Inhaltsverzeichnis

1 Einleitung **5**

2 Grundlegendes **7**

2.1 Installation 7
2.2 Die Headerdateien 7
2.3 Die Bibliotheken 8
2.4 Die ncurses-man-Pages 8
2.5 Fenster 8
2.6 Bildschirme 9
2.7 ncurses und I/O-Befehle von C 9
2.8 Termcap und Terminfo 9
2.9 Ncurses initialisieren und beenden 10
2.10 Das ncurses-Hauptkoordinatensystem 11
2.10.1 Beispiel 11
2.11 Cursor 13
2.12 Textausgabe 14
2.12.1 Hinzufügen von Einzelzeichen 14
2.12.2 Einfügen eines Zeichens 14
2.12.3 Hinzufügen einer Zeichenkette 15
2.12.4 Einfügen einer Zeichenkette 16
2.12.5 Formatierte Ausgabe 16
2.13 Texteingabe 17
2.13.1 Eingabe eines Zeichens 18
2.13.2 Eingabe einer Zeichenkette 18
2.13.3 Formatierte Eingabe 18
2.14 Löschaktionen 18
2.15 Refresh 19
2.16 Scrolling 19
2.17 Echo ein-/ausschalten 19
2.18 Pieps und Blitz 20

3 Farben und andere Attribute **21**
3.1 Beispiel 21
3.2 Farben einstellen 22
3.2.1 Basisfarben 23
3.2.2 Textvorder- und -hintergrundfarbe 23
3.2.3 Fensterhintergrund 24
3.3 Zusätzliche Textattribute 26
3.3.1 Beispiel 27

4 Spezialtasten **29**
4.1 Beispiel 29

5 Fenster **33**
5.1 Neue Fenster 33
5.1.1 Beispiel 34
5.2 Abgeleitete Fenster 35
5.2.1 Beispiel: Aufzeigen des Unterschieds von newwin und subwin 36
5.3 Fensterverzierungen 37
5.3.1 Rahmen 38
5.3.2 Linien 38
5.3.3 Beispiel 39
5.3.4 ACS (Liniengrafik, form characters) 40
5.4 Fenster löschen 44
5.5 Fenster refreshen 44
5.6 Touch und Untouch 44
5.6.1 Beispiel 45

6 Panels **47**
6.1 Beispiel 47

7 Menüs **51**
7.1 Ein Menü erzeugen und wieder löschen 51
7.2 Der Menü-Treiber 52
7.3 Den aktuell angewählten Menüeintrag ermitteln 53
7.3.1 Beispiel 53
7.4 Das Menü formatieren 55
7.4.1 Beispiel 55
7.5 Das Markierungssymbol 57
7.5.1 Beispiel (Programmausschnitt) 58
7.5.2 Beispiel (Programmausschnitt) 58

7.6 Menüfenster 59
7.6.1 Beispiel 59
7.7 Menüs bunt gestalten 61
7.7.1 Beispiel 62
7.8 Optionen für Menüeinträge 64
7.8.1 Beispiel 65
7.9 Menüoptionen 67
7.9.1 Beispiel: O_SHOWDESC 68
7.9.2 Beispiel: Mehrfachauswahl 68

8 Formulare **71**
8.1 Ein Formular erzeugen und wieder löschen 71
8.2 Der Formulartreiber 72
8.3 Feldfarben und andere Darstellungsattribute 72
8.3.1 Beispiel 73
8.4 Zugriff auf den Formularfeldpuffer 75
8.4.1 Beispiel 75
8.5 Textausrichtung 78
8.5.1 Beispiel (Programmausschnitt) 78
8.6 Feldoptionen 78
8.6.1 Beispiel: Auswirkungen O_AUTOSKIP und O_STATIC . 79
8.7 Formularfeldtypen 80
8.7.1 Beispiel 81
8.8 Formularfenster 83

9 Mausunterstützung **85**
9.1 Beispiel 86

10 Pads **89**
10.1 Ein Pad erstellen 89
10.2 Ein Pad refreshen 89
10.3 Beispiel 90

11 Was fehlt **93**

12 Weblinks **95**
12.1 Weitere Informationen zu ncurses 95
12.2 Downloadmöglichkeit 95
12.3 Andere curses-Bibliotheken 95

13 Autoren **97**

14 GNU Free Documentation License **99**

15 Bildnachweis **103**

Kapitel 1

Einleitung

Ncurses ist eine C-Bibliothek für die Steuerung von Textterminals. Hauptzweck dieser Bibliothek ist die Erstellung von TUIs (Text User Interfaces). Typische Beispiele für Programme, deren Benutzeroberflächen die *ncurses*-Bibliothek benutzen sind der Lynx-Browser (alternativ zur *S-Lang*-Bibliothek), der GNU Midnight Commander (alternativ zur *S-Lang*-Bibliothek), das Linux-Kernel-Konfigurationsprogramm in der *ncurses*-Variante oder das ncurses-Frontend des YaST-Installations- und -Konfigurationsprogramms bei der SuSE-Linux-Distribution.

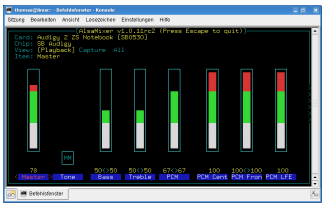

Abbildung 1: alsamixer (Audio-Mischer für ALSA)

Abbildung 2: Lynx (Webbrowser)

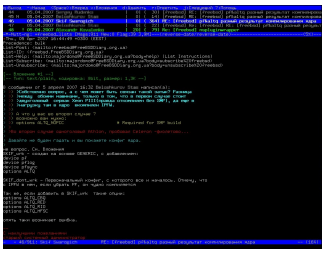

Abbildung 3: Mutt (E-Mail-Programm)

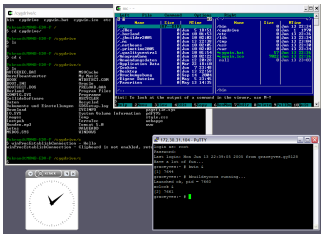

Abbildung 4: Midnight Commander (Dateimanager, oben rechts)

Ncurses ist weitgehend kompatibel mit den SVR4-curses. *Ncurses* ist für diverse Unix-Plattformen erhältlich und steht unter der MIT-Lizenz.

Neben *ncurses* sind auch andere *curses*-Weiterentwicklungen erhältlich. Als Beispiel sei *PDCurses* genannt, welches für MS DOS, MS Windows, OS/2 und X11-Systeme verfügbar ist. Dieses Buch bezieht sich jedoch explizit auf *ncurses*, d.h. die erläuterten Grundlagen und Beispiele können, müssen jedoch nicht auf andere Curses-Bibliotheken gleichermaßen zutreffen.

Kapitel 2

Grundlegendes

2.1 Installation

ncurses ist zumindest unter Linux (fast) immer Bestandteil der Distributionen und kann, wenn nicht bereits bei der initialen Linuxinstallation geschehen, nachträglich einfach vom Installationsmedium installiert werden. Eine *ncurses*-Downloadmöglichkeit wird im Kapitel Weblinks genannt.

2.2 Die Headerdateien

Die ncurses-Standard-Headerdateien sind

- `curses.h`

oder

- `ncurses.h`

`ncurses.h` ist meist nur ein symbolischer Link auf `curses.h`

Desweiteren enthält *ncurses* für Spezialeinsatzfälle noch weitere Header-Dateien:

- Panels: `panel.h`
- Menüs: `menu.h`
- Formulare: `form.h`

2.3 Die Bibliotheken

Die ncurses-Standardfunktionen sind in der Bibliothek

- `libncurses.xx`

versammelt, wobei `xx` für `so` (shared object) oder `a` (statische Programmbibliothek) steht.

Äquivalent zu den Header-Dateien gibt es noch zusätzliche *ncurses*-Bibliotheken:

- Panels: `libpanel.xx`
- Menüs: `libmenu.xx`
- Formulare: `libform.xx`

2.4 Die ncurses-man-Pages

Mit *ncurses* werden umfangreiche und ausführliche Manual-Seiten mitgeliefert.

Der Aufruf der *ncurses*-Übersichtsseite erfolgt mittels `man ncurses`

oder `info ncurses`

Auch die Beschreibungen der einzelnen *ncurses*-Funktionen lassen sich so abfragen, z.B. `man mvaddstr`

oder `man 3ncurses mvaddstr`.

2.5 Fenster

Ein Fenster (Window) repräsentiert einen rechteckigen Bildschirmausschnitt. In einer `WINDOW`-Datenstruktur werden die notwendigen Attribute und Daten des entsprechenden Bildschirmausschnittes gespeichert. In `curses.h` findet sich dieses Konstrukt unter `struct _win_st`. `WINDOW` ist nur ein Synonym für `_win_st` (`typedef struct _win_st WINDOW`).

2.6 Bildschirme

Bildschirme (Screens) sind Zeiger auf eine WINDOW-Datenstruktur.

ncurses kennt beim Start zwei Screens:

- Standardbildschirm (standard screen): `stdscr`
- Aktueller Bildschirm (current screen): `curscr`

Der `stdscr` ist das Hauptfenster, welches vom Programmierer mittels *ncurses*-Anweisungen beeinflusst werden kann. Der `curscr` repräsentiert das, was momentan am Bildschirm angezeigt wird. Bei einem Refresh werden in diesen Screen die vorgenommenen Änderungen eingebracht und die Darstellung am physikalischen Bildschirm aktualisiert.

2.7 ncurses und I/O-Befehle von C

Befindet sich ein Programm im *ncurses*-Modus, so dürfen (oder sollen, können) die C-Standardfunktionen nicht zur Ein- und Ausgabe (z.B. printf, scanf) verwendet werden. Zu diesem Zweck stellt *ncurses* eigene Funktionen zur Verfügung.

2.8 Termcap und Terminfo

termcap steht für *Terminal Capabilities* und ist eine Datei, in der die Fähigkeiten (Zeilen-, Spaltenanzahl, ...) zahlreicher Terminals hinterlegt sind.

terminfo leistet prinzipiell das gleiche wie *termcap*. Allerdings liegen in der *terminfo*-Datenbank die Terminal-Fähigkeitsbeschreibungen nicht in einer einzigen Gesamtdatei vor, sondern als separate Dateien alphabetisch geordnet in Unterverzeichnissen.

Bei der *ncurses*-Initialisierung wird mittels der Environmentvariablen `TERM` die *termcap*/*terminfo* ausgewertet. Daraus kann *ncurses* die relevanten Daten des verwendeten Terminals ermitteln (z.B. Anzahl der Zeilen und Spalten).

2.9 Ncurses initialisieren und beenden

Die ncurses-Initialisierung geschieht mittels der Funktion `WINDOW *initscr (void);`

`initscr` wird in den meisten Fällen die erste *ncurses*-Funktion sein, die in einem Programm aufgerufen wird.

Beendet wird der ncurses-Modus mit der Funktion `int endwin (void);`

Ein *ncurses*-Programm soll unbedingt ordentlich mit der `endwin`-Funktion beendet werden.

Der *ncurses*-Modus kann durch `endwin` auch zeitweilig unterbrochen werden. Die Rückkehr zum *ncurses*-Modus erfolgt durch eine Refresh-Anweisung.

Mit der Funktion `bool isendwin()` kann abgefragt werden, ob der *ncurses*-Modus noch aktiv ist.

`bool` ist übrigens ein Boolean-Datentyp den *ncurses* mitbringt. Zulässige Werte sind `TRUE` (= 1) und `FALSE` (= 0). Dass C++-Compiler oder andere Bibliotheken ebenfalls einen `bool`-Datentyp kennen, wird in der *ncurses*-Headerdatei berücksichtigt.

2.10 Das ncurses-Hauptkoordinatensystem

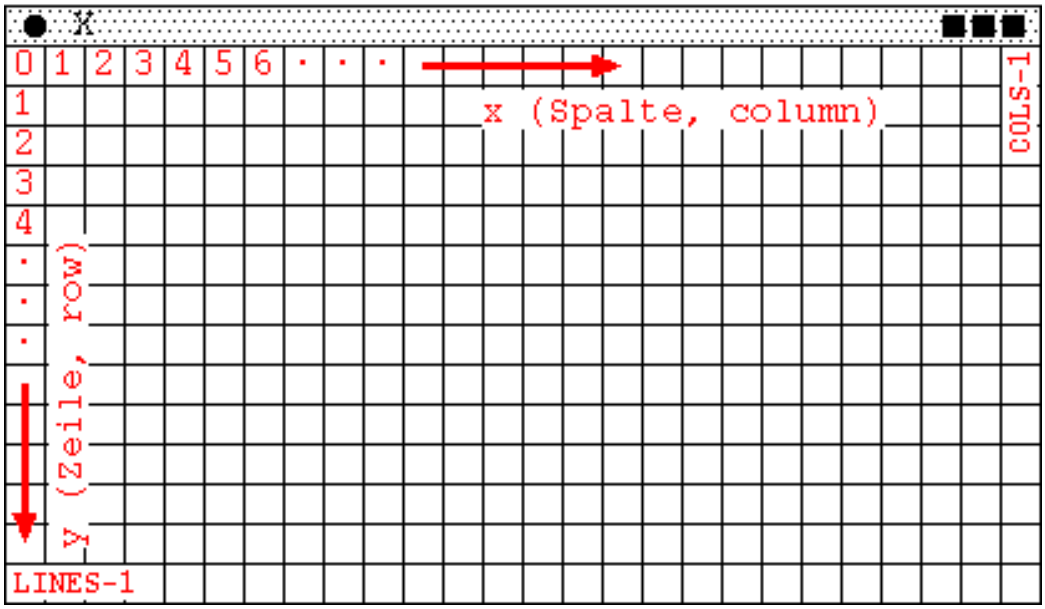

Abbildung 5: Bild:Ncurses_koord.png

Zwecks Abfrage von Fenster- und Cursorkoordinaten stehen folgende Makros zur Verfügung:

```
getyx(win, y, x);     // Koordinaten der aktuellen Cursorposition
getparyx(win, y, x);  // Koordinatenursprung von win bezogen auf das
                      // übergeordnete Fenster
getbegyx(win, y, x);  // Koordinatenursprung
getmaxyx(win, y, x);  // Fenstergröße (Anzahl der Zeilen und Spalten)
```

Wobei `win` ein Zeiger auf das abzufragende `WINDOW` ist und die Koordinatenwerte als Integerwerte in `y` (Zeile) und `x` (Spalte) geliefert werden.

