by Reha K. Gerçeker
<gerceker/at/itu.edu.tr>

About the author:

Reha ist Student der Computer−Ingenieurwissenschaften in Istanbul, Türkei. Er mag die Freiheiten, die Linux als Software−Entwicklungsplattform bietet. Er verbringt viel seiner Zeit vor dem Computer mit dem Schreiben von Programmen. Er möchte eines Tages ein geschickter Programmierer werden.

Translated to English by:
Reha K. Gerçeker <gerceker/at/itu.edu.tr>

Einführung in Ncurses

Abstract:

Ncurses ist eine Bibliothek, die Funktionstasten−Belegung, Bildschirmbeschreibungs−Funktionen und die Fähigkeit bietet, mehrere sich nicht überlappende Fenster auf textbasierten Terminals zu benutzen.

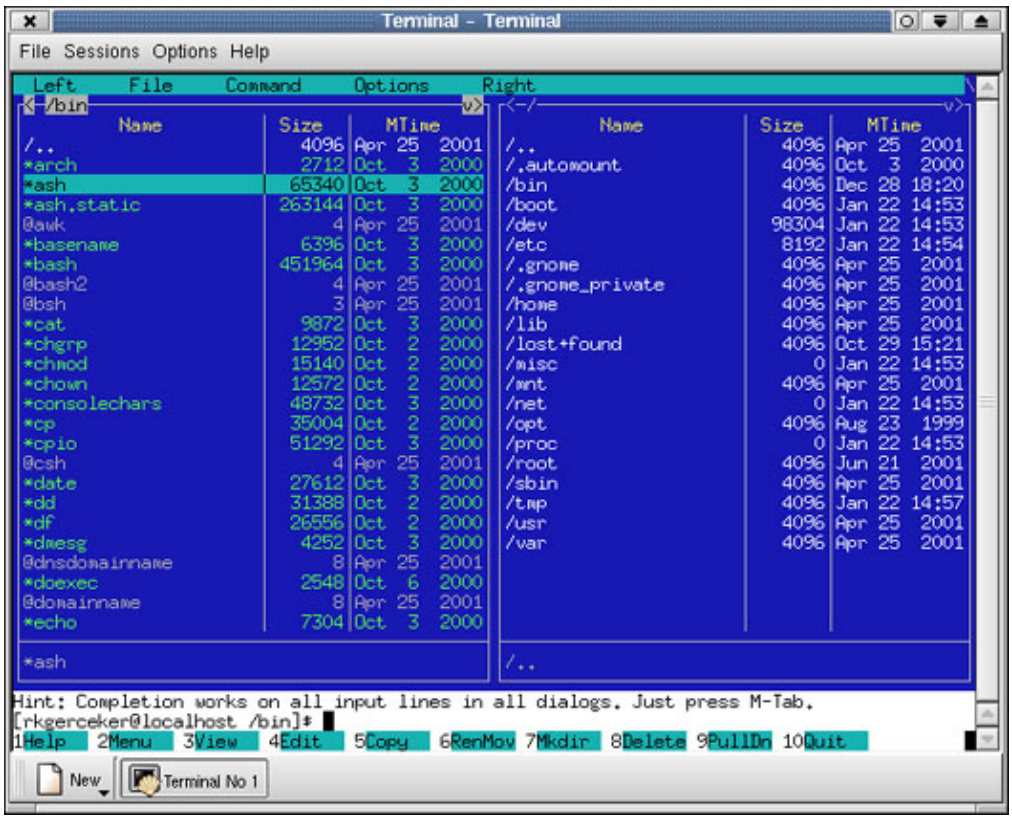
Was ist Ncurses?

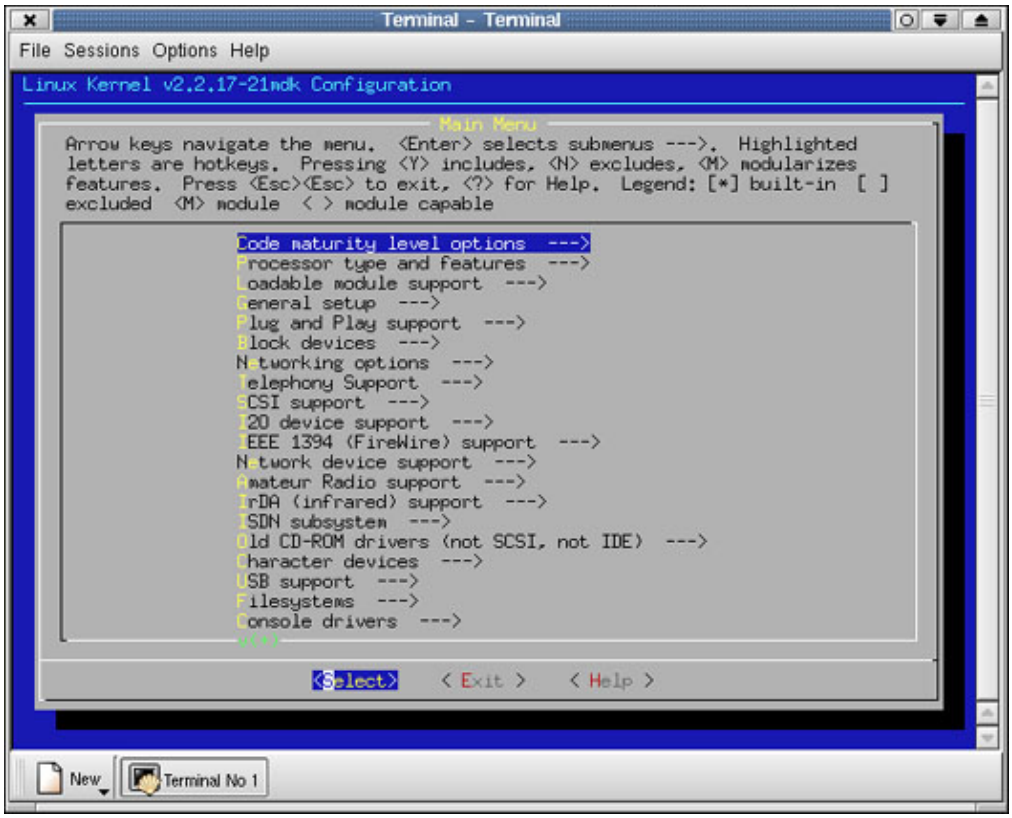
Möchten Sie, das Ihre Programme über eine farbenfrohe terminalbasierte Schnittstelle verfügen? Ncurses ist eine Bibliothek, die Fensterfunktionalitäten für textbasierte Terminals bereitstellt. Sachen, zu denen Ncurses fähig ist:

