en C

Fiche de TP 2

L2 Informatique - 2024 – 2025

Introduction à `ncurses`



Dans ce TP, nous allons introduire la librairie **`ncurses`** qui permet de développer des applications complètes. La très grande majorité des TP à venir seront des mini projets dans lesquels une application complète sera demandée. Cette dernière devra être programmée de sorte à proposer une interface **`ncurses`** soignée, minimaliste et fonctionnelle.

Il sera judicieux de noter au fur et à mesure des exercices les fonctions découvertes afin de réaliser un aide-mémoire. (voir exercice 10).

1 Apprentissage des notions de base

La librairie **`ncruses`** offre une gestion radicalement performante de la sortie standard. Il devient possible, par exemple, d'écrire à un endroit arbitraire d'une fenêtre. Elle permet d'écrire véritablement des applications robustes et de qualité professionnelle. Voici quelques applications connues basées sur cette librairie :

- `vim` (https://www.vim.org, https://github.com/vim/vim), l'un des meilleurs éditeurs de textes pour programmeurs ;
- `ranger` (https://github.com/ranger/ranger), un explorateur de fichiers très efficace écrit en **`python`** ;
- `htop` (http://hisham.hm/htop, https://github.com/hishamhm/htop), un moniteur et gestionnaire de processus interactif ;
- `lynx` (http://lynx.browser.org), un navigateur internet en mode texte ;
- `moc` (http://moc.daper.net), un lecteur de musique complet et minimaliste.

Pour utiliser **`ncurses`**, il nécessaire d'inclure l'en-tête `ncurses.h`. Un programme `Prog.c` minimal est

```c
#include <ncurses.h>

int main(void) {
    /* Initialise la fenetre. */
    initscr();

    /* Imprime une chaine de caracteres. */
    printw("Bonjour.");

    /* Rafraichit la fenetre. */
    refresh();

    /* Attend l'appui d'une touche. */
    getch();

    /* Ferme la fenetre. */
    endwin();

    return 0;
}
```

Il se compile par la commande

```
gcc -std=c17 -Wall -o Prog Prog.c -lncurses
```

L'option `-lncurses` permet de lier la librairie **ncurses** afin de créer l'exécutable. Le rôle de cette option sera élucidé en détail dans les cours à venir.

Pour apprendre l'utilisation basique d'une librairie, l'une des meilleurs choses à faire consiste à étudier des exemples et à se les approprier en les modifiant légèrement. C'est la stratégie adoptée par les exercices suivants.

1.1 Faire bonne impression

Exercice 1 En étudiant l'exemple

```
#include <ncurses.h>

int main(void) {
    initscr();

    printw("1");

    move(2, 10);
    addch('2');
    addch('3');

    move(LINES - 1, COLS - 1);
    addch('4');
    mvaddch(4, 2, '5');
    mvprintw(3, 3, "ABCD");
    printw("**");

    refresh();
    getch();
    endwin();
    return 0;
}
```

1. décrire la géométrie de la fenêtre (position de l'origine et façon dont les positions sont indexées) ;
2. expliquer ce que représentent les entités `LINES` et `COLS` [1] ;
3. décrire le rôle de la fonction `move` ;
4. comparer et expliquer les différences entre les fonctions `addch`,`mvaddch`, `printw` et `mvprintw` ;
5. écrire un programme dans lequel la chaîne de caractères `"4!+2!"` est imprimée en position `(8, 4)`, signifiant que le 1er caractère de la chaîne occupe cette position.

   Proposer trois versions : une première utilisant uniquement les fonctions `move` et `addch`, une autre uniquement la fonction `mvaddch` et une dernière uniquement la fonction `mvprintw`.

Exercice 2 En étudiant l'exemple

```
#include <ncurses.h>

int main(void) {
    initscr();

    attron(A_NORMAL);
    printw("Normal              : ABCabc012\n");

    attron(A_REVERSE);
    printw("Inverse             : ABCabc012\n");
    attroff(A_REVERSE);
```

[1] contrairement à ce que leurs noms laissent penser, ce ne sont pas des constantes mais des variables dont la valeur change quand le terminal change de taille

```
12
13      attron(A_BOLD);
14      printw("Gras                : ABCabc012\n");
15      attroff(A_BOLD);
16
17      attron(A_UNDERLINE);
18      printw("Souligne            : ABCabc012\n");
19      attroff(A_UNDERLINE);
20
21      attron(A_REVERSE | A_UNDERLINE);
22      printw("Inverse et souligne : ABCabc012\n");
23
24      refresh();
25      getch();
26      endwin();
27
28      return 0;
29  }
```

1. décrire le rôle de la fonction `attron` et des arguments qu'elle accepte;
2. commenter[2] la ligne 11 et en déduire le rôle de la fonction `attroff`;
3. écrire un programme dans lequel la chaîne de caractères `"*10*"` est imprimée en position `(0, 0)` en gras et souligné.