2.10.1 Beispiel

```
#include <curses.h>
#include <stdlib.h> //noetig fuer atexit()


void quit()
{
```

```
  endwin();
}

int main(void)
{
  int x, y;

  initscr();
  atexit(quit);
  curs_set(0);

  mvprintw(3, 5, "LINES: %d", LINES);
  mvprintw(4, 5, "COLS:  %d", COLS);

  getyx(stdscr, y, x);
  mvprintw(5, 5, "Momentane Cursorposition:  [%d, %d]", y, x);

  getbegyx(stdscr, y, x);
  mvprintw(6, 5, "Koordinatenursprung:       [%d, %d]", y, x);

  getmaxyx(stdscr, y, x);
  mvprintw(7, 5, "Fenstergröße:              [%d, %d]", y, x);

  mvaddstr(11, 2, "Taste drücken -> Ende");
  refresh();

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

Die Bedeutung der einzelnen Bildschirmausgabefunktionen wird später näher beschrieben. Das Compilieren und Linken des Beispieles kann mit folgendem Kommando geschehen

```
gcc -o bsp bsp.c -lncurses
```

Voraussetzungen für das erfolgreiche Compilieren/Linken dieses Beispiels in der vorgeschlagenen Art und Weise:

- Das Beispiel wurde unter dem Dateinamen `bsp.c` abgespeichert und der Compileraufruf erfolgt aus dem selben Verzeichnis in dem `bsp.c` gespeichert wurde.

- Sie verwenden das C-Frontend der GNU Compiler Collection (GCC). Diese GCC muss für das Compilieren/Linken des Beispiels auf ihrem System installiert sein. Die Verwendung eines anderen Compilers wird einen etwas anderen Compileraufruf erfordern.
- Sie geben das Kommando direkt nach dem Prompt in ein Terminal ein. Für andere Arten der Programmerstellung, z.B. via IDE, sei hier nur der allgemeine Hinweis *"die Einbindung der* `ncurses`*-Bibliothek nicht vergessen"* gegeben.

Das ausführbare Programm sollte nach erfolgreichem Compilerlauf unter dem Dateinamen `bsp` vorhanden sein. Der Programmstart kann mit

```
./bsp
```

direkt aus dem Speicherverzeichnis des Beispielprogramms erfolgen. Das Ergebnis sollte folgendem Screenshot ähneln.

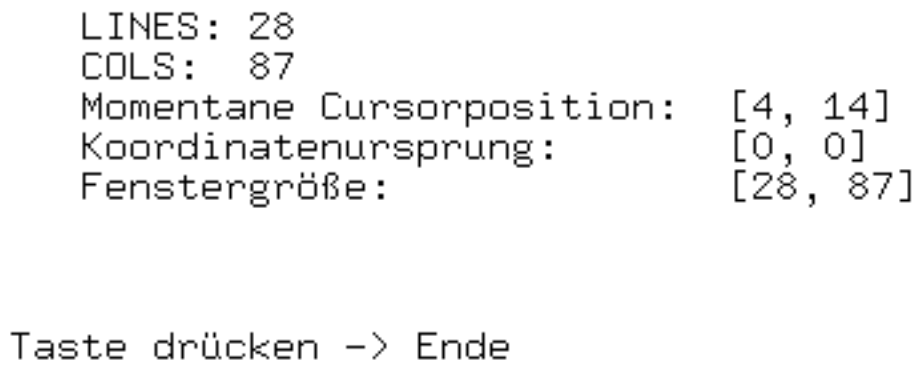

Abbildung 6: Bild:Ncurses55.png

Die konkret angezeigten Zahlenwerte hängen natürlich vom verwendeten Terminal ab.

2.11 Cursor

Zwecks Positionierung des Cursors steht die Funktion `int move(int y, int x);`

zur Verfügung. Der Cursor bestimmt, an welcher Position die Textausgabe oder ein Echo erfolgt. *ncurses*-Text-I/O-Funktionen existieren meist zusätzlich als Variante mit einem `mv`-Präfix, sodass Cursorpositionierung und Daten-I/O in einem Rutsch erledigt werden können und nicht jeweils zwei Funktionen nacheinander angeschrieben werden müssen.

Die Cursoranzeige lässt sich mit der Funktion `int curs_set(int visibility);`

modifizieren. Für `visibility` sind folgende Werte möglich:

- 0 ... unsichtbar
- 1 ... sichtbar
- 2 ... ”besonders” sichtbar.

2.12 Textausgabe

Die *ncurses*-Bibliothek kennt, ihrem Einsatzzweck entsprechend, ein Vielzahl von Textausgabefunktionen.

2.12.1 Hinzufügen von Einzelzeichen

```
int addch(const chtype ch);
int mvaddch(int y, int x, const chtype ch);
int echochar(const chtype ch);
```

`echochar` entspricht einem `addch` mit nachfolgendem `refresh`.

2.12.2 Einfügen eines Zeichens

```
int insch(chtype ch);
int mvinsch(int y, int x, chtype ch);
```

Beispiel: Unterschied zwischen addch() und insch()

```
#include <curses.h>
#include <stdlib.h> //noetig fuer atexit()
```

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

int main(void)
{
  initscr();
  atexit(quit);
  curs_set(0);

  mvaddstr(3, 2, "Der Unterschied zwischen addch() und insch():");

  mvaddstr(5, 5, "ADDCH: Hallo, Welt");
  mvaddch(5, 13, 'A');

  mvaddstr(6, 5, "INSCH: Hallo, Welt");
  mvinsch(6, 13, 'A');
  refresh();

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

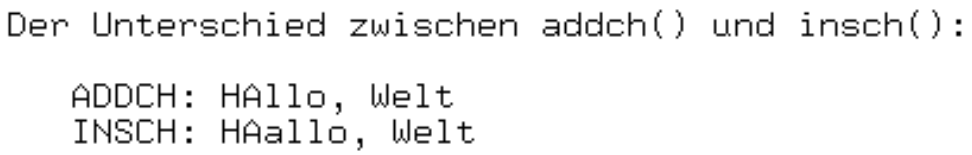

Abbildung 7: Bild:Ncurses56.png

2.12.3 Hinzufügen einer Zeichenkette

```
int addstr(const char *str);
int addnstr(const char *str, int n);
int mvaddstr(int y, int x, const char *str);
int mvaddnstr(int y, int x, const char *str, int n);
```

2.12.4 Einfügen einer Zeichenkette

```
int insstr(const char *str);
int insnstr(const char *str, int n);
int mvinsstr(int y, int x, const char *str);
int mvinsnstr(int y, int x, const char *str, int n);
```

2.12.5 Formatierte Ausgabe

```
int printw(const char *fmt, ...);
int mvprintw(int y, int x, const char *fmt, ...);
int vwprintw(WINDOW  *win,  const char *fmt, va_list var glist);
```

Die Parameter, insbesondere `fmt`, von `printw` entsprechen der `printf`-Funktion in `stdio.h`. Nachfolgend sind auszugsweise einige Konvertierungszeichen zwecks Verwendung in `fmt` gegeben

%d, %i	vorzeichenbehaftete Ganzzahl (int)
%o	vorzeichenlose Ganzahl im Oktalformat
%u	vorzeichenlose Ganzzahl (unsigned int)
%x	vorzeichenlose Ganzahl im Hexadezimalformat
%f, %F	Gleitkommazahl (double)
%e, %E	Gleitkommazahl (double) in Exponentialdarstellung
%a, %A	Gleitkommazahl (double) in Hexadezimaldarstellung
%s	Zeichenkette (const char *)
%c	Ein Zeichen (char)

Beispiel

```
#include <curses.h>
#include <stdlib.h> //noetig fuer atexit()

void quit()
```

```
{
  endwin();
}

int main(void)
{
  const int    i    = 23456;
  const double f    = -12345e-3;

  initscr();
  atexit(quit);
  curs_set(0);

  mvprintw(3, 2, "Eine Ganzzahl in Oktal- und Hexadezimaldarstellung: %o | %x\n", i, i);
  mvprintw(4, 2, "Eine Ganzzahl mit führenden Nullen: %010d\n", i);
  mvprintw(5, 2, "Eine Gleitkommazahl: %8.3f\n", f);
  mvprintw(6, 2, "Eine Gleitkommazahl: %8.3e\n", f);
  mvprintw(7, 2, "Eine Gleitkommazahl in Hexadezimaldarstellung: %a\n", f);

  refresh();

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

```
Eine Ganzzahl in Oktal- und Hexadezimaldarstellung: 55640 | 5ba0
Eine Ganzzahl mit führenden Nullen: 0000023456
Eine Gleitkommazahl:  -12.345
Eine Gleitkommazahl: -1.235e+01
Eine Gleitkommazahl in Hexadezimaldarstellung: -0x1.8b0a3d70a3d71p+3
```

Abbildung 8: Bild:Ncurses300.png

2.13 Texteingabe

Eine Benutzeroberfläche ohne die Möglichkeit von Benutzereingaben ist in den meisten Fällen relativ nutzlos. Die *ncurses*-Bibliothek stellt eine Reihe von Funktionen für die Dateneingabe mittels Tastatur zur Verfügung.

2.13.1 Eingabe eines Zeichens

```
int getch(void);
int mvgetch(int y, int x);
int ungetch(int ch);
int wgetch(WINDOW *win);
```

`getch` entspricht in etwa der `getchar`-Funktion aus `stdio.h`. `ungetch` schreibt das Zeichen `ch` zurück in die Eingabe-Queue. `wgetch` ist eine Funktion für die Zeicheneingabe in eigens festgegelegten Fenstern. Die Programmierung von Fenstern wird aber erst in einem der folgenden Kapitel näher beschrieben.

Die Interpretation eines Return-Tastendrucks ist nicht einheitlich geregelt. *Ncurses* bietet zu diesem Zwecke zwei unterschiedliche Funktionen.

```
int nl(void);
int nonl(void);
```

Bei der Verwendung von `nl()` wird ein Return-Tastendruck (Zeilenendezeichen) als 0xA (10dec, LF, Line Feed) interpretiert. Bei der Verwendung von `nonl()` entspricht ein Return-Tastendruck nur einem 0xD (13dec, CR, Carriage Return).

2.13.2 Eingabe einer Zeichenkette

getstr(char *str); getnstr(char *str, int n); mvgetstr(int y, int x, char *str); mvgetnstr(int y, int x, char *str, int n);

2.13.3 Formatierte Eingabe

```
int scanw(char *fmt, ...);
int mvscanw(int y, int x, char *fmt, ...);
int vwscanw(WINDOW *win, char *fmt, va_list varglist);
```

`scanw` ist vergleichbar mit dem konventionellen `scanf`.

2.14 Löschaktionen

Ab und zu wird auch eine ”tabula rasa”-Aktion fällig.

```
int clear(void);
int clrtobot(void);
int clrtoeol(void);
```

`clear` löscht den gesamten Standardscreen. `clrtobot` löscht die Inhalte von der aktuellen Cursorposition bis zum Ende des Screens. `clrtoeol` löscht die Inhalte von der aktuellen Cursorposition bis zum Zeilenende.

Die Funktion `int deleteln(void);`

löscht die gesamte aktuelle Zeile. Alle darauffolgenden Zeilen werden um eine Zeile nach oben verschoben. Andererseits fügt die Funktion `int insertln(void);`

eine Leerzeile an der aktuellen Cursorposition ein. Alle folgenden Zeilen werden um eine Zeile nach unten verschoben.

2.15 Refresh

Damit die Änderungen im `stdscr` in den `curscr` übernommen werden und somit der Bildschirm aktualisiert wird, ist folgende Anweisung vorhanden:

```
int refresh(void);
```

2.16 Scrolling

```
int scroll(WINDOW *win);
int scrl(int n);
```

`scroll` verschiebt den Fensterinhalt um eine Zeile nach oben. `scrl` verschiebt den Fensterinhalt von `stdscr` um `n` Zeilen nach oben.

Damit Scrolling funktioniert, muss vorab folgende Funktion aufgerufen werden.

```
int scrollok(WINDOW *win, bool bf);
```

Der Parameter `bf` muss für die Aktivierung der Scrollfähigkeit natürlich auf `TRUE` gesetzt werden.

2.17 Echo ein-/ausschalten

Die Anzeige der Eingabedaten (das Echo) kann mittels der Funktion `noecho()`

unterdrückt werden.

```
echo()
```

schaltet diese Anzeige wieder ein. Dies ist das Standardverhalten von *ncurses*.

2.18 Pieps und Blitz

Manchmal kann auch ein kurzes Warnsignal an den Benutzer sinnvoll sein. *Ncurses* bietet zu diesem Zweck zwei standardmäßig vorhandene Funktionen:

```
int beep(void);
int flash(void);
```

`beep()` soll einen kurzen akustischen Ton und `flash()` einen kurzen optischen "Flash" erzeugen. Das Ganze ist aber stark abhängig von den Fähigkeiten und Einstellungen des verwendeten Terminals. Besitzt das verwendete Terminal weder Audio- noch Flashfähigkeiten, dann erfolgt weder eine akustische noch eine visuelle Warnung. Teilweise können diese Signale vom Benutzer auch deaktiviert werden, so z.B. bei der KDE-Konsole. Der Gebrauch dieser Funktionen ist deshalb nur von eingeschränktem Nutzen.

Kapitel 3

Farben und andere Attribute

Damit *ncurses*-Programme in ihrer ganzen Farbenpracht erstrahlen können, muss im Programmcode die Funktion `int start_color(void);`

initial aufgerufen werden.

