

Présentation de ncurses

Cyril Rabat

`cyril.rabat@univ-reims.fr`

Licence 3 Informatique - Info0601 - Systèmes d'exploitation - concepts avancés

2021-2022



Cours n°1

Présentation générale de ncurses

Fonctions de bases et gestion des fenêtres

Version 13 décembre 2021

Table des matières

- 1 Présentation de `ncurses`
 - Introduction
 - Sortie écran
 - Gestion des saisies clavier
 - Gestion de la souris
- 2 Les fenêtres dans `ncurses`
 - Présentation des fenêtres
 - Fenêtres multiples
 - Souris et fenêtres
- 3 Conclusion et références

Qu'est-ce que ncurses ?

- Bibliothèque permettant de créer des interfaces dans la console :
 - Création de menus
 - Gestion de "fenêtres" (pas au sens classique)
 - Manipulation du clavier et de la souris
 - Gestion des couleurs
- Pourquoi ?
 - Proposer une IHM sans environnement graphique
 - ↪ On reste en programmation système
 - Gommer les spécificités des consoles
 - Rester en bas niveau
- Très utilisé notamment dans les installations *Linux* ou de logiciels

Installation, inclusion et compilation

- Bibliothèque installée normalement par défaut (sous *Ubuntu*)...
- ... mais pas les fichiers d'en-tête nécessaires pour la compilation
↪ Fichiers d'en-tête C *.h
- Commande (sous *Ubuntu*) :
↪ `sudo apt-get install libncurses-dev`
- Pour le manuel (le man) :
↪ `sudo apt-get install ncurses-doc`
- Dans le code C : inclure `ncurses.h` (un seul fichier)
↪ `#include <ncurses.h>`
- Pour l'édition de liens, spécifier la bibliothèque :
↪ `gcc -o test -lncurses`

Utilisation de ncurses

- Avant utilisation, activation du «mode» ncurses
⇨ Fonction `initscr`
- Configuration globale de ncurses :
 - Comportement du clavier
 - Activation des touches spéciales
 - Initialisation des couleurs
 - Initialisation de la souris
- Conseil, création de fonctions spécifiques (avec .h et .c) :
⇨ Par exemple :
 - `ncurses_init()`
 - `ncurses_stop()`

Exemple de fonctions d'initialisation et d'arrêt

```
void ncurses_init() {
    initscr();           // Démarre le mode ncurses
    cbreak();            // Désactive la mise en buffer
    noecho();            // Désactive l'affichage des
                        // caractères saisis
    keypad(stdscr, TRUE); // Active les touches spé
                        // cifiques (flèches)
    refresh();           // Met à jour l'affichage
    curs_set(FALSE);     // Masque le curseur
}

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

Dans la suite, ces fonctions seront appelées systématiquement.

Vérification des dimensions du terminal

- Problématiques classiques d'une interface :
 - ↪ Les dimensions du terminal
- Possible de vérifier les dimensions du terminal :
 - Constantes COLS et LINES
 - Fonction `getmaxyx(stdscr, lignes, colonnes)`

Remarques

- `stdscr` correspond à la fenêtre principale :
 - ↪ Gestion des fenêtres dans la suite
- Impossible de modifier les dimensions du terminal depuis le code
 - ↪ Des solutions non portables existent (mais à proscrire)

Afficher un caractère : fonction addch

- Affiche un caractère à la position courante (curseur)
- Utilisation basique : `addch('B');`
- Possibilité d'ajouter des attributs :
 - `A_BOLD`
 - `A_UNDERLINE`
- Exemple : `addch('B' | A_UNDERLINE);`
- Possibilité de placer un caractère à la position souhaitée :
 - `move(ligne, colonne)` : déplacement du curseur
 \hookrightarrow Puis `addch`
 - `mvaddch(ligne, colonne, car)` : affiche le caractère à la position spécifiée

Attention à l'ordre : $(\text{ligne}, \text{colonne}) \rightarrow (y, x)$

Écrire une chaîne de caractères : `printw` et `addstr`

- Équivalent de `printf` :
 \hookrightarrow `printw("Coucou %d %s", 12, "Toto")`
- À une position donnée : `mvprintw(posY, posX, "Toto")`
- `addstr` : affiche une chaîne sans format

Ces fonctions écrivent sur le `stdscr`.

Leurs équivalents `wprintw`, `wmvprintw`... sont utilisés pour les fenêtres.

Gestion des attributs

- Activation / désactivation d'attributs :
↔ `attron` et `attroff`
- Attributs définis dans `ncurses.h` :
↔ Exemples : `A_BOLD`, `A_UNDERLINE`...

Exemple : écriture en gras