Exercice 3 En étudiant l'exemple

```
1   #include <ncurses.h>
2
3   int main(void) {
4       initscr();
5
6       start_color();
7       init_pair(1, COLOR_RED, COLOR_CYAN);
8       init_pair(2, COLOR_YELLOW, COLOR_BLACK);
9
10      curs_set(FALSE);
11
12      attron(COLOR_PAIR(1));
13      mvprintw(2, 3, "Abc123 **  *");
14      attroff(COLOR_PAIR(1));
15
16      attron(COLOR_PAIR(2));
17      mvprintw(2, 16, "2121");
18      attroff(COLOR_PAIR(2));
19
20      refresh();
21      getch();
22      endwin();
23      return 0;
24  }
```

[2] "commenter" et non "documenter", on souhaite neutraliser la ligne en la mettant en commentaire

1. commenter la ligne 10 et en déduire le rôle de l'appel à fonction `curs_set` avec l'argument `FALSE`;
2. modifier le programme de sorte à écrire en vert sur fond bleu la deuxième chaîne de caractères.

Exercice 4 (Point étape : le damier)

Écrire un programme qui affiche un damier de dimensions 10×10 dont le coin inférieur gauche est au centre de la fenêtre. Les cases alternent entre la couleur rouge et verte.

1.2 Entrez s'il vous plaît !

Exercice 5 En étudiant l'exemple

```c
#include <ncurses.h>

int main(void) {
    char chaine[128];
    int entier;

    initscr();

    getstr(chaine);
    mvprintw(3, 0, "Chaine lue : %s", chaine);
    mvscanw(10, 0, "%d", &entier);

    mvprintw(11, 0, "Entier lu : %d", entier);

    refresh();
    getch();
    endwin();
    return 0;
}
```

1. décrire le rôle des fonctions `getstr` et `mvscanw` ;
2. écrire un programme dans lequel une chaîne de caractères est lue en position `(2, 4)` puis affichée en position `(0, 0)` *(Indice : penser à utiliser la fonction* `move` *avant l'appel à* `getstr` *ou utiliser la fonction* `mvgetstr`*.)* ;
3. écrire un programme dans lequel un entier est tout d'abord demandé en position `(0, 0)`. Si celui-ci est nul, l'exécution se termine. Sinon, un nouvel entier est demandé en position `(1, 1)`, puis `(2, 2)`, puis `(3, 3)`, *etc*, jusqu'à ce que l'utilisateur rentre la valeur `0`. *(Indice : utiliser une boucle* `do while`*).*

Exercice 6 En étudiant l'exemple

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

int main(void) {
    int touche, val, delai;

    initscr();
    noecho();
    nodelay(stdscr, TRUE);

    val = 0;
    delai = 1000000;
    mvprintw(0, 0, "Valeur : %3d", val);
```

```
14      while (1) {
15          touche = getch();
16          if (touche != ERR) {
17              if (touche == 'r')
18                  val = 0;
19              if (touche == 'b')
20                  delai /= 2;
21              if (touche == 't')
22                  delai *= 2;
23          }
24          mvprintw(0, 0, "Valeur : %3d", val);
25          refresh();
26
27          val = (val + 1) % 128;
28          usleep(delai);
29      }
30
31      endwin();
32      return 0;
33  }
```

1. Commenter la ligne 8 et en déduire le rôle de l'appel à la fonction `noecho`;
2. Commenter la ligne 9 et en déduire le rôle de l'appel à la fonction `nodelay` avec les arguments `stdscr` (écran standard) et `TRUE`. *(Indice : ceci modifie le mode de comportement de la fonction `getch`.)* ;
3. décrire précisément le rôle de la fonction `usleep` et en particulier l'unité dans laquelle est exprimé son paramètre. Cette fonction est apportée par le fichier d'en-tête `unistd.h`.

   Ne pas tenir compte d'éventuels messages d'avertissement lors de la compilation concernant l'utilisation de cette fonction.
4. modifier le programme afin que la valeur entière affichée soit en gras (et uniquement celle-ci : la chaine de caractères `"Valeur : "` doit rester en écriture normale) :
5. augmenter le programme d'une fonctionnalité permettant de quitter l'exécution par un appui sur la touche `'q'`. *On tachera d'offrir une telle fonctionnalité pour quitter proprement nos programmes plutôt qu'avec un `Ctrl c`* ;
6. par défaut, certaines touches du clavier ne sont pas accessibles. Pour rendre utilisables entre autres les flèches directionnelles, il faut incorporer l'appel `keypad(stdscr, TRUE)` au début du programme, juste après l'appel à `delay`. De cette façon, remplacer l'utilisation des touches `'b'` et `'t'` respectivement par `KEY_UP` et `KEY_DOWN` ;

Exercice 7 (La souris attrape le chat)

Dans l'exemple

```
1  #include <ncurses.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  void dessiner_chat(int y, int x) {
6      mvprintw(y + 0, x, "^   ^");
7      mvprintw(y + 1, x, "*****");
```