3.1 Beispiel

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

void quit(void)
{
  endwin();
}

int main(void)
{
  initscr();
  atexit(quit);
  start_color();
  clear();

  mvaddstr(5, 5, "Hallo");
  mvaddstr(6, 10, "Welt!");
  mvaddstr(15, 1, "Programm beenden durch Drücken einer Taste");
```

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

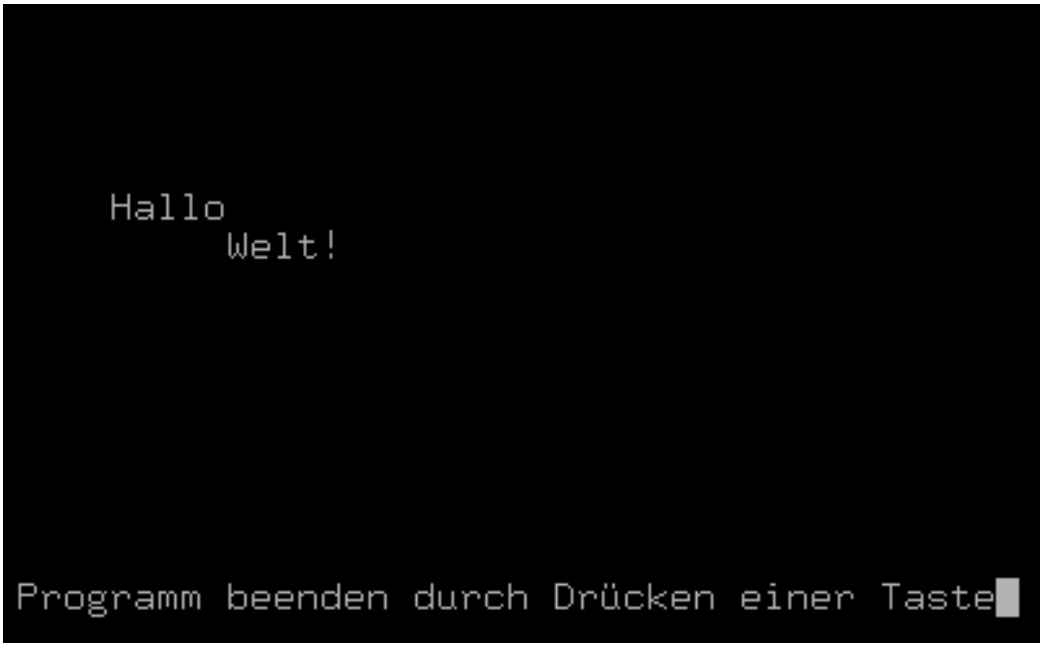

Abbildung 9: Bild:Ncurses2.png

Das Rechteck rechts-unten auf diesem und den folgenden Screenshots stellt übrigens den Cursor dar. Die Anzeige des Cursors kann mit der `curs_set`-Funktion ein-/ausgeschaltet werden.

3.2 Farben einstellen

```
int init_pair(short pair, short f, short b);
int init_color(short color, short r, short g, short b);
```

Parameter:

- `pair`: Paarnummer; `1 <= pair < COLOR_PAIRS`
- `f, b`: foreground-color, background-color; `1 <= (f bzw. b) < COLOR`
- `r, g, b`: ... RGB; `0 <= (r, g bzw. b) <= 1000`

Diese Funktionen sind nur dann sinnvoll einsetzbar, wenn das Terminal Farben, bzw. die Neudefinition von Farbwerten unterstützt. Zwecks Abfrage der Farbfähigkeiten des verwendeten Terminals gibt es die Funktionen `bool has_colors(void);`

und `bool can_change_color(void);`

Erstere liefert die Information, ob das Terminal Farben überhaupt unterstützt. Die zweite Funktion dient der Abfrage, ob der Programmierer die Farbwerte ändern kann.

3.2.1 Basisfarben

Nach der Initalisierung mittels `start_color` sind bei farbfähigen Terminals unmittelbar die acht *ncurses*-Basisfarben verwendbar:

COLOR_BLACK	= 0	♦
COLOR_RED	= 1	♦
COLOR_GREEN	= 2	♦
COLOR_YELLOW	= 3	♦
COLOR_BLUE	= 4	♦
COLOR_MAGENTA	= 5	♦
COLOR_CYAN	= 6	♦
COLOR_WHITE	= 7	♦

3.2.2 Textvorder- und -hintergrundfarbe

```
int color_set(short color_pair_number, void* opts);
```

Farben werden immer paarweise (Vorder-, Hintergrundfarbe) gesetzt (`init_pair`). Der Parameter `opts` ist ein Null-Pointer (0).

Beispiel

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

void quit(void)
{
  endwin();
```

```
}

int main(void)
{
  initscr();
  atexit(quit);
  start_color();
  clear();

  init_pair(1, COLOR_GREEN, COLOR_RED);
  color_set(1, 0);

  mvaddstr(5, 5, "Hallo");
  mvaddstr(6, 10, "Welt!");
  mvaddstr(15, 1, "Programm beenden durch Drücken einer Taste");
  refresh();
  getch();
  return(0);
}
```

Abbildung 10: Bild:Ncurses3.png

3.2.3 Fensterhintergrund

```
int bkgd(chtype ch);
```

Beispiel

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

void quit(void)
{
  endwin();
}

int main(void)
{
  initscr();
  atexit(quit);
  start_color();
  clear();

  init_pair(1, COLOR_GREEN, COLOR_RED);
  bkgd(COLOR_PAIR(1));

  mvaddstr(5, 5, "Hallo");
  mvaddstr(6, 10, "Welt!");
  mvaddstr(15, 1, "Programm beenden durch Drücken einer Taste");
  refresh();
  getch();
  return(0);
}
```

Abbildung 11: Bild:Ncurses4.png

3.3 Zusätzliche Textattribute

Zusätzliche Textattribute lassen sich mit den Funktionen

```
int attrset(int attrs); // setzt Attribute für nachfolgende Texte
int attron(int attrs);  // schaltet zusätzliche Attribute für nachfolgende Texte ein
int attroff(int attrs); // schaltet die angegebenen Attribute wieder aus
int standend(void);     // attrset(0)
int standout(void);     // attrset(A_STANDOUT)
```

einstellen. Einzelattribute lassen sich mittels der OR-Bitoperation (|) verknüpfen. Als Attribute stehen zur Verfügung:

A_NORMAL	normal
A_STANDOUT	Highlight-Modus
A_UNDERLINE	unterstrichen
A_REVERSE	revertiert
A_BLINK	blinkend
A_DIM	gedimmt
A_BOLD	fett
A_PROTECT	geschützt
A_INVIS	unsichtbar
A_ALTCHARSET	alternatives Character-Set

Die genauen Auswirkungen dieser Attribute sind teilweise abhängig von den Fähigkeiten des eingesetzten Terminals.

3.3.1 Beispiel

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

void quit(void)
{
  endwin();
}

int main(void)
{
  initscr();
  atexit(quit);
  start_color();
  clear();

  init_pair(1, COLOR_YELLOW, COLOR_BLUE);
  init_pair(2, COLOR_GREEN, COLOR_BLUE);
  bkgd(COLOR_PAIR(1));

  attrset(A_UNDERLINE);
  mvaddstr(5, 5, "Hallo");
  attrset(A_BOLD);
  mvaddstr(6, 10, "Welt!");
  attrset(A_DIM | COLOR_PAIR(2));
  mvaddstr(15, 1, "Programm beenden durch Drücken einer Taste");

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

KDE-Konsole (Farbschema: Konsole-Standard)

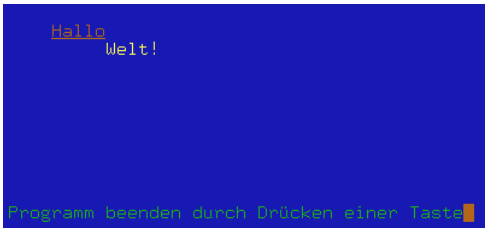

Abbildung 12

KDE-Konsole (Farbschema: XTerm-Farben)

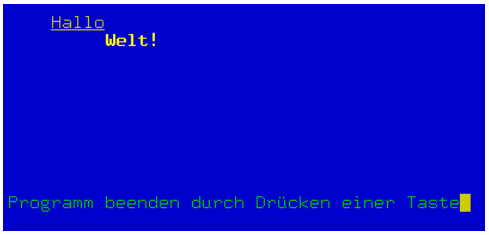

Abbildung 13

rxvt, aterm, xterm

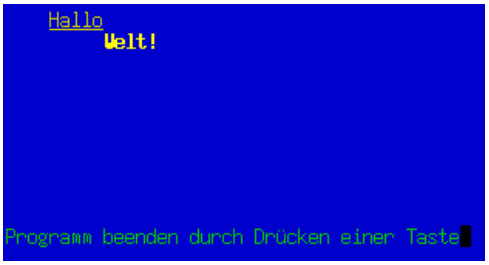

Abbildung 14

Kapitel 4

Spezialtasten

Damit Spezialtasten (z.B. Pfeiltasten, Funktionstasten, Backspace-Taste) korrekt angesprochen werden, müssen mittels `int keypad(WINDOW *win, bool bf);`

deren *ncurses*-Escapesequenzen aktiviert werden.

Nachfolgend die Bezeichnung einiger häufig benötigter Spezialtasten. Zwecks vollständiger Auflistung wird auf die *ncurses*-Header-Datei `curses.h` hingewiesen:

KEY_DOWN	
KEY_UP	
KEY_LEFT	
KEY_RIGHT	
KEY_BACKSPACE	
KEY_F(1), ...	Funktionstaste F1, ...
KEY_END	
KEY_PRINT	

4.1 Beispiel

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

void quit(void)
{
  endwin();
```

```
}

int main(void)
{
  initscr();
  atexit(quit);
  clear();
  noecho();
  curs_set(0);
  cbreak();
  keypad(stdscr, TRUE);

  mvaddstr(5, 5, "Hallo");
  mvaddstr(6, 10, "Welt!");
  mvaddstr(10, 1, "Programm beenden durch Drücken der Taste F1");
  refresh();

  while(getch() != KEY_F(1))
  {
  }

  return(0);
}
```

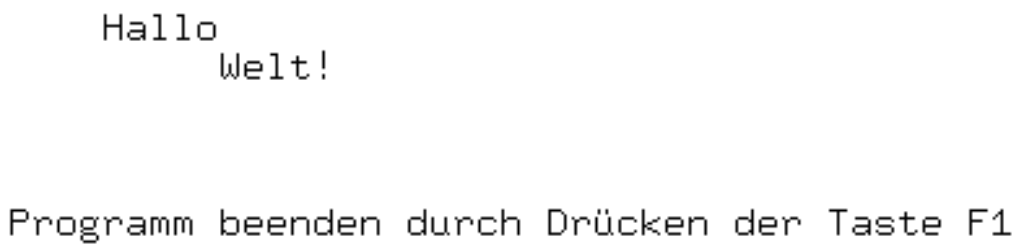

Abbildung 15: Bild:Ncurses60.png

Damit dieses Beispiel ordnungsgemäß funktioniert, muss die Zeilenpufferung und teilweise Einzelzeichenbearbeitung für das Textterminal ausgeschaltet sein. Sicherheitshalber sollte dieser "cbreak"-Modus explizit aktiviert werden, denn es ist nicht garantiert, dass sich das Terminal automatisch in diesem Modus befindet. Testweise kann in diesem Beispiel die `cbreak`-Funktion durch die `nocbreak`-Funktion ersetzt werden. Das Programm lässt sich dann nicht mehr durch Drücken

der F1-Funktionstaste beenden. nocbreak() versetzt das Terminal wieder in den sogenannten ”cooked”-Modus. Zusätzlich kann das Terminal auch noch mittel raw() in den ”raw”-Modus versetzt werden. Dieser unterscheidet sich vom vom ”cbreak”-Modus dadurch, dass die Einzelzeichenbearbeitung komplett ausgeschaltet wird, dem Prozess werden alle Zeichen unverarbeitet zur Verfügung gestellt. Im ”cbreak”-Modus ließe sich das Programm notfalls auch noch durch die Tastenkombination CRTL-C beenden. Im ”raw”-Modus würde dieses Zeichen keinen Interrupt mehr auslösen, sondern wie jedes normale Zeichen an das Programm weitergeleitet werden.

Kapitel 5

Fenster

Bisher wurde bei den Programmen nur der Standardscreen `stdscr` als Fenster verwendet. Mit *ncurses* können aber auch eigene Fenster erzeugt und manipuliert werden. Die Funktionen für Fenster sind sehr ähnlich wie die bisher verwendeten. Allerdings sind die speziellen Fenster-Befehle durch den zusätzlichen Buchstaben `w` markiert, z.B.: `int wrefresh(WINDOW *win);`

```
int wscanw(WINDOW *win, char *fmt, ...);
int mvwaddstr(WINDOW *win, int y, int x, const char *str);
```

Tatsächlich ist es sogar so, dass die bisher verwendeten Standardscreen-Funktionen nur als Makros definiert sind, z.B.: `#define addch(ch) waddch(stdscr,ch)`

```
#define attron(at) wattron(stdscr,at)
#define bkgd(ch)   wbkgd(stdscr,ch)
#define clear()    wclear(stdscr)
```

Zum Erzeugen von Fenstern gibt es mehrere Möglichkeiten:

- Neue Fenster erzeugen: `newwin`
- Abgeleitete Fenster erzeugen: `subwin, derwin`
- Fenster duplizieren: `dupwin`

5.1 Neue Fenster

```
WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x);
```

5.1.1 Beispiel

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

WINDOW *win;

void quit(void)
{
  delwin(win);
  endwin();
}

int main(void)
{

  initscr();
  atexit(quit);
  clear();
  noecho();
  curs_set(0);
  cbreak();
  keypad(stdscr, 1);

  start_color();
  init_pair(1, COLOR_YELLOW, COLOR_BLUE);
  init_pair(2, COLOR_BLUE, COLOR_WHITE);

  win = newwin(5, 20, 10, 10);

  bkgd(COLOR_PAIR(1));
  wbkgd(win, COLOR_PAIR(2));

  mvaddstr(5,5, "Hallo stdscr");
  mvwaddstr(win, 3, 3, "Hallo win");
  mvwaddstr(win, 7, 3, "Diese Zeichenkette wird nicht angezeigt!");
                    // da ausserhalb des win-Anzeigebereichs

  refresh();
  wrefresh(win);
```

```
  while(getch() != KEY_F(1))
  {
  }

  return(0);
}
```

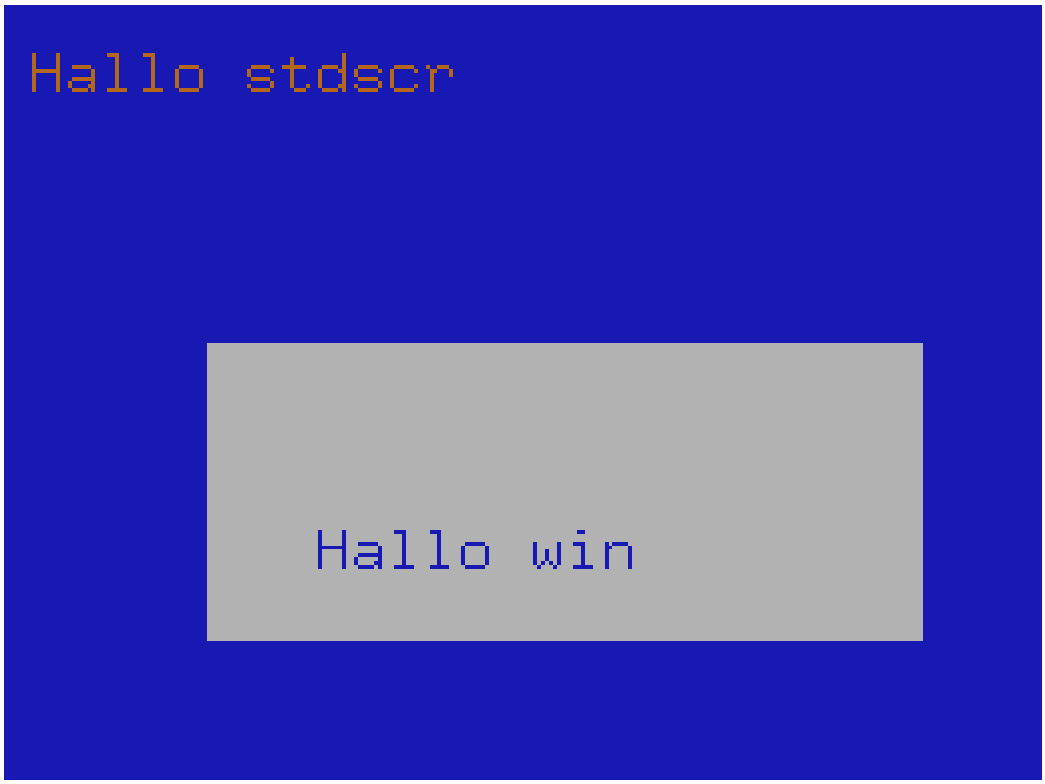

Abbildung 16: Bild:Ncurses8.png

5.2 Abgeleitete Fenster

Ein `subwin` (untergeordnetes Fenster) erbt Eigenschaften vom übergeordneten Fenster.