```
attron(A_BOLD);  
printw("Mangez_moins_gras");  
attroff(A_BOLD);
```

Tous les attributs ne sont pas supportés par tous les terminaux !

Gestion des couleurs

- Possible d'écrire en couleur avec `ncurses`
- Vérification avant si le terminal accepte :
 ↪ `has_colors()` doit retourner `TRUE`
- Activation puis définition du jeu de couleurs :
 ↪ Définition de la palette utilisée
- Utilisation de paires numérotées :
 - Couleur du texte
 - Couleur du fond
 - Fonction `init_pair`
- Récupération des couleurs : `COLOR_PAIR(num)`
- Activation d'une couleur avec `attron`

Notes à l'intention des artistes

Il est possible d'utiliser des couleurs en spécifiant les composantes RGB
N'oubliez pas que vous êtes dans un terminal!!!

Exemple de fonction d'initialisation des couleurs

```
void ncurses_couleurs() {  
    if(has_colors() == FALSE) {  
        ncurses_stop();  
        fprintf(stderr, "Pas_de_gestion_de_la_couleur_pour_  
            ce_terminal.\n");  
        exit(EXIT_FAILURE);  
    }  
  
    start_color();  
}
```

Exemple d'utilisation des couleurs (1/2)

```
void palette() {  
    init_pair(1, COLOR_BLUE, COLOR_BLACK);  
    init_pair(2, COLOR_RED, COLOR_BLACK);  
    init_pair(3, COLOR_GREEN, COLOR_BLACK);  
}
```

Exemple d'utilisation des couleurs (2/2)

```
int main() {
    int i;

    ncurses_init();
    ncurses_couleurs();
    palette();
    for(i = 1; i <= 3; i++) {
        attron(COLOR_PAIR(i));
        printf("Coucou_%d\n", i);
        attroff(COLOR_PAIR(i));
    }
    getch();
    ncurses_stop();

    return EXIT_SUCCESS;
}
```

Gestion des caractères étendus

- L'affichage des accents peut poser problème avec *ncurses*
- Solution : définir la localisation
- Utilisation de la fonction `setlocale`
- La liaison doit être effectuée avec `lncursesw`

L'appel doit être effectué AVANT l'initialisation de *ncurses*

```
#include <locale.h>
```

```
...  
    setlocale(LC_ALL, "");  
    ncurses_init();  
...
```

Lecture au clavier

- `ncurses` permet de gérer les saisies clavier avancées
- Possibilité de gérer les touches spéciales :
 - ↪ Activation : `keypad(stdscr, TRUE)`
 - ↪ Pas de contrôle sur *Control*, *Shift* et *Alt*
- Par défaut, mise en tampon des touches saisies dans la console :
 - Désactivation avec `raw` ou `cbreak`
 - Pas de signal envoyé avec `raw` lors d'un CTRL+C ou CTRL+Z
- Lecture d'un caractère avec la fonction `getch` :
 - ↪ Retourne le caractère (valeur ASCII) pour les caractères normaux
- Pour les touches spéciales, utilisation de constantes :
 - ↪ `KEY_F(2)`, `KEY_DOWN`, *etc.*

getch retourne un int, pas un char.....mais ce n'est pas la peine de comparer avec les codes ASCII!!!

Exemple : déplacement avec les touches fléchées (1/2)