```c
    mvprintw(y + 2, x, "* * *");
    mvprintw(y + 3, x, " *** ");
}

int main(void) {
    srand(time(NULL));

    initscr();
    cbreak();
    noecho();

    keypad(stdscr, TRUE);
    curs_set(FALSE);
    mousemask(ALL_MOUSE_EVENTS | REPORT_MOUSE_POSITION, NULL);

    int chat_x = rand() % (COLS - 4);
    int chat_y = rand() % (LINES - 3);
    dessiner_chat(chat_y, chat_x);

    while (1) {
        MEVENT ev;
        int touche = getch();
        if (touche == 'q')
            break;
        if (touche == KEY_MOUSE && getmouse(&ev) == OK) {
            int souris_x = ev.x;
            int souris_y = ev.y;
            if ((chat_x <= souris_x) && (souris_x <= chat_x + 4) &&
                (chat_y + 1 <= souris_y) && (souris_y <= chat_y + 3)) {
                erase();
                chat_x = rand() % (COLS - 4);
                chat_y = rand() % (LINES - 3);
            }
        }
        dessiner_chat(chat_y, chat_x);
        refresh();
    }
    endwin();
    return 0;
}
```

l'utilisateur doit cliquer sur le chat avec la souris. Un nouveau chat apparaît ensuite aléatoirement dans la fenêtre.

1. Commenter la ligne 21 et en déduire le rôle de l'appel à la fonction `mousemask` avec les arguments mentionnés.

2. Commenter la ligne 37 et en déduire le rôle de la fonction `erase` ;

3. Modifier le programme afin qu'il affiche, lorsque la souris attrape le chat, le message `"Attrape !"` au centre de la fenêtre. Après exactement 500 ms, le message disparaît et un nouveau chat apparaît ensuite.

4. Modifier le programme de sorte que lorsque la souris touche l'un des deux yeux du chat, celui-ci se change en un `'X'`. Le chat n'est pas considéré comme attrapé dans ce cas. Les deux yeux du chat peuvent être ainsi touchés.

2 Notions dans la pratique

Exercice 8 (Triangle)

Écrire un programme qui demande en position `(0, 0)` un entier positif n et qui affiche ensuite à partir de la position `(1, 0)` un triangle fait de lignes de `'*'` de longueur 1, puis 2, ..., jusqu'à n. Par exemple, si $n = 3$, les lignes affichées sont

```
*
**
***
```

Exercice 9 (Marche aléatoire interactive)

En partant de la base suivante :

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

#define DELAI 50000

int main(void) {
    int x, y;

    initscr();

    x = 0;
    y = 0;
    while(1) {
        erase();
        mvaddch(y, x, 'o');
        refresh();
        usleep(DELAI);
    }
    endwin();
    return 0;
}
```

Écrire un programme dans lequel `'o'` est affiché en rouge au centre de la fenêtre. Toutes les secondes, une case voisine orthogonalement au `'o'` est sélectionnée aléatoirement et le `'o'` se déplace sur cette nouvelle case. L'ancienne case occupée est au passage remplacée par un `'x'` vert. Par exemple, un exécution possible pourrait produire au bout de 28 secondes d'exécution l'affichage

```
 xxx
   x
o xx
xxxxxxxx
   x   x
 xxx   x
 xxxxxxx
```

L'utilisateur peut à tout moment doubler la vitesse d'exécution en appuyant sur la flèche du haut, diviser la vitesse d'exécution de moitié en appuyant sur la flèche du bas ou mettre l'exécution en pause en appuyant sur la touche entrée (la fonction `getch` renvoie le caractère `'\n'` en cas d'appui sur cette

touche). Un nouvel appui sur la touche entrée relance l'exécution. La pression de la touche `q` devra terminer le programme.

Enfin, l'utilisateur peut interférer avec la marche aléatoire en téléportant d'un clic le `o`.

---

3 Préparations pour la suite

Exercice 10 (Récapitulons !)

Reprendre toutes les nouvelles fonctions vues dans les exercices précédents en faisant une table dans laquelle apparaît leur prototype, rôle des paramètres, valeur renvoyée et une documentation succincte. L'objectif est de se constituer un aide-mémoire personnalisé pour gagner du temps pour les TP suivants.

Exercice 11 (Ouvertures)

1. Ouvrir la page du manuel de **`ncurses`** (commande `man ncurses`) et la parcourir. Une version interactive se trouve à

   https://www.mkssoftware.com/docs/man3/ncurses.3.asp

   Lire les premières parties pour avoir un apperçu officiel de la librairie.

2. Se rendre sur les pages

   https://invisible-island.net/ncurses/ncurses-intro.html

   http://www.tldp.org/HOWTO/NCURSES-Programming-HOWTO/

   Celles-ci contiennent de nombreux exemples et informations qui couvrent presque l'intégralité des besoins pour les TP suivants mais aussi au delà.
