ncurses

Prise en main de ncurses

Cet article présente comment activer le mode `ncurses` puis comment afficher des informations et récupérer une touche pressée.

1 Les fonctions de base de ncurses

Le fichier `ncurses.c` contient plusieurs fonctions nécessaires pour `ncurses` : l'activation et l'arrêt du mode `ncurses`, l'activation et la configuration des couleurs.

La fonction `ncurses_initialiser` doit être appelée pour activer le mode `ncurses`. Elle contient plusieurs fonctions qui peuvent être modifiées suivant les usages.

```
void ncurses_initialiser() {
  initscr();             /* Démarre le mode ncurses */
  cbreak();              /* Désac. mise en buffer caractères saisis */
  noecho();              /* Désactive affichage des caractères saisis */
  keypad(stdscr, TRUE);  /* Active les touches spécifiques */
  refresh();             /* Met a jour l'affichage */
  curs_set(FALSE);       /* Masque le curseur */
}
```

Avant de quitter un programme `ncurses`, il faut arrêter le mode `ncurses`.

```
void ncurses_stopper() {
  endwin();
}
```

La dernière fonction, `ncurses_couleurs` vérifie si le terminal supporte les couleurs (à l'aide de la fonction `has_colors`) et les active si possible (fonction `start_color`).

```
  if(has_colors() == FALSE) {
    ncurses_stopper();
    fprintf(stderr, "Le terminal ne supporte pas les couleurs.\n");
    exit(EXIT_FAILURE);
  }

  start_color();
```

Ensuite, nous devons spécifier les couleurs qui seront ensuite utilisées dans le programme. Ici, 3 couleurs sont définies (bleu sur fond noir, rouge sur fond noir et vert sur fond noir). Elles seront accessibles à l'aide de la macro `COLOR_PAIR(i)` où i vaut 1, 2 ou 3.

```
    init_pair(1, COLOR_BLUE, COLOR_BLACK);
    init_pair(2, COLOR_RED, COLOR_BLACK);
    init_pair(3, COLOR_GREEN, COLOR_BLACK);
```

2 Premier exemple

Dans ce premier programme, nous illustrons l'usage des couleurs dans `ncurses`. Nous devons dans un premier temps inclure la bibliothèque `ncurses.h` (qui doit être installée sur l'ordinateur) et les fonctions décrites dans la section précédente :

```
#include <ncurses.h>
#include "ncurses.h"
```

Nous initialisons `ncurses` avec les couleurs :

```
ncurses_initialiser();
ncurses_couleurs();
```

Nous affichons ensuite une phrase dans chaque couleur définie dans la fonction `ncurses_couleurs`. Pour cela, nous utilisons la fonction `attron` pour activer la couleur, la fonction `printw` qui est l'équivalent de `printf` en `ncurses` et `attroff` pour désactiver la couleur.

```
or(i = 1; i <= 3; i++) {
  attron(COLOR_PAIR(i));
  printw("Bonjour␣(dans␣la␣couleur␣%d).\n", i);
  attroff(COLOR_PAIR(i));
}
```

Si `ncurses` est stoppé avec la fonction `ncurses_stopper`, l'affichage disparaît. Nous pouvons mettre le programme en pause avec `getch` qui attend que l'utilisateur saisisse une touche :

```
printw("Pressez␣une␣touche...");
getch();
```

3 Gestion du clavier

Ce programme illustre comment utiliser `ncurses` pour gérer la saisie clavier. Un symbole (un losange) est placé au milieu de l'écran et l'utilisateur peut le déplacer en cliquant sur les touches fléchées du clavier.

Nous commençons par déterminer la position de départ du symbole à l'aide des dimensions du terminal récupérées à l'aide des constantes COLS et LINES (accessibles une fois le mode `ncurses` activé). Puis nous affichons le symbole à cette position avec `mvaddch`. Nous devons ensuite rafraîchir l'affichage avec `refresh`.

```
int posX, posY;
posX = COLS / 2 -1;
posY = LINES / 2 - 1;
mvaddch(posY, posX, ACS_DIAMOND);
refresh();
```

La boucle principale correspond à la lecture d'une touche (fonction `getch`) et au déplacement du symbole. Nous choisissons d'arrêter le programme lorsque l'utilisateur presse la touche F2 (la valeur est récupérée à l'aide de la constante `KEY_F(2)`).

```
int ch;
while((ch = getch()) != KEY_F(2)) {
  ...
}
```

Pour "déplacer" le symbole, nous devons l'effacer en affichant un caractère espace à sa position actuelle. Puis nous calculons la nouvelle position (en gérant les débordements) et nous l'affichons.

```
mvaddch(posY, posX, ' ');

switch(ch) {
  case KEY_LEFT:
          if(posX > 0) posX--;
        break;
  case KEY_RIGHT:
          if(posX < COLS - 1) posX++;
        break;
  case KEY_UP:
          if(posY > 0) posY--;
        break;
  case KEY_DOWN:
          if(posY < LINES - 1) posY++;
        break;
}

mvaddch(posY, posX, ACS_DIAMOND);
refresh();
```

4 Compilation et exécution

Le `makefile` fourni permet de compiler les deux programmes précédents. Tapez la commande suivante :

```
make
```

Pour exécuter les programmes, tapez l'une des commandes suivantes :

```
./exemple1
./exemple2
```