```
int main() {  
    int i, ch, posX, posY;  
  
    ncurses_init();  
    curs_set(FALSE); // Cache le curseur  
  
    // Calcul de la position centrale  
    posX = COLS / 2 - 1; posY = LINES / 2 - 1;  
    mvaddch(posY, posX, ACS_DIAMOND);  
    ...  
}
```

Exemple : déplacement avec les touches fléchées (2/2)

```
...  
// Boucle principale : arrêt en pressant F2  
while((ch = getch()) != KEY_F(2)) {  
    mvaddch(posY, posX, '_'); // Efface  
    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);  
}  
...
```

Quelques mots sur la saisie

- Pour masquer le curseur (à réactiver en cas de saisie) :
↪ `curs_set (FALSE)`
- Pour empêcher la mise en tampon des saisies :
↪ `cbreak` ou `raw`
- Pour rendre le `getch` non bloquant :
↪ `nodelay`
↪ `timeout` : idem mais en spécifiant un délai en millisecondes

Remarques

- Certains comportements sont fixés par défaut sur certains terminaux
- Certaines commandes sont sans effet : attention à la compatibilité !

Utilisation de la souris

- Pour utiliser la souris, mise en place d'un masque :
 - Boutons gérés
 - Combinaisons acceptées
- Définition d'un ensemble de constantes :
 - ↪ Exemples : `BUTTON1_CLICKED`, `BUTTON1_PRESSED...`
- Gestion des évènements de la souris comme une touche :
 - `getch` retourne `KEY_MOUSE`
 - Récupération ensuite d'un `MEVENT` avec `getmouse`

Structure MEVENT

```
typedef struct {
    short id;           /* Gestion des périphériques multiples
                        */
    int x, y, z;        /* Coordonnées de l'évènement */
    mmask_t bstate;     /* État des boutons */
}
```

Exemple d'utilisation de la souris (1/3)

```
/**
 * Initialisation de la souris.
 */
void ncurses_initsouris() {
    if(!mousemask(BUTTON1_PRESSED, NULL)) {
        ncurses_stop();
        fprintf(stderr, "Pas_de_gestion_de_la_souris.\n");
        exit(EXIT_FAILURE);
    }
}
```

Exemple d'utilisation de la souris (2/3)

```
/**  
 * Récupération de la position de la souris.  
 */  
int souris_getpos(int *x, int *y) {  
    MEVENT event;  
    int resultat = getmouse(&event);  
    if(resultat == OK) {  
        *x = event.x; *y = event.y;  
    }  
    return resultat;  
}
```

Exemple d'utilisation de la souris (3/3)

```
int i, ch, posX, posY;

ncurses_initialisation();
ncurses_initsouris();
while((ch = getch()) != KEY_F(2))
    switch(ch) {
        case KEY_MOUSE:
            if(souris_getpos(&posX, &posY) == OK) {
                move(LINES - 1, 2);
                clrtoeol();
                mvprintw(LINES - 1, 2,
                        "Clic_a_la_position_(%d,_%d)",
                        posX, posY);
            }
    }
```

Les fenêtres

- Permettent de définir des zones indépendantes :
 - Possibilité d'effacer uniquement une zone
 - Possibilité de gérer le défilement automatique du texte dans la zone
 - *etc.*
- Au moment de la création :
 - Définition de la position
 - Définition de la largeur et la hauteur
 - Possibilité de créer un cadre
- Intérêts :
 - Faciliter la gestion de l'interface
 - Améliorer l'efficacité (pas de rafraichissement complet)

Construction, destruction et propriétés d'une fenêtre

- `newwin` : crée une fenêtre
 - ↪ Allocation d'une structure `WINDOW`
- Ne pas oublier de libérer les ressources :
 - ↪ Utilisation de `delwin` AVANT d'arrêter `ncurses`
- La suppression n'implique pas de rafraîchissement de la fenêtre principale :
 - ↪ À faire manuellement
- Accès aux propriétés (taille, position du curseur) :
 - Utilisation de macros (au lieu d'accéder à la structure)
 - Exemples : `getbegyx`, `getmaxyx`...

Exemple d'utilisation d'une fenêtre