- Benutzen Sie den ganzen Bildschirm, wie Sie möchten.
- Erstellen und Verwalten von Fenstern.
- Nutzen von 8 unterschiedlichen Farben.
- Geben Sie Ihrem Programm Maus−Unterstützung.
- Benutzen Sie die Funktionstasten der Tastatur.

Es ist möglich, Ncurses auf jedem ANSI/POSIX−konformen UNIX−System zu nutzen. Davon abgesehen, ist die Bibliothek in der Lage, Terminal− Eigenschaften aus der Systemdatenbank zu erkennen und sich entsprechend zu verhalten, was zu einer terminal−unabhängigen Schnittstelle führt. Daher kann Ncurses beruhigt genutzt werden für Anwendungen, die auf unterschiedlichen Plattformen und verschiedenen Terminals funktionieren sollen.

Der Midnight Commander ist eines der Beispiele, das unter Benutzung von Ncurses geschrieben wurde. Auch die Schnittstelle für die Kernel−Konfiguration auf der Konsole ist mittels Ncurses geschrieben. Ihre Abbildungen sehen Sie untenstehend:





Wo kann ich es herunterladen?

Ncurses wird unter GNU/Linux entwickelt. Um die letzte Version zu erhalten, detaillierte Informationen zu bekommen und verwandte Links zu finden, besuchen Sie www.gnu.org/software/ncurses/.

Grundlagen

Um die Bibliothek zu benutzen, sollten Sie curses.h in Ihrem Quellcode einfügen und sicherstellen, dass Sie Ihren Code mit der curses−Bibliothek linken. Dies geschieht, indem man gcc mit dem Parameter −lcurses aufruft.

Wenn man unter Ncurses arbeitet, ist es notwendig, über eine grundlegende Datenstruktur Bescheid zu wissen. Diese ist die WINDOW−Struktur, die, wie man leicht aus dem Namen schliessen kann, benutzt wird, um die Fenster zu repräsentieren, die von Ihnen angelegt werden. Fast alle Funktionen der Bibliothek haben einen WINDOW−Zeiger als Parameter.

Die meistgenutzten Ncurses−Komponenten sind Fenster. Selbst wenn Sie keine eigenen Fenster erstellen, wird der Bildschirm als ein eigenständiges Fenster angesehen. Wie der FILE−Deskriptor stdout der Standard−I/O−Bibliothek den Bildschirm repräsentiert (falls keine Redirektion aktiv ist), so benutzt Ncurses den WINDOW−Zeiger stdscr für die gleiche Aufgabe. Zusätzlich zu stdscr ist ein weiterer WINDOW−Zeiger namens curscr in der Bibliothek definiert. Wie stdscr den Bildschirm repräsentiert, so repräsentiert curscr den aktuellen Bildschirm, wie er der Bibliothek bekannt ist. Sie mögen fragen: "Was ist der Unterschied?" Lesen Sie weiter.

Um die Ncurses−Funktionen und Variablen in Ihren Programmen nutzen zu können, müssen Sie die Funktion

initscr aufrufen. Diese Funktion alloziiert Speicher für Variablen wie stdscr, curscr und initialisiert die Bibliothek für die Benutzung. In anderen Worten: alle Ncurses−Funktionen müssen initscr folgen. In der gleichen Weise sollten Sie endwin aufrufen, wenn sie mit Ncurses fertig sind. Dies gibt den von Ncurses benutzten Speicher frei. Nach dem endwin−Aufruf können Sie keine Ncurses−Funktionen mehr benutzen, bevor Sie nicht wieder initscr aufrufen.

Zwischen den Aufrufen von initscr und endwin sollten Sie sicherstellen, das Sie keine Ausgaben über Funktionen der Standard−I/O−Bibliotheken zum Bildschirm schicken. Andernfalls könnten Sie eine ungewollte und normalerweise zerstörte Ausgabe auf dem Schirm erhalten. Wenn Ncurses aktiv ist, benutzen Sie dessen Funktionen, um Ausgaben an den Bildschirm zu senden. Vor dem Aufruf von initscr oder nach dem Aufruf von endwin können Sie tun, was Sie möchten.

