TP1

Axel BAILLARGEAU

Introduction

Le but de ce TP est de se familiariser avec les fonctions de la librairie ncurses. On y a donc appris à utiliser les fonctions de base de ncurses, à utiliser les attributs et les couleurs, et à interagir avec l'utilisateur l'aide du clavier et de la souris.

Exercice 1

Question 1

Question 1.1. a

L'origine de la fenêtre est en haut à gauche. Les positions sont indexees à partir de 0. Les coordonnes font la taille d'un charactère.

Question 1.1. b

LINES represente le nombre de lignes de la fenêtre. COLS represente le nombre de colonnes de la fenêtre.

Question 1.1. c

La fonction move() permet de déplacer le curseur logique à la position (y, x) de la fenêtre.

Question 1.1. d

La fonction addch() permet d'ajouter un charactère à la position courante du curseur logique.

La fonction mvaddch() permet d'ajouter un charactère à la position (y, x) de la fenêtre.

La fonction printw() permet d'ajouter une chaine de charactères à la position courante du curseur logique.

La fonction mvprintw() permet d'ajouter une chaine de charactères à la position (y, x) de la fenêtre.

Question 1.1. e

```c
#include <ncurses.h>

int main() {
    initscr();
    move(8, 4);
    for (int i = 0; i < 5; i++) {
        addch("4!+2!"[i]);
    }
```

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

Question 2

Question 1.2. a

La fonction attron() permet d'activer un attribut de la fenêtre, a savoir A_BOLD, A_UNDERLINE, A_REVERSE, A_BLINK, A_DIM, A_STANDOUT, A_PROTECT, A_INVIS, A_ALTCHARSET, A_CHARTEXT.

Question 1.2. b

La fonction attroff() permet de désactiver un attribut de la fenêtre. Elle prend en paramètre un attribut.

Question 1.2. c

```c
#include <ncurses.h>

int main() {
    initscr();

    attron(A_BOLD);
    attron(A_UNDERLINE);

    printw("Inverse␣␣␣␣␣␣␣␣␣␣␣␣:␣ABCabc012\n");

    attroff(A_BOLD);
    attroff(A_UNDERLINE);

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

Question 3

Question 1.3. a

La fonction curs_set() permet de définir la visibilite du curseur. Elle prend en paramètre un entier entre 0 et 2. Ici FALSE prend la valeur 0 donc le curseur est invisible.

Question 1.3. b

```c
#include <ncurses.h>
int main() {
```

```
    initscr();

    start_color();
    init_pair(1, COLOR_RED, COLOR_CYAN);
    init_pair(2, COLOR_GREEN, COLOR_BLUE);

    curs_set(FALSE);

    attron(COLOR_PAIR(1));
    mvprintw(2, 3, "Abc123␣**␣␣*");
    attroff(COLOR_PAIR(1));

    attron(COLOR_PAIR(2));
    mvprintw(2, 16, "2121");
    attroff(COLOR_PAIR(2));

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

Exercice 2

Question 2.1

Question 2.1. a

La fonction clear() permet de remplir la fenêtre avec des espaces.

Question 2.1. b

Placer le clear apres le refresh() dans la boucle est equivalent à placer le clear au debut de la boucle.

Question 2.1. c

Placer le refresh() apres le clear() dans la boucle est equivalent à placer le refresh() a la fin de la boucle.

Question 2.1. d

La fonction usleep permet de mettre en pause le programme pendant un certain nombre de microsecondes.

Question 2.2

Question 2.2. a

On efface toute la fenetre au lieu de ne supprimer que le charactère précédent. Cela pourrait entrainer des clignotements sur les machines les plus lentes.

Question 2.2. b

```c
#include <ncurses.h>
#include <unistd.h>
#define DELAI 20000
int main() {
    int x, y;
    initscr();

    x = 0;
    y = 0;
    while (1) {
        mvaddch(0, 0, '*');
        mvaddch(0, COLS - 1, '*');
        mvaddch(LINES - 1, 0, '*');
        mvaddch(LINES - 1, COLS - 1, '*');

        mvaddch(y, x, 'o');
        refresh();
        mvaddch(y, x, ' ');

        usleep(DELAI);
        x = (x + 1) % COLS;
        y = (y + 1) % LINES;
    }
    endwin();
    return 0;
}
```