```
WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x);
```

5.2.1 Beispiel: Aufzeigen des Unterschieds von newwin und subwin

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

WINDOW *win1, *win2;

void quit(void)
{
  delwin(win1);
  delwin(win2);
  endwin();
}

int main(void)
{
  initscr();
  atexit(quit);
  clear();
  noecho();
  curs_set(0);
  cbreak();
  keypad(stdscr, 1);

  start_color();
  init_pair(1, COLOR_YELLOW, COLOR_BLUE);

  win1 = newwin(5, 20, 10, 10);
  win2 = subwin(stdscr, 5, 20, 10, 35);

  bkgd(COLOR_PAIR(1));

  mvaddstr(5,5, "Hallo stdscr");
  mvwaddstr(win1, 3, 3, "Hallo newwin");
  mvwaddstr(win2, 3, 3, "Hallo subwin");

  refresh();
  wrefresh(win1);
  wrefresh(win2);
```

```
  while(getch() != KEY_F(1))
  {
  }

  return(0);
}
```

Abbildung 17: Bild:Ncurses80.png

`derwin` ist im Prinzip das selbe wie `subwin`. Während die Position (`begin_y`, `begin_x`) des Fensters bei der Funktion `subwin` aber relativ zum Screen festgelegt wird, ist die Position bei `derwin` relativ zum `orig`-Fenster.

5.3 Fensterverzierungen

Fenster können auch mit Rahmen oder Begrenzungslinien versehen werden. Diese Verzierungen werden mittels Einzelzeichen aufgebaut. Diese Zeichen können gewöhnliche Buchstaben oder Zahlen sein. Schöner wird das Ganze aber wenn spezielle Zeichen verwendet werden. *ncurses* kennt ”form characters” (ACS, Alternative Character Set), die sich für dieses Aufgabengebiet anbieten. Zu beachten ist, dass Rahmen und Linien wie normaler Text geschrieben werden. Aus diesem Grund können diese Rahmen durch unvorsichtig platzierten Text überschrieben werden. Das bringt zwar keine funktionellen Nachteile, sieht aber nicht schön aus. Dies sollte bei der Verwendung von Rahmen und Linien beachtet werden.

5.3.1 Rahmen

```
int border(chtype ls, chtype rs, chtype ts, chtype bs, chtype tl, chtype tr,
           chtype bl, chtype br);
int wborder(WINDOW *win, chtype ls, chtype rs, chtype ts, chtype bs, chtype tl,
            chtype tr, chtype bl, chtype br);
int box(WINDOW *win, chtype verch, chtype horch);
```

Erklärung der Parameterbezeichnungen:

- ver ... vertikal
- hor ... horizontal
- l ... left
- r ... right
- b ... bottom
- t ... top
- s ... side

Z.B. bedeuten

- `ls` ... left side, Kante links
- `tr` ... top right, Ecke oben-rechts

Wird 0 übergeben, so wird jeweils das in der Bibliothek festgelegte Standardzeichen verwendet.

5.3.2 Linien

```
int hline(chtype ch, int n);
int vline(chtype ch, int n);
int whline(WINDOW *win, chtype ch, int n);
int wvline(WINDOW *win, chtype ch, int n);
int mvhline(int y, int x, chtype ch, int n);
int mvvline(int y, int x, chtype ch, int n);
int mvwhline(WINDOW *, int y, int x, chtype ch, int n);
int mvwvline(WINDOW *, int y, int x, chtype ch, int n);
```

5.3.3 Beispiel

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

WINDOW *win;

void quit(void)
{
  delwin(win);
  endwin();
}

int main(void)
{
  initscr();
  atexit(quit);
  clear();
  noecho();
  curs_set(0);
  cbreak();
  keypad(stdscr, 1);

  start_color();
  init_pair(1, COLOR_YELLOW, COLOR_BLUE);
  init_pair(2, COLOR_BLUE, COLOR_WHITE);

  win = newwin(5, 20, 10, 10);

  bkgd(COLOR_PAIR(1));
  wbkgd(win, COLOR_PAIR(2));

  mvaddstr(5,5, "Hallo stdscr");
  mvwaddstr(win, 3, 3, "Hallo win");
  mvwaddstr(win, 7, 3, "Diese Zeichenkette wird nicht angezeigt!");
                    // da ausserhalb des win-Anzeigebereichs

  box(win, 0, 0);
  mvhline(7, 1, ACS_BULLET, 20);
```

```
    refresh();
    wrefresh(win);

    while(getch() != KEY_F(1))
    {
    }

    return(0);
}
```

Abbildung 18: Bild:Ncurses7.png

5.3.4 ACS (Liniengrafik, form characters)

┌	ACS_ULCORNER
└	ACS_LLCORNER
┐	ACS_URCORNER
┘	ACS_LRCORNER
├	ACS_LTEE

Abbildung 19: Bild:Ncurse_acs_1.png

┤	ACS_RTEE
┴	ACS_BTEE
┬	ACS_TTEE
─	ACS_HLINE
│	ACS_VLINE

Abbildung 20: Bild:Ncurse_acs_2.png

┼	ACS_PLUS
⎺	ACS_S1
⎽	ACS_S9
◆	ACS_DIAMOND
▒	ACS_CKBOARD

Abbildung 21: Bild:Ncurse_acs_3.png

°	ACS_DEGREE
±	ACS_PLMINUS
·	ACS_BULLET
<	ACS_LARROW
>	ACS_RARROW

Abbildung 22: Bild:Ncurse_acs_4.png

⊥	ACS_DARROW
^	ACS_UARROW
#	ACS_BOARD
␋	ACS_LANTERN
#	ACS_BLOCK

Abbildung 23: Bild:Ncurse_acs_5.png

⎻	ACS_S3
⎼	ACS_S7
≤	ACS_LEQUAL
≥	ACS_GEQUAL
π	ACS_PI

Abbildung 24: Bild:Ncurse_acs_6.png

≠	ACS_NEQUAL
£	ACS_STERLING

Abbildung 25: Bild:Ncurse_acs_7.png

Obige ACS-Bilder wurden als Screenshoots dieses Programmes erstellt (KDE-Konsole, xterm)

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

void write_pages(void)
{
```

```
chtype acs_symbol[] = {
  ACS_ULCORNER, ACS_LLCORNER, ACS_URCORNER,
  ACS_LRCORNER, ACS_LTEE,     ACS_RTEE,
  ACS_BTEE,     ACS_TTEE,     ACS_HLINE,
  ACS_VLINE,    ACS_PLUS,     ACS_S1,
  ACS_S9,       ACS_DIAMOND,  ACS_CKBOARD,
  ACS_DEGREE,   ACS_PLMINUS,  ACS_BULLET,
  ACS_LARROW,   ACS_RARROW,   ACS_DARROW,
  ACS_UARROW,   ACS_BOARD,    ACS_LANTERN,
  ACS_BLOCK,    ACS_S3,       ACS_S7,
  ACS_LEQUAL,   ACS_GEQUAL,   ACS_PI,
  ACS_NEQUAL,   ACS_STERLING
};

char acs_name[][20] = {
  "ACS_ULCORNER", "ACS_LLCORNER", "ACS_URCORNER",
  "ACS_LRCORNER", "ACS_LTEE",     "ACS_RTEE",
  "ACS_BTEE",     "ACS_TTEE",     "ACS_HLINE",
  "ACS_VLINE",    "ACS_PLUS",     "ACS_S1",
  "ACS_S9",       "ACS_DIAMOND",  "ACS_CKBOARD",
  "ACS_DEGREE",   "ACS_PLMINUS",  "ACS_BULLET",
  "ACS_LARROW",   "ACS_RARROW",   "ACS_DARROW",
  "ACS_UARROW",   "ACS_BOARD",    "ACS_LANTERN",
  "ACS_BLOCK",    "ACS_S3",       "ACS_S7",
  "ACS_LEQUAL",   "ACS_GEQUAL",   "ACS_PI",
  "ACS_NEQUAL",   "ACS_STERLING"
};

int rows = 5, page=0, i, j, flag=0;
int acs_nr = sizeof(acs_symbol) / sizeof(chtype);


for(j=0; j<=acs_nr/rows; j++)
{
  clear ();

  for(i=0; i<rows; i++)
  {
    if(page*rows+i <= acs_nr-1)
    {
      mvaddch(i*2+1, 3, acs_symbol[page*rows + i]);
```

```
        mvaddstr(i*2+1, 8, acs_name[page*rows + i]);
      }
      else
      {
        flag=1;
      }
    }

    if(!flag)
    {
      mvaddstr(rows*2 +2, 1, "Taste drücken -> nächste Seite");
    }
    else
    {
      mvaddstr(rows*2 +2, 1, "Taste drücken -> Ende");
    }

    refresh();
    page++;
    getch();
  }
}

void quit(void)
{
  endwin();
}

int main(void)
{
  initscr();
  atexit(quit);
  noecho();
  curs_set(0);

  write_pages();

  return(0);
}
```

5.4 Fenster löschen

Der durch ein Fenster belegte Speicherplatz kann über die Funktion

```
int delwin(WINDOW *win);
```

wieder freigegeben werden. Auf die Bildschirmdarstellung hat das vorerst keinen Einfluss. Natürlich sollte danach nicht mehr auf das gelöschte Fenster zugegriffen werden, da dies in aller Regel einen Programmabsturz infolge ”Speicherzugriffsfehler” auslöst.

5.5 Fenster refreshen

Zum Refreshen eines Fensters sind diese Funktionen vorgesehen:

```
int wrefresh(WINDOW *win);
int wnoutrefresh(WINDOW *win);
int doupdate(void);
```

Welche Funktion soll wann Verwendung finden?

Die einfachste Möglichkeit ist der Aufruf von `wrefresh`. Diese Funktion bringt den gewünschten Fensterinhalt auf den real existierenden Bildschirm. `wrefresh` besteht im Prinzip aus der sequentiellen Abfolge der Funktionen:

1. `wnoutrefresh` ... kopiert den gewünschten Fensterinhalt in den virtuellen Bildschirmspeicher.
2. `doupdate` ... gleicht virtuellen Bildschirmspeicher mit dem realen Bildschirminhalt ab und vollzieht das Update.

Sind viele Fenster gleichzeitig zu refreshen, dann ist die `wrefresh`-Funktion ineffizient. In diesem Fall ist es besser, zuerst alle Fenster mit einem `wnoutrefresh` zu aktualisieren und am Ende nur einmal die `doupdate`-Funktion aufzurufen.

5.6 Touch und Untouch

Wird der Fensterinhalt geändert, dann wird das Fenster automatisch als ”touched” (berührt) markiert. Die refresh-Funktion erkennt daran, dass das Fenster aktualisiert werden muss. Als ”untouched” markierte Fenster werden bei Refreshs nicht

aktualisiert, da aus Performancegründen virtueller und physikalischer Screen abgeglichen und nur die Änderungen übertragen werden. Ein Fenster kann auch manuell wieder als ”untouched” markiert werden.

```
int touchwin(WINDOW *win);
int untouchwin(WINDOW *win);
bool is_wintouched(WINDOW *win);
```

5.6.1 Beispiel

// ... mvwaddstr(win, 4, 3, ”Beenden -> F1 !”); wrefresh(win); // ...

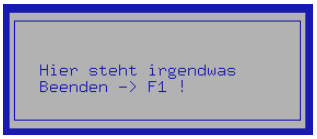

Abbildung 26: Bild:Ncurses81.png

Die Zeichenkette wird wie erwartet in das Fenster eingefügt

// ... mvwaddstr(win, 4, 3, ”Beenden -> F1 !”); untouchwin(win); wrefresh(win); // ...

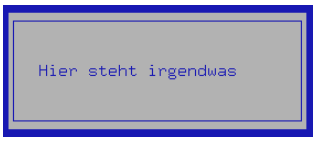

Abbildung 27: Bild:Ncurses82.png

`untouchwin` - der nachfolgende Refresh bewirkt keine Änderung des ursprünglichen Fensterinhaltes

// ... mvwaddstr(win, 4, 3, ”Beenden -> F1 !”); untouchwin(win); wrefresh(win); touchwin(win); wrefresh(win); // ...

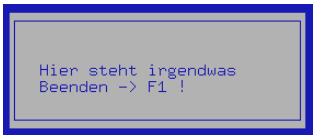

Abbildung 28: Bild:Ncurses81.png

Durch Einfügung der `touchwin`-Anweisung wird dieses Fenster bei einem Refresh wieder aktualisiert

Kapitel 6

Panels

Ncurses-Fenster sind gut verwendbar, solange sie nebeneinander platziert sind. Überlappen sich Fenster jedoch, dann kann es kompliziert werden. Aus diesem Grund kennt *ncurses* sogenannte Panels. Diese bieten Mechanismen, Fenster auch in z-Richtung zu verwalten. Panels gehören zwar zu *ncurses*, sind aber in eine eigene Bibliotheksdatei ausgelagert.