Aktualisierung des Bildschirms: refresh

Die WINDOW−Struktur enthält nicht nur Höhe, Breite und Position des Fensters, sondern auch den Inhalt des Fensters. Wenn Sie in ein Fenster schreiben, ändert sich der Inhalt des Fensters, aber das bedeutet nicht, dass es unmittelbar auf dem Bildschirm erscheint. Um den Bildschirm zu aktualisieren, muss entweder refresh oder wrefresh aufgefufen werden.

Hier findet sich der Unterschied zwischen stdscr und curscr. Während curscr den Inhalt des aktuellen Schirms enthält, kann stdscr unterschiedliche Informationen nach Aufruf der Ncurses−Ausgabefunktionen enthalten. Wenn Sie die letzten Änderungen in stdscr nach curscr übertragen wollen, müssen Sie refresh aufrufen. In anderen Worten, refresh ist die einzige Funktion, die sich mit curscr befasst. Es sei empfohlen, dass Sie nicht mit curscr herumspielen, sondern es der refresh−Funktion überlassen, curscr zu aktualisieren.

refresh enthält einen Mechanismus, um den Bildschirm so schnell wie möglich zu aktualisieren. Wenn die Funktion aufgerufen wird, aktualisiert sie nur die geänderten Zeilen des Fensters. Dies spart CPU−Zeit, weil es das Programm davon abhält, die gleiche Information nochmals auf den Bildschirm zu schreiben. Dieser Mechanismus ist der Grund, warum Ncurses−Funktionen und Standard−I/O−Funktionen schlechte Ergebnisse produzieren, wenn sie zusammen benutzt werden; wenn Ncurses−Funktionen aufgerufen werden, setzen sie ein Flag (Merker in einer Variablen), das refresh darüber informiert, dass die Zeile geändert wurde; nichts dergleichen passiert, wenn Sie eine Standard−I/O−Funktion aufrufen.

Refresh und wrefresh bewirken eigentlich das gleiche. Wrefresh nimmt einen WINDOW−Zeiger als Parameter und aktualisiert nur den Inhalt des Fensters. Refresh() entspricht wrefresh(stdscr). Ich werde später noch darüber sprechen, dass die meisten Ncurses−Funktionen Makros nutzen, die wie wrefresh diese Funktionen auf stdscr anwenden.

Neue Fenster erstellen

Lassen Sie uns nun über subwin und newwin sprechen, das sind die Funktionen, die neue Fenster erstellen. Beide nehmen die Höhe, Breite und die Koordinaten der oberen linken Ecke des neuen Fensters als Parameter. Sie geben einen WINDOW−Zeiger zurück, der das neue Fenster repräsentiert. Sie können diesen neuen Zeiger mit wrefresh und anderen Funktionen, die ich später bespreche, benutzen.

"Wenn sie das gleiche bewirken, warum dupliziert man dann diese Funktionen?" könnten Sie nun fragen. Sie haben recht, sie sind leicht unterschiedlich. Subwin erstellt das neue Fenster als ein Unterfenster eines

anderen. Ein auf diese Art angelegtes Fenster erbt die Eigenschaften des Elternfensters. Diese Eigenschaften können später geändert werden ohne das übergeordnete Fenster zu beeinflussen.

Abgesehen davon gibt es eine Sache, die Eltern– und Kindfenster verbindet. Der Zeichenbereich, der den Inhalt eines Fensters enthält, wird von Eltern– und Kindfenster gemeinsam benutzt. Anders ausgedrückt, Zeichen an der Überschneidung zwischen beiden Fenstern können von jedem von ihnen geändert werden. Wenn das Elternfenster eine solche Überschneidung beschreibt, wird auch der Inhalt des Kindfensters geändert. Andersherum gilt das gleiche.

Im Gegensatz zu subwin erstellt newwin ein völlig neues Fenster. Solch ein Fenster teilt seinen Zeichenbereich mit keinem anderen Fenster, es sei denn, es selber besitzt eigene Kindfenster. Der Vorteil von subwin liegt darin, dass die Benutzung gemeinsamer Zeichenbereiche weniger Hauptspeicher verbraucht. Wenn Fenster sich jedoch gegenseitig überschreiben, hat die Benutzung von newwin seine eigenen Vorteile.

Sie können Ihre Kindfenster in beliebiger Tiefe anlegen. Jedes Kindfenster kann wiederum eigene Kindfenster haben, aber beachten Sie dann bitte, dass der gleiche Zeichenbereich von mehr als 2 Fenstern benutzt wird.

Wenn Sie ein von Ihnen erstelltes Fenster nicht mehr benötigen, können Sie es mit der Funktion delwin löschen. Ich schlage vor, Sie ziehen die Handbuchseiten für die Parameterlisten dieser Funktionen zu Rate.

In Fenster schreiben, aus Fenstern lesen

Wir haben über stdscr, curscr, das Aktualisieren des Bildschirms und das Erstellen neuer Fenster gesprochen. Aber wie schreiben wir in ein Fenster? Oder wie lesen wir Daten aus einem Fenster?

Die für diese Zwecke benutzten Funktionen entsprechen ihren Gegenstücken in der Standard–I/O–Bibliothek. Darunter sind printw anstelle von printf, scanw anstelle von scanf, addch für putc/putchar, getch für getc oder getchar. Sie werden normal benutzt, nur ihre Namen sind anders. In gleicher Weise könnte addstr benutzt werden, um eine Zeichenkette in ein Fenster zu schreiben und getstr, um eine Zeichenkette aus einem Fenster zu lesen. Alle diese Funktionen, die ein 'w' vor ihrem Namen hinzugefügt haben und einen WINDOW–Zeiger als ersten Parameter benutzen, erfüllen ihre Aufgabe in einem anderen Fenster als stdscr. Z. B. sind printw(...) und wprintw(stdscr, ...) äquivalent, ebenso wie refresh() und wrefresh(stdscr).