Question 2.2. c

```c
#include <ncurses.h>
#include <unistd.h>
#define DELAI 20000
int main() {
    int x = 0, y = 0;
    initscr();

    mvaddch(0, 0, '*');
    mvaddch(0, COLS - 1, '*');
    mvaddch(LINES - 1, 0, '*');
    mvaddch(LINES - 1, COLS - 1, '*');
    while (1) {
        mvaddch(y, x, 'o');
        refresh();
        if (x == 0 || x == COLS - 1) {
            if (y == 0 || y == LINES - 1) {
                mvaddch(0, 0, '*');
                mvaddch(0, COLS - 1, '*');
                mvaddch(LINES - 1, 0, '*');
                mvaddch(LINES - 1, COLS - 1, '*');
            }
```

```
        } else {
            mvaddch(y, x, ' ');
        }

        usleep(DELAI);
        x = (x + 1) % COLS;
        y = (y + 1) % LINES;
    }
    endwin();
    return 0;
}
```

Question 2.2. d

Lorsque le mvaddch() est le mvdelch() ne sont pas separés par un refresh() le char n'est pas affiché.

Exercice 3

Question 3.1

Question 3.1. a

getstr et mvscanw permettent de lire une chaine de charactères depuis le clavier. A la difference de mvgetstr, mvscanw permet de specifier le format de la chaine de charactères.

Question 3.1. b

```
#include <ncurses.h>

int main() {
    char chaine[80];
    initscr();
    mvgetstr(2, 4, chaine);
    mvprintw(0, 0, "%s", chaine);
    refresh();
    getch();
    endwin();
    return 0;
}
```

Question 3.1. c

```
#include <ncurses.h>


int main() {
    int chaine;
    int x = 0, y = 0;
```

```
    initscr();

    do {
        mvscanw(x, y, "%d", &chaine);
        refresh();
        x++;
        y++;
    } while (chaine != 0);

    endwin();
    return 0;
}
```

Question 3.2

Question 3.2. a

La fonction noecho() permet de desactiver l'affichage des charactères saisis au clavier.

Question 3.2. b

```
#include <ncurses.h>
int main() {
    int touche;
    int x, y;
    int x_prec, y_prec;
    initscr();
    noecho();

    x = COLS / 2;
    y = LINES / 2;
    mvaddch(y, x, 'o');

    while (1) {
        x_prec = x;
        y_prec = y;

        touche = getch();
        if (touche == 'q')
            x -= 1;
        if (touche == 'd')
            x += 1;
        if (touche == 'z')
            y -= 1;
        if (touche == 's')
            y += 1;

        if (x < 0)
            x = 0;
        else if (x >= COLS)
            x = COLS - 1;
```

```
        if (y < 0)
            y = 0;
        else if (y >= LINES)
            y = LINES - 1;
        mvaddch(y_prec, x_prec, '_');
        mvaddch(y, x, 'o');
        refresh();
    }

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

Question 3.2. c

```
#include <ncurses.h>
int main() {
    int touche;
    int x, y;
    int x_prec, y_prec;
    initscr();
    noecho();

    x = COLS / 2;
    y = LINES / 2;
    mvaddch(y, x, 'o');

    while (1) {
        x_prec = x;
        y_prec = y;

        touche = getch();
        if (touche == 'q')
            x -= 1;
        if (touche == 'd')
            x += 1;
        if (touche == 'z')
            y -= 1;
        if (touche == 's')
            y += 1;
        if (touche == 'i') {
            x = COLS / 2;
            y = LINES / 2;
            x_prec = x;
            y_prec = y;
            clear();
        }

        if (x < 0)
            x = 0;
        else if (x >= COLS)
```

```
            x = COLS - 1;
        if (y < 0)
            y = 0;
        else if (y >= LINES)
            y = LINES - 1;
        mvaddch(y_prec, x_prec, '_');
        mvaddch(y, x, 'o');
        refresh();
    }