Ein neues Panel erzeugen: `PANEL *new_panel(WINDOW *win);`

Reservierten Panel- Speicherplatz freigeben: `int del_panel(PANEL *pan);`

Panelanzeige manipulieren: `int bottom_panel(PANEL *pan); // Panel nach hinten verschieben`

```
int top_panel(PANEL *pan);    // Panel in den Vordergrund holen
int show_panel(PANEL *pan);   // Panel anzeigen
int hide_panel(PANEL *pan);   // Panel verstecken
```

Vorne-, hintenliegendes Panel eruieren (`pan=0`: top bzw. bottom): `PANEL *panel_above(const PANEL *pan)`

```
PANEL *panel_below(const PANEL *pan)
```

Panels updaten: `void update_panels();`

6.1 Beispiel

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

WINDOW *win1, *win2;
```

```
PANEL  *pan1, *pan2;

void quit(void)
{
  del_panel(pan1);
  del_panel(pan2);
  delwin(win1);
  delwin(win2);
  endwin();
}

int main(void)
{
  int flag=0;

  initscr();
  atexit(quit);
  clear();
  noecho();
  curs_set(0);
  cbreak();
  keypad(stdscr, 1);

  start_color();
  init_pair(1, COLOR_YELLOW, COLOR_BLUE);
  init_pair(2, COLOR_BLACK, COLOR_WHITE);
  init_pair(3, COLOR_BLACK, COLOR_YELLOW);

  win1 = newwin(10, 25, 5, 10);
  win2 = newwin(10, 25, 10, 15);
  box(win1, ACS_VLINE, ACS_HLINE);
  box(win2, ACS_VLINE, ACS_HLINE);
  pan1 = new_panel(win1);
  pan2 = new_panel(win2);

  bkgd(COLOR_PAIR(1));
  wbkgd(win1, COLOR_PAIR(2));
  wbkgd(win2, COLOR_PAIR(3));

  mvaddstr(2,4, "F1 beendet das Programm");
  mvwaddstr(win1, 2, 3, "Drücke eine Taste");
```

```
  mvwaddstr(win2, 7, 3, "Drücke eine Taste");

  update_panels();
  doupdate();

  while(getch() != KEY_F(1))
  {
    if (flag==0)
    {
      top_panel(pan1);
      flag = 1;
    }
    else
    {
      top_panel(pan2);
      flag = 0;
    }

    update_panels();
    doupdate();
  }

  return (0);
}
```

Compilieren, Linken:

```
gcc -o bsp bsp.c -lpanel -lncurses
```

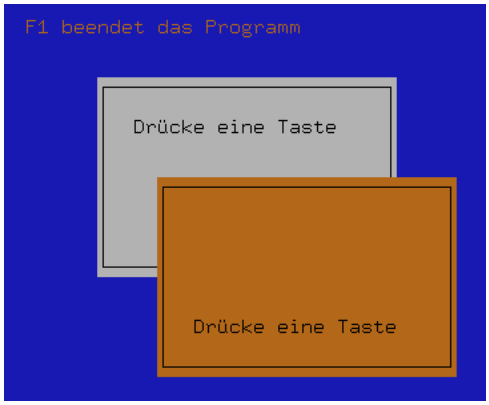

Abbildung 29

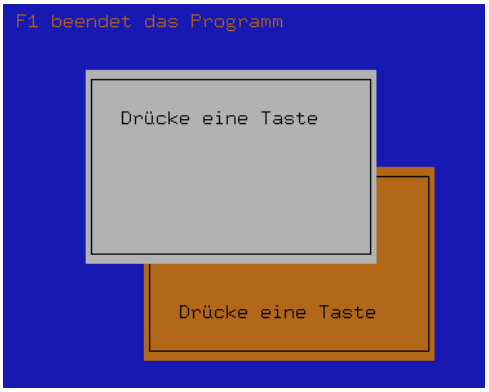

Abbildung 30

Kapitel 7

Menüs

Ein *ncurses*-Menü ist ein Menü in der ursprünglichen Bedeutung, also eine einfache Auswahlbox. Allerdings bietet auch die *menu*-Bibliothek genügend Möglichkeiten um diese Auswahlboxen sehr schön und sinnvoll auszugestalten.

7.1 Ein Menü erzeugen und wieder löschen

Die Erzeugung eines Menüs untergliedert sich in folgende Schritte:

- Speicherplatz für die Items (Menüeinträge) reservieren: `calloc`-Funktion
- Items erzeugen: `ITEM *new_item(const char *name, const char *description);`
- Menü erzeugen: `MENU *new_menu(ITEM **items);`
- Menü ”posten” (post = anheften, ankleben -> ”PostIt”) : `int post_menu(MENU *menu);`

Das ”Abbauen” eines Menüs geschieht in umgekehrter Reihenfolge:

- Menü ”unposten”: `int unpost_menu(MENU *menu);`
- Menü freigeben: `int free_menu(MENU *menu);`
- Items freigeben: `int free_item(ITEM *item);`
- Reservierten Items-Speicherplatz freigeben: `free`-Funktion

7.2 Der Menü-Treiber

Eingabeereignisse für das Menü können mit der Funktion `int menu_driver(MENU *menu, int c);`

abgehandelt werden.

Der Parameter `c` bstimmt, welche Menüaktion durchgeführt werden soll:

REQ_LEFT_ITEM	bewegt den Menücursor um einen Eintrag nach links
REQ_RIGHT_ITEM	bewegt den Menücursor um einen Eintrag nach rechts
REQ_UP_ITEM	bewegt den Menücursor um einen Eintrag nach oben
REQ_DOWN_ITEM	bewegt den Menücursor um einen Eintrag nach unten
REQ_SCR_ULINE	eine Zeile aufwärts scrollen
REQ_SCR_DLINE	eine Zeile abwärts scrollen
REQ_SCR_UPAGE	eine Seite aufwärts scrollen
REQ_SCR_DPAGE	eine Seite abwärts scrollen
REQ_FIRST_ITEM	bewegt den Menücursor zum ersten Eintrag
REQ_LAST_ITEM	bewegt den Menücursor zum letzten Eintrag
REQ_NEXT_ITEM	bewegt den Menücursor zum nächsten Eintrag
REQ_PREV_ITEM	bewegt den Menücursor zum vorherigen Eintrag
REQ_TOGGLE_ITEM	An- oder abwählen eines Eintrags
REQ_CLEAR_PATTERN	Suchmusterpuffer löschen
REQ_BACK_PATTERN	Das vorherige Zeichen aus dem Suchmusterpuffer löschen
REQ_NEXT_MATCH	Menücursor zum nächsten Eintrag, der zum Suchmuster passt, bewegen
REQ_PREV_MATCH	Menücursor zum vorigen Eintrag, der zum Suchmuster passt, bewegen

7.3 Den aktuell angewählten Menüeintrag ermitteln

```
ITEM *current_item(const MENU *menu);
int item_index(const ITEM *item);
```

7.3.1 Beispiel

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

ITEM **it;
MENU *me;

void quit(void)
{
  int i;

  unpost_menu(me);
  free_menu(me);

  for(i=0; i<=4; i++)
  {
    free_item(it[i]);
  }

  free(it);
  endwin();
}

int main(void)
{
  int ch;

  initscr();
  atexit(quit);
  clear();
  noecho();
  curs_set(0);
  cbreak();
```

```
  nl();
  keypad(stdscr, TRUE);

  it = (ITEM **)calloc(5, sizeof(ITEM *));
  it[0] = new_item("M1", "");
  it[1] = new_item("M2", "");
  it[2] = new_item("M3", "");
  it[3] = new_item("Ende", "");
  it[4] = 0;
  me = new_menu(it);
  post_menu(me);

  mvaddstr(7, 3, "Programm mittels Menü oder F1-Funktionstaste beenden");
  refresh();

  while((ch=getch()) != KEY_F(1))
  {
    switch(ch)
    {
      case KEY_DOWN:
        menu_driver(me, REQ_DOWN_ITEM);
        break;
      case KEY_UP:
        menu_driver(me, REQ_UP_ITEM);
        break;
      case 0xA: /* Return- bzw. Enter-Taste -> ASCII-Code */
        if(item_index(current_item(me)) == 3)
          exit(0);
    }
  }

  return (0);
}
```

Compilieren, Linken: `gcc -o bsp bsp.c -lmenu -lncurses`

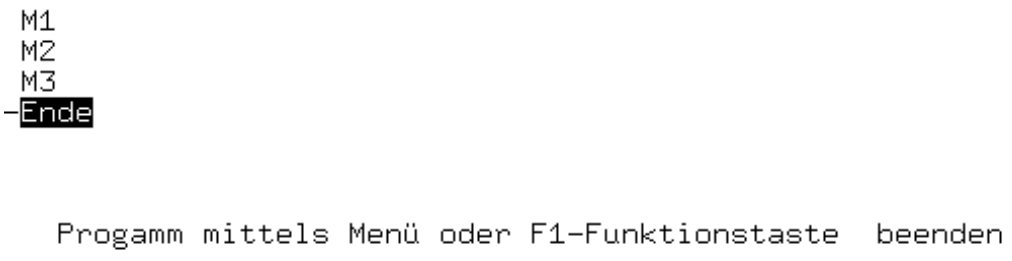

Abbildung 31: Bild:Ncurses10.png

7.4 Das Menü formatieren

Ein Menü ist standardmäßig bis zu 16 Zeilen hoch und 1 Spalte breit. Zur Einstellung einer anderen Größe, z.B. zur Generierung eines mehrspaltigen Menüs, existiert die Funktion

```
int set_menu_format(MENU *menu, int rows, int cols);
```

Diese Funktion muss im Bedarfsfall aufgerufen werden, bevor das Menu gepostet wird.

7.4.1 Beispiel

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

ITEM **it;
MENU *me;

void quit(void)
{
  int i;

  unpost_menu(me);
  free_menu(me);

  for(i=0; i<=4; i++)
  {
```

```
    free_item(it[i]);
  }

  free(it);
  endwin();
}

int main(void)
{
  int ch;

  initscr();
  atexit(quit);
  clear();
  noecho();
  curs_set(0);
  cbreak();
  nl();
  keypad(stdscr, TRUE);

  it = (ITEM **)calloc(5, sizeof(ITEM *));
  it[0] = new_item("M1", "");
  it[1] = new_item("M2", "");
  it[2] = new_item("M3", "");
  it[3] = new_item("Ende", "");
  it[4] = 0;
  me = new_menu(it);
  set_menu_format(me, 2, 2);
  post_menu(me);

  mvaddstr(7, 3, "Programm mittels Menü oder F1-Funktionstaste beenden");
  refresh();

  while((ch=getch()) != KEY_F(1))
  {
    switch(ch)
    {
      case KEY_DOWN:
        menu_driver(me, REQ_DOWN_ITEM);
        break;
      case KEY_UP:
```

```
        menu_driver(me, REQ_UP_ITEM);
        break;
      case KEY_RIGHT:
        menu_driver(me, REQ_RIGHT_ITEM);
        break;
      case KEY_LEFT:
        menu_driver(me, REQ_LEFT_ITEM);
        break;
      case 0xA: /* Return- bzw. Enter-Taste -> ASCII-Code */
        if(item_index(current_item(me)) == 3)
          exit(0);
    }
  }

  return (0);
}
```

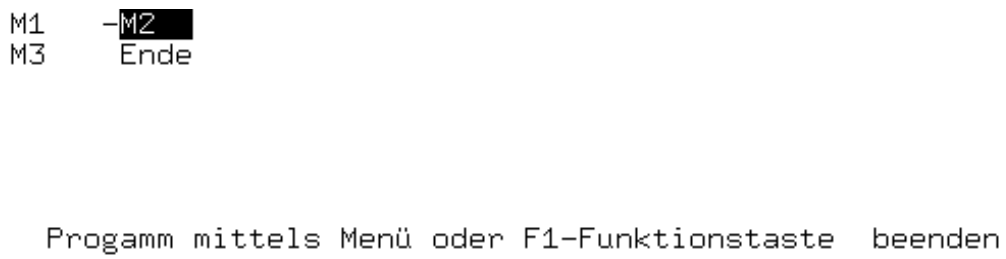

Abbildung 32: Bild:Ncurses11.png

7.5 Das Markierungssymbol

Zwecks besserer Sichtbarkeit wird vor dem ausgewählten Menüeintrag zusätzlich ein Zeichen oder eine Zeichenkette als Markierungssymbol gesetzt. Normalerweise ist dies ein Bindestrich. Dieses Verhalten kann aber mit der Funktion `int set_menu_mark(MENU *menu, const char *mark);`

geändert werden. Diese Funktion wird nur dann das gewünschte Resultat liefern, wenn sie vor dem Posten des Menüs aufgerufen wird.

7.5.1 Beispiel (Programmausschnitt)

```
// ...
set_menu_mark(me, "->");
post_menu(me);
// ...
```

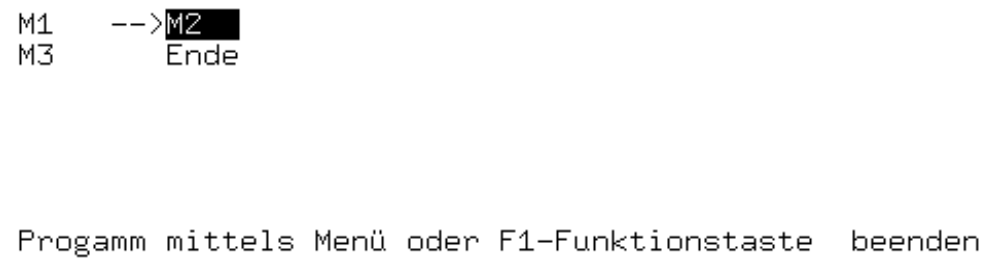

Abbildung 33: Bild:Ncurses12.png

7.5.2 Beispiel (Programmausschnitt)

```
// Ausschalten des Markierungssysmbols
//...
set_menu_mark(me, "");
post_menu(me);
// ...
```

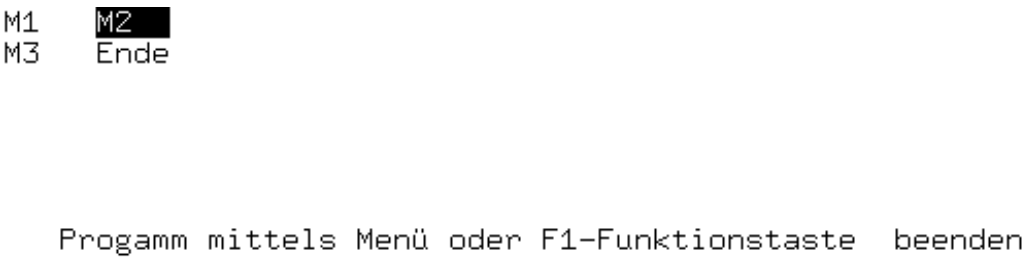

Abbildung 34: Bild:Ncurses13.png

Das Markierungssymbol sollte aber nur dann ausgeschaltet werden, wenn absolut sichergestellt ist, dass das Programm nur auf Terminals mit "Highlighting"- oder

Farbunterstützung eingesetzt wird. Das ”Erraten” der jeweiligen Menücursorposition kann sich sonst sehr nervenaufreibend gestalten.

7.6 Menüfenster

Jedes Menü kann mit einem Haupt- und Unterfenster verknüpft werden. Das Hauptfenster kann z.B. zur Aufnahme eines Menütitels und zur Umrahmung des Unterfensters dienen. Im Unterfenster werden die Menüeinträge dargestellt.