Es würde eine lange Geschichte werden, alle Details dieser Funktionen zu besprechen. Die Handbuchseiten sind die beste Quelle, um ihre Beschreibung, Prototypen, Rückgabewerte und andere Hinweise zu lesen. Ich schlage vor, Sie lesen die Handbuchseite zu jeder von Ihnen benutzten Funktion. Sie bieten detaillierte und wertvolle Informationen. Der letzte Abschnitt dieses Artikels, in dem ich ein Beispielprogramm zeige, kann auch als ein Tutorial über die Benutzung dieser Funktionen dienen.

Physische und logische Cursor

Es ist erforderlich, physische und logische Cursor zu erklären, nachdem wir über das Lesen aus und das Schreiben in Fenster gesprochen haben. Mit dem physischen Cursor ist der normalerweise blinkende Cursor auf dem Bildschirm gemeint und es gibt nur diesen einen. Andererseits gehört der logische Cursor zum Ncurses–Fenster und jedes Fenster hat einen logischen Cursor. Es kann also mehrere logische Cursor geben.

Der logische Cursor ist in dem Feld des Fensters, wo der Schreib− oder Lesevorgang beginnt. D. h., da man den logischen Cursor beliebig bewegen kann, bedeutet dies, dass Sie jederzeit jeden Punkt des Bildschirms oder Fensters beschreiben können. Dies ist ein Vorteil von Ncurses gegenüber der Standard−I/O−Bibliothek.

Die Funktion zum Bewegen des logischen Cursors ist entweder move oder, wie Sie leicht erraten können, wmove. Move ist die Makro−Version von wmove, geschrieben für stdscr.

Ein weiteres Thema ist die Koordination von physischem und logischen Cursoren. Die Position des physischen Cursors nach einem Schreibvorgang wird bestimmt durch das _leave−Flag, welches in der WINDOW−Struktur existiert. Wenn es gesetzt ist, wird der logische Cursor nach dem erfolgten Schreiben auf die Position des physischen gesetzt (wo das letzte Zeichen geschrieben wurde). Wenn _leave nicht gesetzt ist, wird der physische Cursor nach dem Schreiben auf die Position des logischen gesetzt (wo das erste Zeichen geschrieben wurde). Das _leave−Flag wird durch die leaveok−Funktion kontrolliert.

Die Funktion zum Bewegen des physischen Cursors ist mvcur. Im Gegensatz zu anderen Funktionen wirkt sich mvcur sofort aus und nicht erst beim nächsten refresh. Wenn Sie möchten, dass der physische Cursor nicht sichtbar ist, benutzen Sie die Funktion curs_set. Schauen Sie in die Handbuchseiten für Einzelheiten.

Es gibt ausserdem Makros, die die Setz− und Schreibfunktionen in einem einfachen Aufruf zusammenfassen. Diese werden gut erläutert in den gleichen Handbuchseiten, die sich mit addch, addstr, printw, getch, getstr, scanw usw. befassen.

Fensterputzen

Das Schreiben in Fenster ist geschafft. Aber wie löschen wir Fenster, Zeilen oder Zeichen?

Löschen in Ncurses bedeutet, die Zelle, Zeile oder den Fensterinhalt mit nicht sichtbaren Zeichen zu füllen. Funktionen, die ich unten erklärt habe, füllen die erforderlichen Felder mit nicht sichtbaren Zeichen und bereinigen daher den Bildschirm.

Lassen Sie uns zunächst über Funktionen sprechen, die sich mit der Löschung eines Zeichens oder einer Zeile befassen. Die Funktionen delch und wdelch löschen das Zeichen unter dem logischen Cursor des Fensters und rücken die folgenden Zeichen auf der gleichen Zeile nach links. deleteln und wdeleteln löschen die Zeile des logischen Cursors und rücken alle folgenden Zeilen nach oben.

Die Funktionen clrtoeol und wclrtoeol löschen alle Zeichen auf der gleichen Zeile rechts vom logischen Cursor. Clrtobot und wclrtobot rufen zuerst wclrtoeol auf, um alle Zeichen rechts vom logischen Cursor zu löschen und löschen dann alle folgenden Zeilen.

Daneben gibt es noch Funktionen, die den ganzen Schirm oder ein Fenster vollständig löschen. Es gibt zwei Methoden, einen ganzen Schirm zu löschen. Die erste ist es, alle Felder mit nicht sichtbaren Zeichen zu füllen und dann refresh aufzurufen, die andere benutzt die eingebauten Terminal−Steuerzeichen. Die erste Methode ist langsamer als die zweite, weil sie es erfordert, alle Felder des Schirms neu zu schreiben, während die zweite Methode den ganzen Schirm unmittelbar löscht.

Erase und werase füllen den Zeichenbereich eines Fensters mit nicht sichtbaren Zeichen. Beim nächsten refresh wird das Fenster neu geschrieben. Wenn das zu löschende Fenster jedoch den ganzen Bildschirm umfasst, ist es nicht besonders klug, diese Funktionen zu nutzen. Sie benutzen die oben zuerst beschriebene Methode. Wenn das zu löschende Fenster die ganze Bildschirmbreite einnimmt, ist es vorteilhaft, die untenstehenden Funktionen zu nutzen.