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

Question 3.2. d

```
#include <ncurses.h>
int main() {
    int touche;
    int x, y;
    int x_prec, y_prec;
    int pas = 1;
    initscr();
    noecho();

    x = COLS / 2;
    y = LINES / 2;
    mvaddch(y, x, 'o');
    mvprintw(LINES - 1, 0, "Pas : %d", pas);

    while (1) {
        x_prec = x;
        y_prec = y;

        touche = getch();
        switch (touche) {
            case 'p':
                pas++;
                break;
            case 'm':
                if (pas > 1)
                    pas--;
                break;
            case 'q':
                x -= pas;
                break;
            case 'd':
                x += pas;
                break;
            case 'z':
                y -= pas;
```

```
                break;
            case 's':
                y += pas;
                break;
            case 'i':
                x = COLS / 2;
                y = LINES / 2;
                x_prec = x;
                y_prec = y;
                clear();
                break;
        }

        if (x < 0)
            x = 0;
        else if (x >= COLS)
            x = COLS - 1;
        if (y < 0)
            y = 0;
        else if (y >= LINES)
            y = LINES - 1;
        mvaddch(y_prec, x_prec, '_');
        mvaddch(y, x, 'o');
        move(LINES - 1, 0);
        for (int i = 0; i < 6 + (int)(pas / 10) + 2; i++) {
            delch();
        }
        mvprintw(LINES - 1, 0, "Pas : %d", pas);
        refresh();
    }

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

Question 3.3

Question 3.3. a

La fonction nodelay() permet de rendre le getch() non bloquant. L'argument stdscr est le pointeur sur la fenêtre courante.

Question 3.3. b

```
#include <ncurses.h>
#include <unistd.h>
int main() {
    int touche, val, delai;
    initscr();
    noecho();
```

```
    nodelay(stdscr, TRUE);

    val = 0;
    delai = 1000000;
    mvprintw(0, 0, "Valeur : ");
    attron(A_BOLD);
    printw("%3d", val);
    attroff(A_BOLD);
    while (1) {
        touche = getch();
        if (touche != ERR) {
            if (touche == 'r')
                val = 0;
            if (touche == 'b')
                delai /= 2;
            if (touche == 't')
                delai *= 2;
        }
        mvprintw(0, 0, "Valeur : ");
        attron(A_BOLD);
        printw("%3d", val);
        attroff(A_BOLD);
        refresh();

        val = (val + 1) % 128;
        usleep(delai);
    }

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

Question 3.3. c

```
#include <ncurses.h>
#include <unistd.h>
int main() {
    int touche, val, delai;
    initscr();
    noecho();
    nodelay(stdscr, TRUE);

    val = 0;
    delai = 1000000;
    mvprintw(0, 0, "Valeur : ");
    attron(A_BOLD);
    printw("%3d", val);
    attroff(A_BOLD);
    while (1) {
        touche = getch();
        if (touche != ERR) {
```

```
            if (touche == 'r')
                val = 0;
            if (touche == 'b')
                delai /= 2;
            if (touche == 't')
                delai *= 2;
            if (touche == 'q')
                break;
        }
        mvprintw(0, 0, "Valeur : ");
        attron(A_BOLD);
        printw("%3d", val);
        attroff(A_BOLD);
        refresh();

        val = (val + 1) % 128;
        usleep(delai);
    }

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

Question 3.3. d

```
#include <ncurses.h>
#include <unistd.h>
int main() {
    int touche;
    int x, y;
    int x_prec, y_prec;
    initscr();
    noecho();

    x = COLS / 2;
    y = LINES / 2;
    mvaddch(y, x, 'o');

    while (1) {
        x_prec = x;
        y_prec = y;

        touche = getch();
        if (touche == 'q')
            x -= 1;
        if (touche == 'd')
            x += 1;
        if (touche == 'z')
            y -= 1;
        if (touche == 's')
            y += 1;
```

```
        if (touche == 'i') {
            x = COLS / 2;
            y = LINES / 2;
            x_prec = x;
            y_prec = y;
            clear();
        }

        if (x < 0)
            x = 0;
        else if (x >= COLS)
            x = COLS - 1;
        if (y < 0)
            y = 0;
        else if (y >= LINES)
            y = LINES - 1;
        mvaddch(y_prec, x_prec, '_');
        mvaddch(y, x, 'o');
        refresh();
    }