```
int set_menu_win(MENU *menu, WINDOW *win);
int set_menu_sub(MENU *menu, WINDOW *sub);
```

Werden diese Funktionen in einem Programm nicht eingesetzt, so sind die Menüs mit dem Standardscreen verbunden.

7.6.1 Beispiel

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

ITEM   **it;
MENU   *me;
WINDOW *win;

void quit(void)
{
  int i;

  unpost_menu(me);
  free_menu(me);

  for(i=0; i<=4; i++)
  {
    free_item(it[i]);
  }

  free(it);
  delwin(win);
  endwin();
```

```
}

int main(void)
{
  int ch;

  initscr();
  atexit(quit);
  clear();
  noecho();
  curs_set(0);
  cbreak();
  nl();
  keypad(stdscr, TRUE);

  it = (ITEM **)calloc(5, sizeof(ITEM *));
  it[0] = new_item("M1", "Menueeintrag 1");
  it[1] = new_item("M2", "Menueeintrag 2");
  it[2] = new_item("M3", "Menueeintrag 3");
  it[3] = new_item("Ende", "Programm beenden");
  it[4] = 0;
  me = new_menu(it);

  win = newwin(8, 30, 5, 5);
  set_menu_win (me, win);
  set_menu_sub (me, derwin(win, 4, 28, 3, 2));
  box(win, 0, 0);
  mvwaddstr(win, 1, 2, "***** Testmenü *****");
  post_menu(me);

  mvaddstr(14, 3, "Programm mittels Menü oder F1-Funktionstaste beenden");

  refresh();
  wrefresh(win);

  while((ch=getch()) != KEY_F(1))
  {
    switch(ch)
    {
      case KEY_DOWN:
        menu_driver(me, REQ_DOWN_ITEM);
```

```
        break;
      case KEY_UP:
        menu_driver(me, REQ_UP_ITEM);
        break;
      case 0xA: /* Return- bzw. Enter-Taste -> ASCII-Code */
        if(item_index(current_item(me)) == 3)
          exit(0);
    }

    wrefresh(win);
  }

  return (0);
}
```

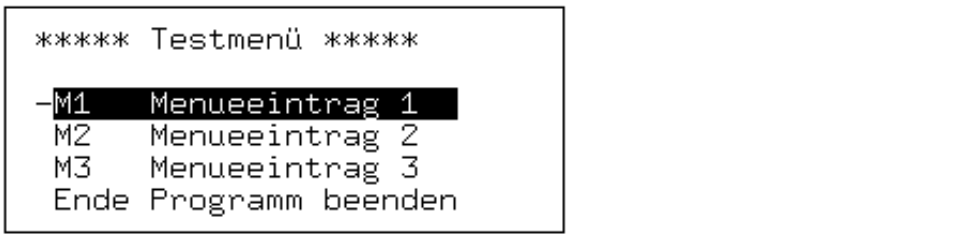

Abbildung 35: Bild:Ncurses14.png

Die minimal notwendige Größe eines Unterfensters kann über die Funktion

```
int scale_menu(const MENU *menu, int *rows, int *columns);
```

ermittelt werden.

7.7 Menüs bunt gestalten

Die Farbgebung und Darstellungsattribute des selektierten Items festlegen:

```
int set_menu_fore(MENU *menu, chtype attr);
```

Die Farbgebung und Darstellungsattribute der unselektierten Items festlegen:

```
int set_menu_back(MENU *menu, chtype attr);
```

Das Erscheinungsbild des Menühauptfensters kann natürlich über die konventionellen *ncurses*-Befehle (`wbkgd, wattrset`, etc.) gestaltet werden.

7.7.1 Beispiel

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

ITEM   **it;
MENU   *me;
WINDOW *win;

void quit(void)
{
  int i;

  unpost_menu(me);
  free_menu(me);

  for(i=0; i<=4; i++)
  {
    free_item(it[i]);
  }

  free(it);
  delwin(win);

  endwin();
}

int main(void)
{
  int ch;

  initscr();
  atexit(quit);
  clear();
  noecho();
  curs_set(0);
  cbreak();
```

```
nl();
keypad(stdscr, TRUE);
start_color();

init_pair(1, COLOR_WHITE, COLOR_BLUE);
init_pair(2, COLOR_BLUE, COLOR_YELLOW);

bkgd(COLOR_PAIR(1));

it = (ITEM **)calloc(5, sizeof(ITEM *));
it[0] = new_item("M1", "Menueeintrag 1");
it[1] = new_item("M2", "Menueeintrag 2");
it[2] = new_item("M3", "Menueeintrag 3");
it[3] = new_item("Ende", "Programm beenden");
it[4] = 0;
me = new_menu(it);

win = newwin(8, 30, 5, 5);
set_menu_win (me, win);
set_menu_sub (me, derwin(win, 4, 28, 3, 2));
box(win, 0, 0);
mvwaddstr(win, 1, 2, "***** Testmenü *****");
set_menu_fore(me, COLOR_PAIR(1)|A_REVERSE);
set_menu_back(me, COLOR_PAIR(1));
wbkgd(win, COLOR_PAIR(2));

post_menu(me);

mvaddstr(14, 3, "Programm mittels Menü oder F1-Funktionstaste beenden");

refresh();
wrefresh(win);

while((ch=getch()) != KEY_F(1))
{
  switch(ch)
  {
    case KEY_DOWN:
      menu_driver(me, REQ_DOWN_ITEM);
      break;
    case KEY_UP:
```

```
        menu_driver(me, REQ_UP_ITEM);
        break;
      case 0xA: /* Return- bzw. Enter-Taste -> ASCII-Code */
        if(item_index(current_item(me)) == 3)
          exit(0);
    }

    wrefresh(win);
  }

  return (0);
}
```

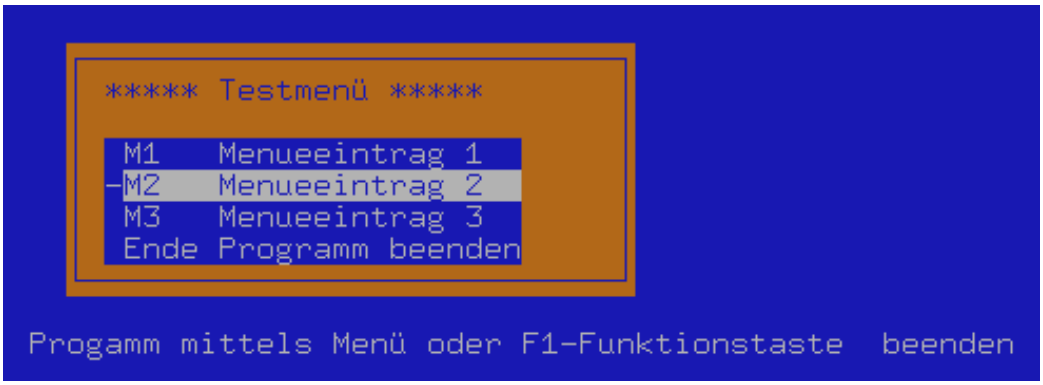

Abbildung 36: Bild:Ncurses15.png

7.8 Optionen für Menüeinträge

```
int set_item_opts(ITEM *item, OPTIONS opts);
int item_opts_on(ITEM *item, OPTIONS opts);
int item_opts_off(ITEM *item, OPTIONS opts);
```

Die Funktionsbezeichnungen sind selbsterklärend. Es gibt hier nur eine Option, nämlich `O_SELECTABLE`

Wird die Selektierbarkeit für einen Menüeintrag hiermit ausgeschaltet, so ist dieser Menüeintrag als nicht selektierbar dargestellt. Dies bezieht sich aber nur auf die Menüdarstellung, das Item ist trotzdem immer noch auswählbar. Die ”Nichtselektierbarkeit” eines Items muss vom Programmierer im weiteren Code berück-

sichtigt werden. Möglich ist dies durch die Abfrage der Itemoptionen `OPTIONS item_opts(const ITEM *item);`

Die Farbgebung für derartige nicht selektierbare Items kann über die Funktion `int set_menu_grey(MENU *menu, chtype attr);`

gesteuert werden.

7.8.1 Beispiel

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

ITEM   **it;
MENU   *me;
WINDOW *win;

void quit(void)
{
  int i;

  unpost_menu(me);
  free_menu(me);

  for(i=0; i<=4; i++)
  {
    free_item(it[i]);
  }

  free(it);
  delwin(win);

  endwin();
}

int main(void)
{
  int ch;

  initscr();
  atexit(quit);
```

```
clear();
noecho();
curs_set(0);
cbreak();
nl();
keypad(stdscr, TRUE);
start_color();

init_pair(1, COLOR_WHITE, COLOR_BLUE);
init_pair(2, COLOR_BLUE, COLOR_YELLOW);
init_pair(3, COLOR_BLACK, COLOR_BLUE);

bkgd(COLOR_PAIR(1));

it = (ITEM **)calloc(5, sizeof(ITEM *));
it[0] = new_item("M1", "Menueeintrag 1");
it[1] = new_item("M2", "Menueeintrag 2");
it[2] = new_item("M3", "Menueeintrag 3");
it[3] = new_item("Ende", "Programm beenden");
it[4] = 0;
item_opts_off(it[3], O_SELECTABLE);
me = new_menu(it);

win = newwin(8, 30, 5, 5);
set_menu_win (me, win);
set_menu_sub (me, derwin(win, 4, 28, 3, 2));
box(win, 0, 0);
mvwaddstr(win, 1, 2, "***** Testmenü *****");
set_menu_fore(me, COLOR_PAIR(1)|A_REVERSE);
set_menu_back(me, COLOR_PAIR(1));
set_menu_grey(me, COLOR_PAIR(3));
wbkgd(win, COLOR_PAIR(2));
post_menu(me);

mvaddstr(14, 3, "Programm mittels F1-Funktionstaste beenden");

refresh();
wrefresh(win);

while((ch=getch()) != KEY_F(1))
{
```

```
    switch(ch)
    {
      case KEY_DOWN:
        menu_driver(me, REQ_DOWN_ITEM);
        break;
      case KEY_UP:
        menu_driver(me, REQ_UP_ITEM);
        break;
      case 0xA: /* Return- bzw. Enter-Taste -> ASCII-Code */
        if(item_index(current_item(me))==3 &&
           item_opts(current_item(me))==O_SELECTABLE)
        exit(0);
    }

    wrefresh(win);
  }

  return (0);
}
```

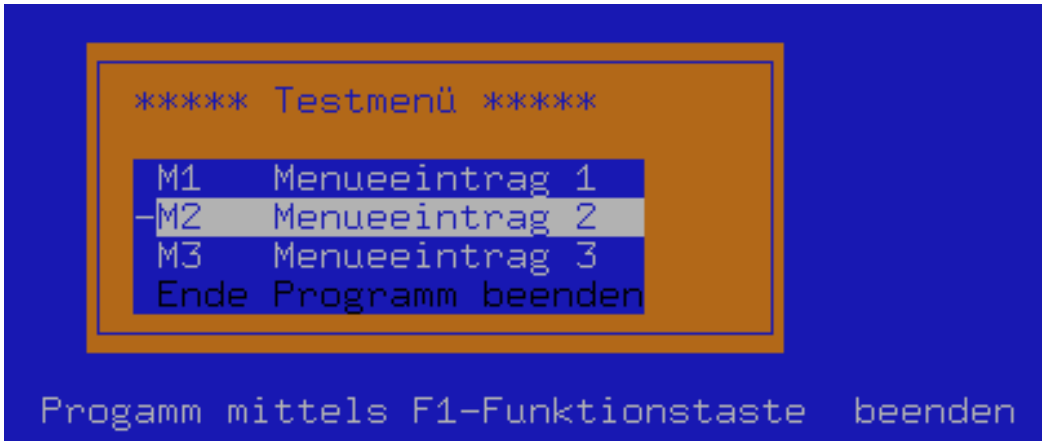

Abbildung 37: Bild:Ncurses16.png

7.9 Menüoptionen

Zum Setzen von Menüoptionen werden diese Funktionen verwendet: `int set_menu_opts(MENU *menu, OPTIONS opts);`

```
int menu_opts_on(MENU *menu, OPTIONS opts);
int menu_opts_off(MENU *menu, OPTIONS opts);
```

Einige der verfügbaren Optionswerte sind

`O_SHOWDESC`	Beschreibungen zu den Einträgen anzeigen
`O_NONCYCLIC`	Nicht-zyklisch, am Menüende nicht automatisch zum Menübeginn springen und umgekehrt
`O_ONEVALUE`	Nur ein Menüeintrag ist auswählbar

Alle Optionen sind standardmäßig eingeschaltet.