Bevor wir weitere Funktionen besprechen, ist es an der Zeit, das _clear−Flag zu erwähnen. Es existiert in der WINDOW−Struktur und wenn es gesetzt ist, weist es refresh an, das entsprechende Steuerzeichen an das Terminal zu senden. Wenn sie aufgerufen wird, prüft die refresh−Funktion ob das Fenster die volle Bildschirmbreite umfasst (mittels des _FULLWIN−Flags), und löscht dann den Schirm mittels der eingebauten Terminal−Methode. Dann schreibt es nur die sichtbaren Zeichen auf den Schirm. Dies macht das Löschen des Schirms viel schneller. Der Grund, warum die Terminal−Methode nur bei Fenstern benutzt wird, die die volle Bildschirmbreite umfassen, ist, dass das Terminal−Steuerzeichen den ganzen Bildschirm neu schreibt und nicht nur das Fenster. Das _clear−Flag wird von der Funktion clearok gesteuert.

Die Funktionen clear und wclear werden zum Löschen von Fenstern benutzt, die volle Bildschirmbreite haben. Tatsächlich entsprechen diese Funktionen dem Aufruf von werase und clearok. Zuerst füllen sie den Zeichenbereich des Fensters mit nicht sichtbaren Zeichen. Dann löschen sie durch Setzen des _clear−Flags den Schirm mittels der eingebauten Terminalmethode, wenn das Fenster volle Bildschirmbreite hat oder sie erneuern alle Felder des Fensters durch Beschreiben mit nicht sichtbaren Zeichen.

Als Ergebnis können wir festhalten, wenn Sie wissen, dass das zu löschende Fenster Bildschirmbreite besitzt, dann benutzen Sie clear oder wclear. Es produziert schneller das gewünschte Ergebnis. Jedoch macht es keinen Unterschied, wclear oder werase zu benutzen, wenn das Fenster nicht volle die Bildschirmbreite besitzt.

Benutzung von Farben

Die Farben, die Sie auf dem Schirm sehen, können Sie sich als ein Farbpaar vorstellen. Das kommt daher, weil jedes Feld eine Hintergrund− und eine Vordergrundfarbe besitzt. Farben in Ncurses zu benutzen heisst, Ihre eigenen Farbpaare zu erstellen und diese zum Schreiben in ein Fenster zu benutzen.

Genau wie initscr aufgerufen werden muss, um Ncurses zu starten, ist der Aufruf von start_color zum Initialisieren der Farbnutzung erforderlich. Die Funktion, die Sie zum Erstellen Ihrer Farbpaare benötigen, ist init_pair. Wenn Sie ein Farbpaar durch init_pair erstellen, wird dieses Paar mit der Nummer assoziiert, die Sie der Funktion als ersten Parameter übergeben haben. Wann immer Sie dann dieses Paar benutzen wollen beziehen Sie sich darauf, indem Sie COLOR_PAIR mit der verknüpften Nummer aufrufen.

Neben dem Erstellen von Farbpaaren benötigen Sie die erforderlichen Funktionen, die mit verschiedenen Farbpaaren schreiben. Dies wird erledigt von den Funktionen attron und wattron. Solange nicht attroff oder wattroff aufgerufen werden, bewirken diese Funktionen, das alles mit dem von Ihnen gewählten Farbpaar in das entsprechende Fenster geschrieben wird.

Es gibt auch die Funktionen bkgd und wbkgd, die das mit einem ganzen Fenster verbundene Farbpaar ändern. Wenn sie aufgerufen werden, ändern sie die Hintergrund− und Vordergrundfarbe aller Felder des Fensters. Das bedeutet, das beim nächsten refresh jedes Feld des Fensters mit dem neuen Farbpaar beschrieben wird.

Schauen Sie in den Handbuchseiten nach den verfügbaren Farben und den Einzelheiten der hier erwähnten Funktionen.

Fensterrahmen

Sie können Rahmen um Ihre Fenster erstellen, um Ihren Programmen ein schönes Aussehen zu verleihen. Es gibt ein Makro namens box in der Bibliothek, was dies für Sie erledigt. Es gibt allerdings kein wbox; box nimmt einen WINDOW−Zeiger als ein Argument.

Die einfachen Details zu box finden Sie in den Handbuchseiten. Etwas sollte hier erwähnt werden. Ein Fenster in einen Rahmen zu setzen, bedeutet einfach, die erforderlichen Zeichen in die Felder des Fensters zu schreiben, die mit den Rahmenfeldern korrespondieren. Wenn Sie später in diese Felder schreiben, wird der Rahmen zerstört. Um dieses zu verhindern, erstellen Sie mittels subwin ein inneres Fenster in dem Originalfenster, stellen das Originalfenster in einen Rahmen und benutzen das innere Fenster zum Schreiben, wenn es nötig ist.