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

Question 3.3. e

```
#include <ncurses.h>
#include <unistd.h>
int main() {
    int touche, val, delai;
    initscr();
    cbreak();
    noecho();
    nodelay(stdscr, TRUE);
    keypad(stdscr, TRUE);

    val = 0;
    delai = 1000000;
    mvprintw(0, 0, "Valeur : ");
    attron(A_BOLD);
    printw("%3d", val);
    attroff(A_BOLD);
    while (1) {
        touche = getch();
        if (touche != ERR) {
            if (touche == 'r')
                val = 0;
            if (touche == KEY_UP)
                delai /= 2;
            if (touche == KEY_DOWN)
```

```
                delai *= 2;
            if (touche == 'q')
                break;
        }
        mvprintw(0, 0, "Valeur : ");
        attron(A_BOLD);
        printw("%3d", val);
        attroff(A_BOLD);
        refresh();

        val = (val + 1) % 128;
        usleep(delai);
    }

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

Exercice 4

Question 4.1

La fonction mousemask() permet de définir les événements souris à surveiller. ALL_MOUSE_EVENTS et REPORT_MOUSE_POSITION sont des constantes définies dans le fichier ncurses.h.

Question 4.2

```
int main() {
    int touche;
    int chat_x, chat_y;
    int souris_x, souris_y;
    MEVENT ev;

    srand(time(NULL));

    initscr();
    cbreak();
    noecho();
    keypad(stdscr, TRUE);
    nodelay(stdscr, TRUE);
    curs_set(FALSE);
    mousemask(ALL_MOUSE_EVENTS | REPORT_MOUSE_POSITION, NULL);

    chat_x = rand() % (COLS - 4);
    chat_y = rand() % (LINES - 2);
    while (1) {
        touche = getch();
        if (touche == KEY_MOUSE && getmouse(&ev) == OK) {
            souris_x = ev.x;
            souris_y = ev.y;
            if ((chat_x <= souris_x) && (souris_x <= chat_x + 4) &&
```

```
    (chat_y <= souris_y) && (souris_y <= chat_y + 2)) {
                    effacer_chat(chat_y, chat_x);
                    chat_x = rand() % (COLS - 4);
                    chat_y = rand() % (LINES - 2);
                    mvprintw(LINES / 2, COLS / 2 - (int)(strlen("Attrape !") /
    2), "Attrape !");
                    refresh();
                    usleep(500000);
                    mvprintw(LINES / 2, COLS / 2 - (int)(strlen("Attrape !") /
    2), "          ");
                }
            }
            dessiner_chat(chat_y, chat_x);
            refresh();
        }

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

Question 4.3

Les mouvements de souris n'etaient pas reconnus par ma machine. Je n'ai pas pu tester le code.

```
    int main() {
        int touche;
        int chat_x, chat_y;
        int souris_x, souris_y;
        MEVENT ev;

        srand(time(NULL));

        initscr();
        cbreak();
        noecho();
        keypad(stdscr, TRUE);
        nodelay(stdscr, TRUE);
        curs_set(FALSE);
        mousemask(ALL_MOUSE_EVENTS | REPORT_MOUSE_POSITION, NULL);

        chat_x = rand() % (COLS - 4);
        chat_y = rand() % (LINES - 2);
        while (1) {
            FILE *f = fopen("./test.txt", "w");
            fprintf(f, "chat_x = %d, chat_y = %d", chat_x, chat_y);
            fclose(f);
            touche = getch();
            int tmp = getmouse(&ev);
            if (touche == KEY_MOUSE && tmp == OK) {
                souris_x = ev.x;
                souris_y = ev.y;
```

```
            if ((chat_x <= souris_x) && (souris_x <= chat_x + 4) &&
(chat_y <= souris_y) && (souris_y <= chat_y + 2)) {
                effacer_chat(chat_y, chat_x);
                chat_x = rand() % (COLS - 4);
                chat_y = rand() % (LINES - 2);
                mvprintw(LINES / 2, COLS / 2 - (int)(strlen("Attrape !") /
2), "Attrape !");
                refresh();
                usleep(500000);
                mvprintw(LINES / 2, COLS / 2 - (int)(strlen("Attrape !") /
2), "          ");
            }
        } else if ((chat_x == souris_x || souris_x == chat_x + 4) &&
(chat_y == souris_y + 1)) {
            fprintf(stderr, "test");
            effacer_chat(chat_y, chat_x);
            dessiner_chat_Croix(chat_y, chat_x);
        } else {
            dessiner_chat(chat_y, chat_x);
        }
        refresh();
    }

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

Exercice 5