```
int i;
WINDOW * fenetre = newwin(10, 10, 5, 20);
for(i = 0; i < 15; i++) {
    wprintw(fenetre, "Bonjour_%d\n", i);
    wrefresh(fenetre);
    getch();
}
```

- Fenêtre de 10 par 10
- Position (5, 20) (premier caractère en haut à gauche)
- Affichage à l'intérieur : `wprintw`
- Ne pas oublier de rafraichir avec `wrefresh`

Écrire dans une fenêtre

- Utilisation des fonctions précédées de "w" :
 - `waddch(fenetre, ...)`, `wprintw`, `mvwprintw`
 - `wmove`
 - `wattron`, `wattroff`
 - `werase`
- Équivalents (à l'affichage) :
 - `wprintw(stdscr, "Bonjour")`
 - `printw("Bonjour")`
- Permettre le défilement dans une fenêtre :
↪ `scrollok(fenetre, TRUE)`

Créer un cadre autour d'une fenêtre

- Fonction `box` :
 - Utilise les caractères par défaut pour les coins
 - Permet de sélectionner les caractères pour les lignes et les colonnes
- Fonction `wborder` :
 - Permet de spécifier tous les caractères
- Pour les deux : caractère 0 = caractère par défaut
- Équivalents :
 - ↪ `box(fenetre, 0, 0)`
 - ↪ `box(fenetre, ACS_VLINE, ACS_HLINE)`
 - ↪ `wborder(fenetre, ACS_VLINE, ACS_VLINE, ACS_HLINE, ACS_HLINE, ACS_ULCORNER, ACS_URCORNER, ACS_LLCORNER, ACS_LRCORNER)`

Fonctionnement

- Possibilité de créer plusieurs fenêtres indépendantes
- Intérêt : rafraichir uniquement une zone
- Problème lors du recouvrement :
 - ⇨ Comment rafraichir les fenêtres ?
 - ⇨ Dans quel ordre ?
- Exemple de création de boites de dialogue :
 - Création de la fenêtre (la boîte)
 - Affichage du contenu
 - Destruction de la fenêtre
 - Rafraichissement des fenêtres du dessous

C'est un exemple, nous n'aurons pas besoin de boîtes de dialogue...

Forcer le rafraichissement d'une fenêtre

- `wrefresh` n'a pas d'effet si aucune modification dans la fenêtre :
↳ Gestion des modifications automatique
- Comment forcer le rafraichissement d'une fenêtre ?
↳ Utilisation de `touchwin` :
↳ L'intégralité de la fenêtre est rafraichie avec `wrefresh`
- Possibilité de spécifier uniquement des lignes données :
↳ `wtouchline`

Exemple de superposition de fenêtres (1/2)

```
// Création de la première fenêtre
WINDOW *fenetre1 = newwin(10, 15, 10, 2);
box(fenetre1, 0, 0);
for(i = 1; i < 9; i++)
    for(j = 1; j < 14; j++)
        mvwaddch(fenetre1, i, j, 'p');
wrefresh(fenetre1);
getch();
...
```

Exemple de superposition de fenêtres (2/2)

```
// Création de la deuxième fenêtre  
WINDOW *fenetre2 = newwin(6, 11, 12, 4);  
box(fenetre2, 0, 0);  
wrefresh(fenetre2);  
getch();  
// La fenêtre apparaît au-dessus de la 1ère  
  
touchwin(fenetre1);  
wrefresh(fenetre1);  
getch();  
// La première fenêtre réapparaît, la 2nd
```


Séparer le cadre d'une fenêtre du contenu

- Problème :
 - Supposons qu'une fenêtre possède un cadre et que le défilement est activé (avec `scrollok`)
 - Le cadre défilera vers le haut avec le texte !
- Une solution :
 - Utilisation de deux fenêtres
 - Cadre dans la première
 - Contenu dans la seconde (avec défilement)
 - Seconde fenêtre inclue dans la première
- Autre solution :
 - Utilisation d'une fenêtre pour le contenu
 - Créer un cadre dans la fenêtre principale
 - ↪ Attention, cadre réalisé manuellement

Les sous-fenêtres

- Permettent de partager la mémoire d'une fenêtre
- Fonctionnent comme les fenêtres :
↔ Allocation d'une structure `WINDOW`
- Attention à l'ordre de destruction :
 - 1 Destruction des sous-fenêtres
 - 2 Destruction de la fenêtre
- Fonctions associées :
 - `subwin` : création d'une sous-fenêtre
 - `derwin` : idem mais à partir des coordonnées de la fenêtre parente
 - `delwin` : destruction d'une fenêtre/sous-fenêtre

Certaines routines nécessitent d'appeler `touchline` ou `touchwin` sur la fenêtre parente.

Exemple de cadre