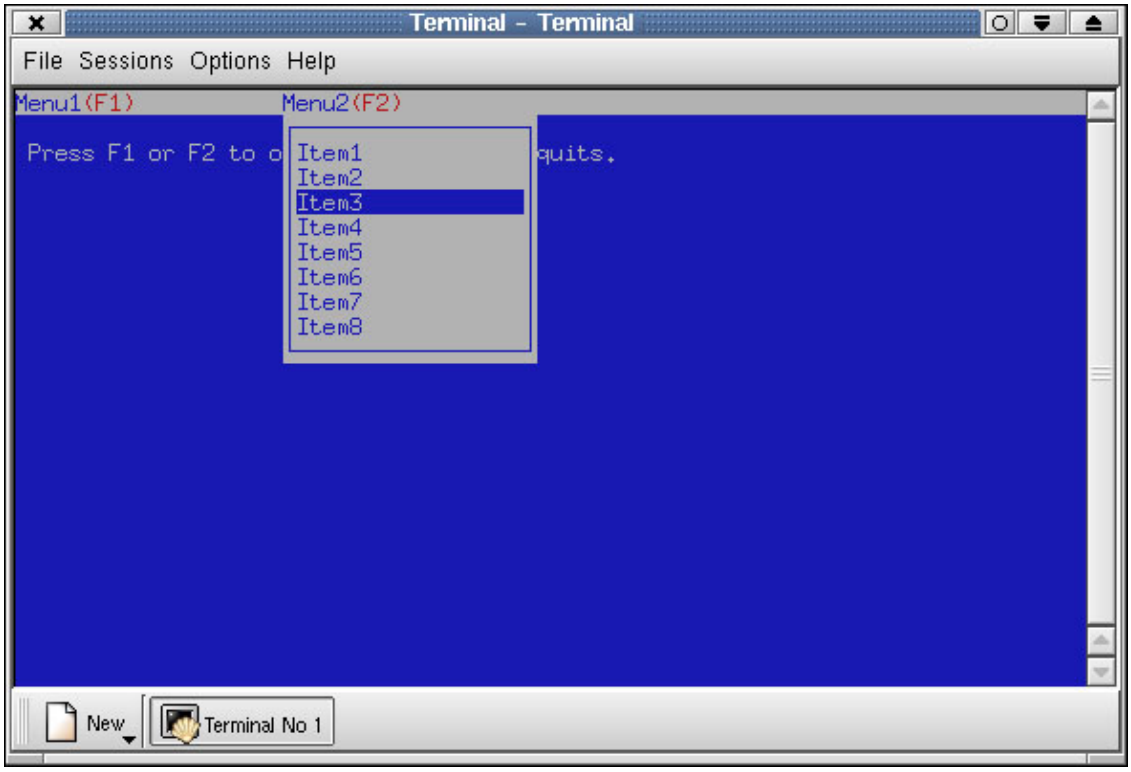
Funktionstasten

Um die Funktionstasten benutzen zu können, muss das _use_keypad−Flag in dem Fenster gesetzt sein, von dem Sie Eingaben erhalten. Keypad ist die Funktion, die den Wert von _use_keypad setzt. Wenn Sie _use_keypad setzen, können Sie Eingaben von der Tastatur wie üblich mit den Eingabe− Funktionen bekommen.

In diesem Fall, wenn Sie z. B. getch zum Abholen von Daten benutzen, sollten Sie sorgfältig darauf achten, die Daten in einer int−Variablen anstatt einer char−Variablen zu speichern. Das liegt daran, dass die numerischen Werte der Funktionstasten größer sind als die Werte, die eine char−Variable aufnehmen kann. Sie müssen diese numerischen Werte nicht kennen, stattdessen benutzen Sie deren definierten Namen aus der Bibliothek. Diese Namen sind in der Handbuchseite zu getch aufgeführt.

Ein Beispiel

Wir werden nun ein einfaches und nettes Programm analysieren. In diesem Programm werden Menues mittels Ncurses erstellt und die Auswahl einer Option aus einem Menue wird demonstriert. Ein interessanter Aspekt dieses Programms ist die Benutzung von Ncurses−Fenstern um einen Menü−Effekt zu erzeugen. Unten sehen Sie einen Schnappschuss.



Das Programm beginnt wie gewöhnlich mit den zu benutzenden Header–Dateien. Dann definieren wir Konstanten, die den ASCII–Werten der Enter– und der Escape–Taste entsprechen.

```
#include <curses.h>
#include <stdlib.h>

#define ENTER 10
#define ESCAPE 27
```

Die untenstehende Funktion wird zuerst aufgerufen, wenn das Programm läuft. Sie ruft zuerst initscr zur Initialisierung von Ncurses auf und dann start_color, um die Benutzung von Farben zu ermöglichen. Farbpaare, die während des Programmlaufs benutzt werden, werden später definiert. Der Aufruf curs_set(0) macht den physischen Cursor unsichtbar. Noecho bewirkt, das die Eingaben über die Tastatur nicht auf dem Bildschirm angezeigt werden. Sie können die noecho–Funktion auch benutzen, um die Eingaben von der Tastatur zu kontrollieren und nur die Teile anzuzeigen, die Sie anzeigen möchten. Die echo–Funktion sollte, wenn nötig, aufgerufen werden, um den Effekt von noecho rückgängig zu machen. Die untenstehende Funktion ruft keypad auf, um die Funktionstasten zu aktivieren, wenn Eingaben aus stdscr erwartet werden. Dies ist erforderlich, weil wir F1, F2 und die Cursortasten später im Programm benutzen werden.

```
void init_curses()
{
    initscr();
    start_color();
    init_pair(1,COLOR_WHITE,COLOR_BLUE);
    init_pair(2,COLOR_BLUE,COLOR_WHITE);
    init_pair(3,COLOR_RED,COLOR_WHITE);
    curs_set(0);
    noecho();
    keypad(stdscr,TRUE);
}
```