```
int main() {
    initscr();
    start_color();
    init_pair(1, COLOR_BLACK, COLOR_RED);
    init_pair(2, COLOR_BLACK, COLOR_GREEN);
    curs_set(FALSE);

    for (int i = 0; i < 10; i++) {
        move(LINES - i - 1, 0);
        for (int j = 0; j < 10; j++) {
            int swi = (j + i) % 2 + 1;
            attron(COLOR_PAIR(swi));
            printw(" ");
            attroff(COLOR_PAIR(swi));
        }
    }
    refresh();
    getch();
    endwin();
    return 0;
}
```

Question 6

```
int main() {
    initscr();
    int n = 0;
    scanw("%d", &n);
    for (int i = 0; i < n + 1; i++) {
        move(i, 0);
        for (int j = 0; j < i; j++) {
            printw("*");
        }
    }
    refresh();
    getch();
    endwin();
    return 0;
}
```

Question 7

```
int main() {
    initscr();
    nodelay(stdscr, TRUE);
    keypad(stdscr, TRUE);
    // cbreak();

    int x = COLS / 2, y = LINES / 2;
    int old_x = x, old_y = y;
    int touche;
    int sleep = 100000;
    int loop = 1;

    while (1) {
        touche = getch();
        move(old_y, old_x);
        printw("x");

        old_x = x;
        old_y = y;
        move(y, x);
        printw("o");
        refresh();

        switch (touche) {
            case KEY_UP:
                sleep /= 2;
                break;

            case KEY_DOWN:
                sleep *= 2;
```

```
                break;

            case '\n':
                loop = 0;
                break;
        }

        while (loop == 0) {
            touche = getch();
            if (touche == '\n') {
                loop = 1;
            }
        }

        x = x + rand() % 3 - 1;
        y = y + rand() % 3 - 1;
        if (x < 0) {
            x = 0;
        } else if (x > COLS - 1) {
            x = COLS - 1;
        }
        if (y < 0) {
            y = 0;
        } else if (y > LINES - 1) {
            y = LINES - 1;
        }
        usleep(sleep);
    }

    endwin();
    return 0;
}
```

Question 8

```
int main() {
    initscr();
    nodelay(stdscr, TRUE);
    keypad(stdscr, TRUE);
    curs_set(FALSE);
    mousemask(ALL_MOUSE_EVENTS | REPORT_MOUSE_POSITION, NULL);
    start_color();
    init_pair(1, COLOR_BLUE, COLOR_BLUE);

    int x = 0, y = 0;
    MEVENT ev;
    int touche;

    for (int i = 0; i < COLS; i++) {
        for (int j = 0; j < LINES; j++) {
            move(j, i);
            printw("0");
```

```
            }
        }

        do {
            touche = getch();
            if (touche == 'p') {
                break;
            }
            int tmp = getmouse(&ev);
            if (touche == KEY_MOUSE && tmp == OK) {
                x = ev.x;
                y = ev.y;
                move(y, x);
                chtype res = winch(stdscr);
                move(y, x);

                if (res == '0') {
                    printw("1");
                } else {
                    attron(COLOR_PAIR(1));
                    printw("2");
                    attroff(COLOR_PAIR(1));
                }

                refresh();
            }
        } while (1);

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

Question 9

Fonction	Description
`int initscr(void)`	Initialise la fenêtre
`int endwin(void)`	Termine la fenêtre
`int refresh(void)`	Actualise la fenêtre
`int getch(void)`	Récupère une touche
`int addch(chtype)`	Ajoute un caractère
`int move(y, x)`	Déplace le curseur
`int printw(char*, ...)`	Affiche une chaîne de caractères
`int scanw(char*, ...)`	Récupère une chaîne de caractères
`int clear(void)`	Efface la fenêtre

Fonction	Description
`int erase(void)`	Efface la ligne courante
`int delch(void)`	Supprime le caractère courant
`int attron(attr)`	Active un attribut
`int attroff(attr)`	Désactive un attribut
`cbreak(void)`	Désactive le buffer
`nocbreak(void)`	Active le buffer
`int nodelay(WINDOW*, bool)`	Active/désactive le mode non bloquant
`int keypad(WINDOW*, bool)`	Active/désactive le clavier étendu
`int mousemask(int, int*)`	Active/désactive la souris
`int getmouse(MEVENT*)`	Récupère les événements de la souris
`int start_color(void)`	Active les couleurs
`int init_pair(short, short, short)`	Initialise une paire de couleurs
`int curs_set(int)`	Active/désactive le curseur
`int usleep(useconds_t)`	Met en pause le programme
`chtype winch(WINDOW*)`	Récupère le caractère courant