7.9.1 Beispiel: O_SHOWDESC

```
// ...
menu_opts_off(me, O_SHOWDESC);
// ...
```

Abbildung 38: Bild:Ncurses130.png

7.9.2 Beispiel: Mehrfachauswahl

Bei einer möglichen Mehrfachauswahl kann nicht mehr einfach mittels `current_item` der ausgewählte Menüeintrag bestimmt werden. Stattdessen kann die Funktion

```
bool item_value(const ITEM *item);
```

Verwendung finden. Diese Funktion liefert für selektierte Menüeinträge `TRUE`, für unselektierte Menüeinträge `FALSE`.

```
// ...
menu_opts_off(me, O_ONEVALUE);
// ...

while((ch=getch()) != KEY_F(1))
{
  switch(ch)
  {
    // ...
    case 0x20: /* Leertaste */
      menu_driver(me, REQ_TOGGLE_ITEM);

      int i;

      for(i=0; i<4; i++)
      {
        mvprintw(16+i, 3, "Item %i: %i", i+1, item_value(it[i]));
      }
      break;
    // ...
  }
}
//...
```

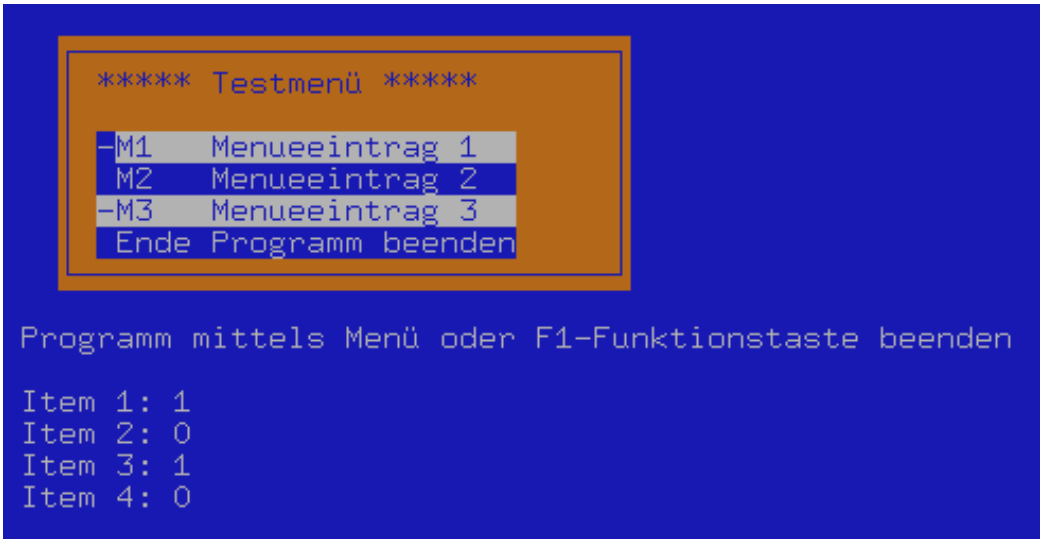

Abbildung 39: Bild:Ncurses132.png

Damit sind die *ncurses*-Menüs zwar bei weitem noch nicht umfassend abgehandelt. Jedoch wären weitere Themen teilweise schon sehr komplex und nur für spezielle Einsatzfälle wirklich interessant. Mit dieser Bemerkung soll dieses Kapitel hier vorerst abgeschlossen werden.

Kapitel 8

Formulare

8.1 Ein Formular erzeugen und wieder löschen

Der Auf- und Abbau von Formularen folgt dem gleichen Prinzip wie bei Menüs.

Die Erzeugung eines Formulars untergliedert sich in folgende Schritte:

- Speicherplatz für die Formularfelder reservieren: z.B. mittels `calloc`-Funktion
- Formularfelder erzeugen:
  - `FIELD *new_field(int height, int width, int toprow, int leftcol, int offscreen, int nbuffers);`
  - `FIELD *dup_field(FIELD *field, int toprow, int leftcol);`
  - `FIELD *link_field(FIELD *field, int toprow, int leftcol);`

  Formular erzeugen: `FORM *new_form(FIELD **fields);`
- Formular "posten": `int post_form(FORM *form);`

Zu beachten ist, dass das letzte Formularfeld zwingend ein Null-Pointer sein muss.

Das "Abbauen" eines Formulars geschieht in umgekehrter Reihenfolge:

- Formular "unposten": `int unpost_form(FORM *form);`
- Formular freigeben: `int free_form(FORM *form);`

- Felder freigeben: `int free_field(FIELD *field);`
- Reservierten Feld-Speicherplatz freigeben: `free`-Funktion

8.2 Der Formulartreiber

Eingabeereignisse für ein Formular werden durch den Formulartreiber abgearbeitet.

```
int form_driver(FORM *form, int c);
```

Welche Aktion konkret ausgeführt werden soll, wird durch den Parameter `c` bestimmt. Eine schiere Unzahl von Optionen ist verfügbar. Nachfolgend werden nur ein paar dieser Request-Optionen aufgelistet:

REQ_NEXT_FIELD	Cursor zum nächsten Feld bewegen
REQ_PREV_FIELD	Cursor zum vorherigen Feld bewegen
REQ_FIRST_FIELD	Cursor zum ersten Feld bewegen
REQ_LAST_FIELD	Cursor zum letzten Feld bewegen
REQ_BEG_LINE	Cursor zum Zeilenanfang bewegen
REQ_END_LINE	Cursor zum Zeilenende bewegen
REQ_LEFT_CHAR	Cursor im Feld nach links bewegen
REQ_RIGHT_CHAR	Cursor im Feld nach rechts bewegen
REQ_UP_CHAR	Cursor im Feld nach oben bewegen
REQ_DOWN_CHAR	Cursor im Feld nach unten bewegen
REQ_INS_CHAR	An der Cursorposition ein Leerzeichen einfügen
REQ_DEL_CHAR	An der Cursorposition ein Zeichen löschen
REQ_DEL_PREV	Das Zeichen vor der Cursorposition löschen
REQ_CLR_FIELD	Das ganze Formularfeld löschen
REQ_OVL_MODE	Überschreibmodus aktivieren
REQ_INS_MODE	Einfügemodus aktivieren

8.3 Feldfarben und andere Darstellungsattribute

Hintergrundattribute festlegen: `int set_field_fore(FIELD *field, chtype attr);`

Vordergrundattribute festlegen: `int set_field_back(FIELD *field, chtype attr);`

8.3.1 Beispiel

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

FIELD  **fi;
FORM   *fo;

void quit(void)
{
  int i;

  unpost_form(fo);
  free_form(fo);

  for(i=0; i<=3; i++)
  {
    free_field(fi[i]);
  }

  free(fi);
  endwin();
}

int main(void)
{
  int ch, i;

  initscr();
  atexit(quit);
  clear();
  noecho();
  cbreak();
  keypad(stdscr, TRUE);
  start_color();

  init_pair(1, COLOR_YELLOW, COLOR_BLUE);
  init_pair(2, COLOR_BLACK, COLOR_WHITE);
```

```
  bkgd(COLOR_PAIR(1));

  fi = (FIELD **)calloc(4, sizeof(FIELD *));
  fi[0] = new_field(1, 10, 2, 3, 0, 0);
  fi[1] = new_field(1, 10, 2, 18, 0, 0);
  fi[2] = new_field(1, 15, 2, 33, 0, 0);
  fi[3] = 0;

  for(i=0; i<3; i++)
  {
    set_field_fore(fi[i], COLOR_PAIR(2));
    set_field_back(fi[i], COLOR_PAIR(2));
  }

  fo = new_form(fi);
  post_form(fo);

  mvaddstr(2, 15, "+");
  mvaddstr(2, 30, "=");
  mvaddstr(5, 3, "Programm mittels F1-Funktionstaste beenden");

  refresh();

  while((ch=getch()) != KEY_F(1))
  {
    switch(ch)
    {
      case KEY_RIGHT:
        form_driver(fo, REQ_NEXT_FIELD);
        break;
      case KEY_LEFT:
        form_driver(fo, REQ_PREV_FIELD);
        break;
      default: /* Feldeingabe */
        form_driver(fo, ch);
    }
  }

  return (0);
}
```

Compilieren, Linken: `gcc -o bsp bsp.c -lform -lncurses`

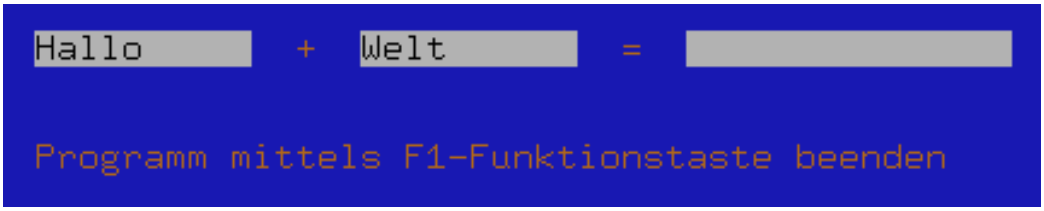

Abbildung 40: Bild:Ncurses40.png

8.4 Zugriff auf den Formularfeldpuffer

Auf ein bestimmtes Feld zugreifen: `FIELD *current_field(const FORM *);`
`int field_index(const FIELD *field);`

Feldpuffer auslesen: `char *field_buffer(const FIELD *field, int buffer);`

Feldpuffer belegen: `int set_field_buffer(FIELD *field, int buf, const char *value);`

8.4.1 Beispiel

```
#include <form.h>
#include <stdlib.h>
#include <string.h>

FIELD  **fi;
FORM   *fo;

void quit(void)
{
  int i;

  unpost_form(fo);
  free_form(fo);

  for(i=0; i<=3; i++)
  {
    free_field(fi[i]);
```

```
  }

  free(fi);
  endwin();
}

int main(void)
{
  int ch, i;

  initscr();
  atexit(quit);
  clear();
  noecho();
  cbreak();
  keypad(stdscr, TRUE);
  start_color();

  init_pair(1, COLOR_YELLOW, COLOR_BLUE);
  init_pair(2, COLOR_BLACK, COLOR_WHITE);

  bkgd(COLOR_PAIR(1));

  fi = (FIELD **)calloc(4, sizeof(FIELD *));
  fi[0] = new_field(1, 10, 2, 3, 0, 0);
  fi[1] = new_field(1, 10, 2, 18, 0, 0);
  fi[2] = new_field(1, 15, 2, 33, 0, 0);
  fi[3] = 0;

  for(i=0; i<3; i++)
  {
    set_field_fore(fi[i], COLOR_PAIR(2));
    set_field_back(fi[i], COLOR_PAIR(2));
  }

  fo = new_form(fi);
  post_form(fo);

  mvaddstr(2, 15, "+");
  mvaddstr(2, 30, "=");
  mvaddstr(5, 3, "Programm mittels F1-Funktionstaste beenden");
```

```
  refresh();

  while((ch=getch()) != KEY_F(1))
  {
    switch(ch)
    {
      case KEY_RIGHT:
      {
        char str[20];

        form_driver(fo, REQ_NEXT_FIELD);

        if(field_index(current_field(fo)) == 2)
        {
          snprintf(str, 20, "%s%s", field_buffer(fi[0], 0), field_buffer(fi[1], 0));
          set_field_buffer(fi[2], 0, str);
          refresh();
        }
        break;
      }
      case KEY_LEFT:
        form_driver(fo, REQ_PREV_FIELD);
        break;
      default: /* Feldeingabe */
        form_driver(fo, ch);
    }
  }
  return (0);
}
```

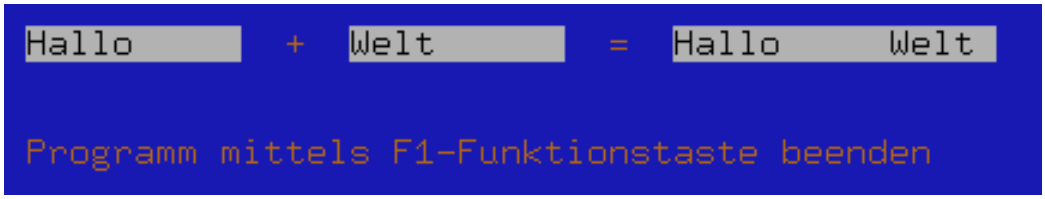

Abbildung 41: Bild:Ncurses41.png

8.5 Textausrichtung

Die Ausrichtung eines Textes im Formularfeld ist mittels `int set_field_just(FIELD *field, int justification);` einstellbar.

Mögliche Ausrichtungsoptionen sind:

- JUSTIFY_RIGHT
- JUSTIFY_LEFT
- JUSTIFY_CENTER
- NO_JUSTIFICATION

8.5.1 Beispiel (Programmausschnitt)

```
// ...
set_field_fore(fi[i], COLOR_PAIR(2));
set_field_back(fi[i], COLOR_PAIR(2));
set_field_just(fi[i], JUSTIFY_RIGHT);
// ...
```

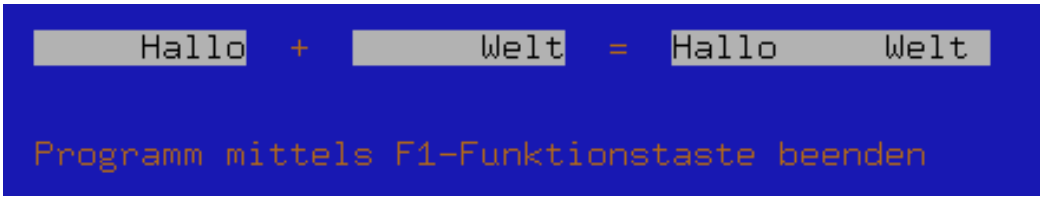

Abbildung 42: Bild:Ncurses42.png

8.6 Feldoptionen

Selbstverständlich gibt es auch für Formularfelder ein Menge Optionen. Gesetzt und abgefragt werden können sie mit diesen Funktionen: `int set_field_opts(FIELD *field, OPTIONS opts);`

`int field_opts_on(FIELD *field, OPTIONS opts);`