Die nächste Funktion erstellt die Menüleiste, die am oberen Rand des Bildschirms erscheint. Sie können die main–Funktion überprüfen und werden feststellen, das die Menueleiste, die als einzelne Zeile am oberen

Rand des Bildschirms erscheint, tatsächlich als ein einzeiliges Unterfenster von stdscr definiert ist. Die Funktion darunter nimmt den Zeiger auf dieses Fenster als Parameter, wechselt zunächst die Hintergrundfarbe und schreibt dann die Menü−Namen. Wir benutzen waddstr um die Menü−Namen zu schreiben, wir hätten auch eine andere Funktion benutzen können. Achten Sie auf den wattron−Aufruf, der benutzt wird, um mit einem anderen Farbpaar (Nummer 3) anstelle des Standard−Farbpaares (Nummer 2) zu schreiben. Denken Sie daran, dass Paar Nummer 2 als Standard durch wbkgd gesetzt wurde. Wattroff wird aufgerufen, wenn wir auf das Standard−Farbpaar zurückschalten wollen.

```
void draw_menubar(WINDOW *menubar)
{
    wbkgd(menubar,COLOR_PAIR(2));
    waddstr(menubar,"Menu1");
    wattron(menubar,COLOR_PAIR(3));
    waddstr(menubar,"(F1)");
    wattroff(menubar,COLOR_PAIR(3));
    wmove(menubar,0,20);
    waddstr(menubar,"Menu2");
    wattron(menubar,COLOR_PAIR(3));
    waddstr(menubar,"(F2)");
    wattroff(menubar,COLOR_PAIR(3));
}
```

Die nächste Funktion zeichnet die Menüs, wenn F1 oder F2 gedrückt wird. Um den Menü−Effekt zu erzielen, wird ein neues Fenster mit der gleichen weissen Farbe über dem blauen Fenster erzeugt, welches den Hintergrund ausmacht. Wir möchten nicht, dass dieses neue Fenster bereits im Hintergrund geschriebene Zeichen überschreibt. Sie sollten erhalten bleiben, wenn das Menü geschlossen wird. Darum kann das Menü−Fenster nicht als Unterfenster von stdscr erstellt werden. Wie Sie unten sehen, wird das Fenster items[0] mit der Funktion newwin und die anderen 8 items−Fenster als Unterfenster von items[0]] erstellt. items[0] wird hier benutzt, um einen Rahmen um das Menü zu zeichnen und die anderen items−Fenster werden genutzt, um den gewählten Eintrag im Menü zu zeigen und um auch nicht die Zeichen des Rahmens um das Menü herum zu überschreiben. Um einen Eintrag als selektiert anzuzeigen, reicht es aus, seine Hintergrundfarbe anders darzustellen als den Rest der Einträge. Das wird in der 3. Zeile von unten erledigt; die Hintergrundfarbe des ersten Eintrags wird geändert und wenn das Menü aufklappt, ist es der erste Eintrag, der selektiert ist.

```
WINDOW **draw_menu(int start_col)
{
    int i;
    WINDOW **items;
    items=(WINDOW **)malloc(9*sizeof(WINDOW *));

    items[0]=newwin(10,19,1,start_col);
    wbkgd(items[0],COLOR_PAIR(2));
    box(items[0],ACS_VLINE,ACS_HLINE);
    items[1]=subwin(items[0],1,17,2,start_col+1);
    items[2]=subwin(items[0],1,17,3,start_col+1);
    items[3]=subwin(items[0],1,17,4,start_col+1);
    items[4]=subwin(items[0],1,17,5,start_col+1);
    items[5]=subwin(items[0],1,17,6,start_col+1);
    items[6]=subwin(items[0],1,17,7,start_col+1);
    items[7]=subwin(items[0],1,17,8,start_col+1);
    items[8]=subwin(items[0],1,17,9,start_col+1);
    for (i=1;i<9;i++)
        wprintw(items[i],"Item%d",i);
    wbkgd(items[1],COLOR_PAIR(1));
    wrefresh(items[0]);
    return items;
}
```

Die nächste Funktion löscht einfach das Menü−Fenster, das von der obigen Funktion angelegt wurde. Es löscht zuerst das items−Fenster mittels delwin und gibt dann den Speicher frei, der für die items−Zeiger zugeordnet war.

```
void delete_menu(WINDOW **items,int count)
{
    int i;
    for (i=0;i<count;i++)
        delwin(items[i]);
    free(items);
}
```