```
// Fenêtre principale (contenant le cadre)
fenetre = newwin(10, 15, 10, 2);
box(fenetre, 0, 0);
mvwprintw(fenetre, 0, 1, "_Titre_");

// Sous-fenêtre (contenant le texte)
sousFenetre = subwin(fenetre, 8, 13, 11, 3);
scrollok(sousFenetre, TRUE);
wprintw("Le_texte");

wrefresh(fenetre);
wrefresh(sousFenetre);
```

Problématique et solutions

- Lors du clic avec la souris, récupération des coordonnées (`getmouse`) :
↔ Par rapport au terminal et non par rapport aux fenêtres
- Première solution :
 - Test pour chaque fenêtre pour savoir si le clic est dans la fenêtre
 - Calcul des coordonnées de la souris en fonction de celles de la fenêtre
- Deuxième solution :
 - Utilisation de `wmouse_trafo`
 - Vérification + calcul des coordonnées en un seul appel

Consultez le [man](#) pour rechercher les fonctions `ncurses`

Table des matières

- 1 Présentation de ncurses
 - Introduction
 - Sortie écran
 - Gestion des saisies clavier
 - Gestion de la souris
- 2 Les fenêtres dans ncurses
 - Présentation des fenêtres
 - Fenêtres multiples
 - Souris et fenêtres
- 3 Conclusion et références

Quelques mots sur la structure des applications ncurses

- Comme toute application C : attention à la structuration de l'application
- Interface en `ncurses` : beaucoup de code !
- Conseils :
 - Créez vos propres structures (fenêtres, interface)
 ↪ En plus des autres nécessaires
 - Séparez le code associé aux fenêtres (autant que possible)
 - Encore : faites des interfaces simples !
 - Dessinez sur papier vos interfaces

Conclusion

- `ncurses` permet de réaliser des interfaces sommaires
- Intérêt : faciliter l'affichage pour certains projets
- Attention lors du développement :
 - ↪ Déboguage plus complexe
 - ↪ Problèmes d'affichage (notamment lors d'interruption d'exécution)
- Solution :
 - D'abord développer les fonctions hors `ncurses`
 - Ajouter ensuite `ncurses`
- Dans tous les cas : ne pas perdre de temps sur l'habillage !

N'utilisez pas d'autres bibliothèques que `ncurses` pour Info0601 !

Remarques sur `ncurses`

- Il sera demandé d'utiliser `ncurses` lors des TP
- L'utilisation doit être basique :
 - ↪ Les fioritures n'apporteront rien de plus
 - ↪ Une simple interface est la portée de tous et toutes
- Utilisez de préférence les 3/4 couleurs de base
- Attention aux boutons de souris ou aux touches utilisés
 - `ncurses` gomme les différences entre terminaux
 - Mais certains terminaux ont leur fonctionnements propres !
 - ↪ F1, par exemple, affiche l'aide dans certains systèmes
 - ↪ Le bouton droit ouvre le menu contextuel
 - ↪ ...

Bibliographie/Webographie

- *Programmer's Guide To NCurses*, Dan Gookin, Wiley (2007)
- *NCURSES Programming HOWTO*, Pradeep Padala (2005)
 - ↪ `http://www.tldp.org/HOWTO/NCURSES-Programming-HOWTO/`
 - ↪ Existe en version PDF
- Fonctions ncurses :
 - ↪ Utilisation du `man`
 - ↪ Ou en tapant la commande dans un moteur de recherche