Die scroll_menu−Funktion ermöglicht es, zwischen und innerhalb der Menüs zu wechseln. Sie liest mittels getch die auf der Tastatur gedrückten Tasten. Wenn die Cursor−rauf− oder Cursor−runter−Tasten gedrückt werden, dann wird der darüber oder darunter liegende Eintrag selektiert. Wie Sie sich erinnern, wird dies dadurch errreicht, das die Hintergrundfarbe des selektierten Eintrags sich von den anderen unterscheidet. Wenn Cursor−rechts oder Cursor−links gedrückt wird, wird das offene Menü geschlossen und das andere geöffnet. Wenn die Enter−Taster gedrückt wird, wird der gewählte Eintrag zurückgegeben. Wenn Escape gedrückt wird, werden die Menüs ohne Auswahl eines Eintrages geschlossen. Die Funktion ignoriert andere Eingabetasten. In dieser Funktion kann getch die Cursor−Tasten von der Tastatur lesen. Lassen Sie mich daran erinnern, das dies möglich ist, weil die erste Funktion init_curses die Funktion keypad(stdscr,TRUE) aufruft und der Rückgabewert von getch in einer int−Variablen anstatt einer char−Variablen gehalten wird, weil die Werte der Funktionstasten größer sind als eine char−Variable aufnehmen kann.

```
int scroll_menu(WINDOW **items,int count,int menu_start_col)
{
    int key;
    int selected=0;
    while (1) {
        key=getch();
        if (key==KEY_DOWN || key==KEY_UP) {
            wbkgd(items[selected+1],COLOR_PAIR(2));
            wnoutrefresh(items[selected+1]);
            if (key==KEY_DOWN) {
                selected=(selected+1) % count;
            } else {
                selected=(selected+count-1) % count;
            }
            wbkgd(items[selected+1],COLOR_PAIR(1));
            wnoutrefresh(items[selected+1]);
            doupdate();
        } else if (key==KEY_LEFT || key==KEY_RIGHT) {
            delete_menu(items,count+1);
            touchwin(stdscr);
            refresh();
            items=draw_menu(20-menu_start_col);
            return scroll_menu(items,8,20-menu_start_col);
        } else if (key==ESCAPE) {
            return -1;
        } else if (key==ENTER) {
            return selected;
        }
    }
}
```

Zum Schluß ist da noch die main−Funktion. Sie benutzt alle Funktionen, die ich geschrieben und oben beschrieben habe, damit das Programm ordentlich funktioniert. Sie liest außerdem mittels getch die gedrückten Tasten und zeichnet die entsprechenden Menüs, wenn F1 oder F2 gedrückt wird. Danach ruft sie scroll_menu auf und lässt die Benutzerin eine Auswahl in den Menüs vornehmen. Nachdem scroll_menu

beendet ist, löscht sie die Menü−Fenster und zeigt den gewählten Eintrag in der Nachrichtenzeile an. .

Ich sollte noch die Funktion touchwin erwähnen. Falls refresh direkt nach dem Schliessen der Menüs ohne touchwin aufgerufen wird, würde das letzte geöffnete Menü auf dem Bildschirm stehen bleiben. Dies kommt daher, weil die Menü−Funktionen stdscr gar nichts verändern und wenn refresh aufgerufen wird, wird nichts neu geschrieben, weil es annimmt, dass das Fenster nicht verändert wurde. Touchwin setzt alle Flags in der WINDOW−Struktur, die refresh signalisieren, dass sich alle Zeilen des Fensters geändert haben und daher muss beim nächsten refresh−Aufruf das gesamte Fenster neu gezeichnet werden, obwohl sich der Inhalt des Fensters nicht geändert hat. Die Informationen, die in stdsrc geschrieben wurden, bleiben nach dem Schliessen der Menüs dort, weil die Menüs nicht auf stdscr schreiben, sondern als neue Fenster angelegt wurden.

```
int main()
{
    int key;
    WINDOW *menubar,*messagebar;

    init_curses();

    bkgd(COLOR_PAIR(1));
    menubar=subwin(stdscr,1,80,0,0);
    messagebar=subwin(stdscr,1,79,23,1);
    draw_menubar(menubar);
    move(2,1);
    printw("Press F1 or F2 to open the menus. ");
    printw("ESC quits.");
    refresh();

    do {
        int selected_item;
        WINDOW **menu_items;
        key=getch();
        werase(messagebar);
        wrefresh(messagebar);
        if (key==KEY_F(1)) {
            menu_items=draw_menu(0);
            selected_item=scroll_menu(menu_items,8,0);
            delete_menu(menu_items,9);
            if (selected_item<0)
                wprintw(messagebar,"You haven't selected any item.");
            else
                wprintw(messagebar,
                  "You have selected menu item %d.",selected_item+1);
            touchwin(stdscr);
            refresh();
        } else if (key==KEY_F(2)) {
            menu_items=draw_menu(20);
            selected_item=scroll_menu(menu_items,8,20);
            delete_menu(menu_items,9);
            if (selected_item<0)
                wprintw(messagebar,"You haven't selected any item.");
            else
                wprintw(messagebar,
                  "You have selected menu item %d.",selected_item+1);
            touchwin(stdscr);
            refresh();
        }
    } while (key!=ESCAPE);

    delwin(menubar);
    delwin(messagebar);
    endwin();
```

```
    return 0;
}
```

Wenn Sie den Programmtext in eine Datei namens example.c kopieren und meine Erläuterungen entfernen, können Sie den Code kompilieren mittels:

```
gcc -Wall example.c -o example -lcurses
```

und das Programm testen. Sie können den Programmtext auch herunterladen (s. Abschnitt Referenzen).

Zusammenfassung

Ich habe über die grundlegenden Kenntnisse in Ncurses gesprochen, die ausreichend sind, um eine gute Schnittstelle/Oberfläche für Ihr Programm zu erstellen. Die Fähigkeiten der Bibliothek sind jedoch nicht auf das hier erläuterte beschränkt. Sie werden viele weitere Sachen in den Handbuchseiten entdecken. Ich habe Sie oft gebeten, diese zu lesen und Sie werden verstehen, dass die hier präsentierten Informationen nur eine Einführung sind.

Referenzen

- Das Beispielprogramm: example.c
- Die Ncurses−Webseite:www.gnu.org/software/ncurses/

2005−01−11, generated by lfparser_pdf version 2.51