```
int field_opts_off(FIELD *field, OPTIONS opts);
OPTIONS field_opts(const FIELD *field);
```

Einige der möglichen Optionen sind:

O_VISIBLE	Formularfeldsichtbarkeit
O_ACTIVE	Feld ist aktiv
O_PUBLIC	Text ist bei der Eingabe sichtbar (z.B. bei Passworteingaben diese Option deaktivieren)
O_EDIT	Im Feld kann editiert werden
O_WRAP	Zeilenumbbruch
O_AUTOSKIP	Wenn Feld vollgeschrieben ist, gehe automatisch zum nächsten Feld
O_STATIC	Ein Feld kann nur die Zeichenanzahl entsprechend der Formularfeldgröße aufnehmen. Werden mehr Zeichen eingegeben so wird zum nächsten Formularfeld gesprungen. Ist O_AUTOSKIP deaktiviert, so werden zusätzliche Zeichen ignoriert. Ist O_STATIC ausgeschaltet, so kann das Formularfeld über die Formularfeldgröße Zeichen aufnehmen (die Darstellung wird gescrollt).

8.6.1 Beispiel: Auswirkungen O_AUTOSKIP und O_STATIC

Beim Eintippen der Zeichenkette ”Das ist nicht OK” in ein Formularfeld passiert je nach gesetzten Optionen folgendes

O_AUTOSKIP an, O_STATIC an (Standard):

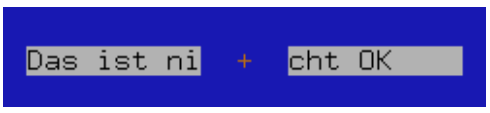

Abbildung 43: Bild:Ncurses43.png

O_AUTOSKIP aus:

Abbildung 44: Bild:Ncurses44.png

O_STATIC aus:

Abbildung 45: Bild:Ncurses45.png

8.7 Formularfeldtypen

Oft ist es sinnvoll und notwendig die Eingabemöglichkeiten in ein Formularfeld einzuschränken (z.B. nur Zahlen oder alphabetische Zeichen sind erlaubt). Dies ist mit der *form*-Bibliothek recht weitgehend möglich. Mit der Funktion `int set_field_type(FIELD *field, FIELDTYPE *type, ...);`

lassen sich die Feldtypen einstellen. Folgende Alternativen sind möglich

TYPE_ALPHA	nur Alphabetzeichen sind erlaubt
TYPE_ALNUM	Alphanumerische Zeichen sind erlaubt
TYPE_ENUM	Nur Einträge aus einer Stringliste sind erlaubt
TYPE_INTEGER	Nur ganze Zahlen sind erlaubt (optional mit vorangestelltem + oder -)
TYPE_NUMERIC	Numerische Daten (optional mit vorangestelltem + oder - und mit Dezimalpunkt)
TYPE_REGEXP	Feldeintrag muss zu einem regulären Ausdruck passen
TYPE_IPV4	Eine IPv4-Adresse

Es können auch eigene Formularfeldtypen festgelegt werden. Die Abhandlung dieses Themas würde jedoch im Rahmen dieses Tutorials zu weit führen. Nachfolgend ein einfaches Beispiel mit INTEGER- und NUMERIC-Formularfeldern.

8.7.1 Beispiel

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

FIELD  **fi;
FORM   *fo;

void quit(void)
{
  int i;

  unpost_form(fo);
  free_form(fo);

  for(i=0; i<=3; i++)
  {
    free_field(fi[i]);
  }

  free(fi);
  endwin();
}

int main(void)
{
  int ch, i;

  initscr();
  atexit(quit);
  clear();
  noecho();
  curs_set(0);
  cbreak();
  keypad(stdscr, TRUE);
  start_color();

  init_pair(1, COLOR_YELLOW, COLOR_BLUE);
  init_pair(2, COLOR_BLACK, COLOR_WHITE);
```

```
bkgd(COLOR_PAIR(1));

fi = (FIELD **)calloc(4, sizeof(FIELD *));
fi[0] = new_field(1, 10, 2, 3, 0, 0);
fi[1] = new_field(1, 10, 2, 18, 0, 0);
fi[2] = new_field(1, 15, 2, 33, 0, 0);
fi[3] = 0;

for(i=0; i<3; i++)
{
  set_field_fore(fi[i], COLOR_PAIR(2));
  set_field_back(fi[i], COLOR_PAIR(2));
  field_opts_off(fi[i], O_AUTOSKIP);
}

set_field_type(fi[0], TYPE_INTEGER, 0, -9999999, 9999999);
set_field_type(fi[1], TYPE_NUMERIC, 3, -9999999.999, 9999999.999);
field_opts_off(fi[2], O_EDIT);

fo = new_form(fi);
post_form(fo);

mvaddstr(2, 15, "+");
mvaddstr(2, 30, "=");
mvaddstr(5, 3, "Programm mittels F1-Funktionstaste beenden");

refresh();

while((ch=getch()) != KEY_F(1))
{
  switch(ch)
  {
    case KEY_RIGHT:
    {
      double z1, z2;
      char str[20];

      form_driver(fo, REQ_NEXT_FIELD);

      if(field_index(current_field(fo)) == 2)
      {
```

```
          z1 = atof(field_buffer(fi[0], 0));
          z2 = atof(field_buffer(fi[1], 0));
          snprintf(str, 20, "%f", z1+z2);
          set_field_buffer(fi[2], 0, str);
          refresh();
        }
        break;
      }
      case KEY_LEFT:
        form_driver(fo, REQ_PREV_FIELD);
        break;
      default: /* Feldeingabe */
        form_driver(fo, ch);
    }
  }

  return (0);
}
```

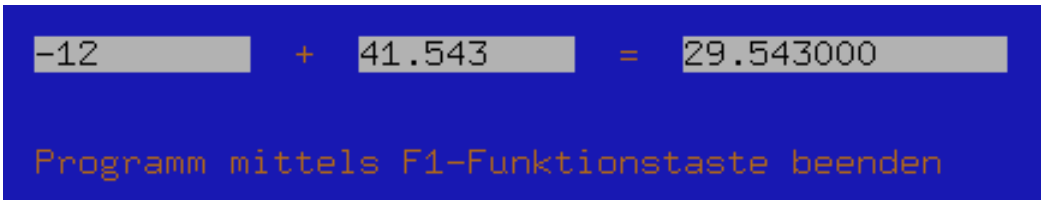

Abbildung 46: Bild:Ncurses46.png

8.8 Formularfenster

Das Zuweisen von Haupt- und Unterfenster geschieht äquivalent der Vorgehensweise bei einem Menü.

```
int set_form_win(FORM *form, WINDOW *win);
int set_form_sub(FORM *form, WINDOW *sub);
int scale_form(const FORM *form, int *rows, int *columns);
```

Kapitel 9

Mausunterstützung

Ncurses bietet Mausunterstützung. Standardmäßig ist diese Funktionalität deaktiviert. Zur Aktivierung ist die Funktion `mmask_t mousemask(mmask_t newmask, mmask_t *oldmask);`

mit der gewünschten Bitmaske aufzurufen. Vordefinierte Masken sind (auszugsweise):

ALL_MOUSE_EVENTS	Alle möglichen Mausereignisse
REPORT_MOUSE_POSITION	Mausposition melden
BUTTON1_CLICKED	Maustaste 1 geklickt
BUTTON1_DOUBLE_CLICKED	Maustaste 1 doppelgeklickt
BUTTON2_CLICKED	Maustaste 2 geklickt
BUTTON2_DOUBLE_CLICKED	Maustaste 2 doppelgeklickt
BUTTON3_CLICKED	Maustaste 3 geklickt
BUTTON3_DOUBLE_CLICKED	Maustaste 3 doppelgeklickt
BUTTON_SHIFT	Zusätzlich SHIFT-Taste gedrückt
BUTTON_CTRL	Zusätzlich STRG-Taste gedrückt
BUTTON_ALT	Zusätzlich ALT-Taste gedrückt

Die Abfrage des Auftretens eines durch die Mausmaske festgelegten sichtbaren Mausereignisses kann durch `int getmouse(MEVENT *event);`

erfolgen.

9.1 Beispiel

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

MEVENT *mev;

void quit(void)
{
  free(mev);
  endwin();
}

int main(void)
{
  int ch;
  mev = (MEVENT *)malloc(sizeof(MEVENT));

  initscr();
  atexit(quit);
  clear();
  noecho();
  curs_set(0);
  cbreak();
  keypad(stdscr, TRUE);
  start_color();
  mousemask(BUTTON1_CLICKED, 0);

  init_pair(1, COLOR_YELLOW, COLOR_BLUE);
  bkgd(COLOR_PAIR(1));
  mvaddstr(5, 3, "Programm durch anklicken der Maustaste 1 beenden");
  refresh();

  for(;;)
  {
    ch=getch();

    switch(ch)
    {
      case KEY_MOUSE:
```

```
      {
        if(getmouse(mev) == OK)
        {
          exit(0);
        }
      }
    }
  }

  return (0);
}
```

Abbildung 47: Bild:Ncurses50.png

Kapitel 10

Pads

Ein Pad (Schreibblock) ist eine Variante des konventionellen *ncurses*-Fensters. In der Anwendung unterscheidet es sich deutlich von diesem. Pads können größer als der Screen selbst sein und der Programmierer entscheidet erst beim Refresh, welcher Teil des Pads wo sichtbar ist.

10.1 Ein Pad erstellen

```
WINDOW *newpad(int nlines, int ncols);
```

Bedeutung der Parameter:

- `nlines, ncols` ... Anzahl der Pad-Zeilen und -Spalten

10.2 Ein Pad refreshen

```
int prefresh(WINDOW *pad, int pminrow, int pmincol,
             int sminrow, int smincol, int smaxrow, int smaxcol);
int pnoutrefresh(WINDOW *pad, int pminrow, int pmincol,
                 int sminrow, int smincol, int smaxrow, int smaxcol);
```

Die Pad-Refreshfunktionen haben prinzipiell die selben Aufgaben wie ihre Window-Pendants.

Bedeutung der Parameter:

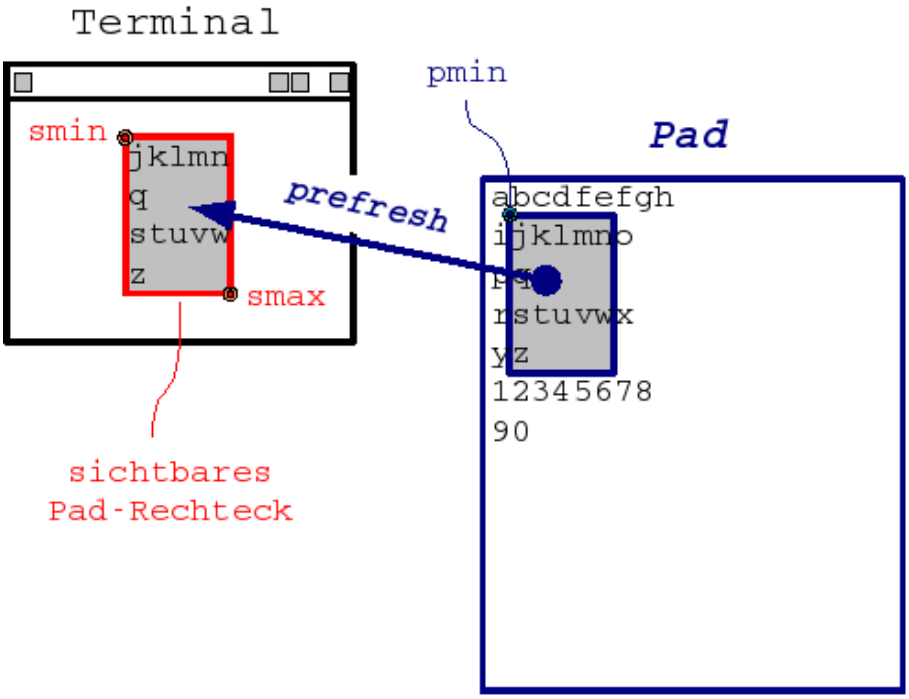

Abbildung 48

- `pminrow, pmincol` ... Ecke links-oben im Pad (p ... **p**ad).
- `sminrow, smincol, smaxrow, smaxcol` ... Ecke links-oben, Ecke rechts-unten. Diese Parameter bestimmen die Größe des auf dem Screen angezeigten Pad-Rechtecks, innerhalb dessen der Pad-Inhalt dargestellt wird (s ... **s**creen).

10.3 Beispiel

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

WINDOW *pad;

void quit(void)
{
  delwin(pad);
```

```
  endwin();
}

int main(void)
{
  initscr();
  atexit(quit);
  clear();
  noecho();
  curs_set(0);
  cbreak();
  keypad(stdscr, 1);

  start_color();
  init_pair(1, COLOR_YELLOW, COLOR_BLUE);
  init_pair(2, COLOR_BLUE, COLOR_WHITE);

  pad = newpad(300, 100);

  bkgd(COLOR_PAIR(1));
  wbkgd(pad, COLOR_PAIR(2));

  waddstr(pad, "Zeile 1 \n");
  waddstr(pad, "Zeile 2\n");
  waddstr(pad, "Diese Zeichenkette befindet sich in Zeile 3\n");
  waddstr(pad, "und diese in Zeile 4");

  refresh();
  prefresh(pad, 0, 0, 3, 3, 10, 30);

  getch();
  refresh();
  prefresh(pad, 2, 2, 12, 2, 20, 45);

  while(getch() != KEY_F(1))
  {
  }

  return(0);
}
```

Nach dem Programmstart

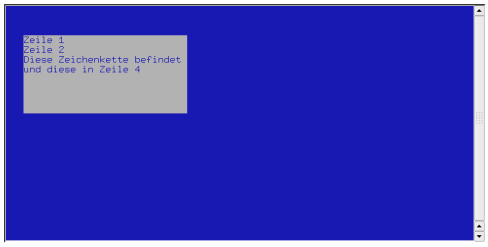

Abbildung 49

Nach dem ersten Tastendruck

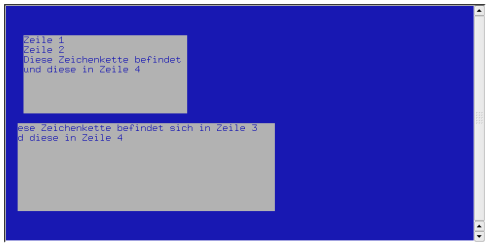

Abbildung 50

Kapitel 11

Was fehlt

Jedes einzelne Kapitel könnte noch beliebig erweitert werden. Viele *ncurses*-Funktionen passend zu den einzelnen Kapiteln wurden überhaupt nicht erwähnt. Aber dieses Buch soll auch ein Tutorial darstellen und kein Referenzhandbuch. Vieles von dem, was gezeigt wurde lässt sich sicher auch effizienter und kompakter codieren. Die Beispiele wurden möglichst einfach gehalten und sollen dem grundlegenden Verständnis dienen. Dementsprechend fehlen praxisbezogene Beispiele.

Welche Themen wurden komplett ausgespart?

- Die `intrflush()`-Funktion
- User-Pointer
- Hook-Funktionen: z. B. für Menüs und Formulare.
- Diverse Utility-Funktionen
- Low-level-Funktionen für verschiedene spezielle ncurses-Fähigkeiten
- slk: soft function-key labels

Desweiteren bietet *ncurses* auch Anbindungen an die Programmiersprachen Ada und C++. Interessant und hilfreich sind auch speziell auf *curses* aufbauende Widgetbibliotheken, wie z. B. CDK (Curses Development Kit).

Kapitel 12

Weblinks

12.1 Weitere Informationen zu ncurses

- Thomas E. Dickey: ncurses-FAQ
- Free Software Foundation: Announcing ncurses
- invisible-island.net: ncurses
- Eric S. Raymond, Zeyd M. Ben-Halim: Writing Programs with NCURSES
- Pradeep Padala: NCURSES Programming HOWTO
- Bernd Wocker, Klaus Furman: Die curses-Window-Bibliothek
- Reha K. Gerceker: Einführung in Ncurses
- Jürgen Pfeifer: Ada95 Binding for ncurses

12.2 Downloadmöglichkeit

- ncurses

12.3 Andere curses-Bibliotheken

- PDCurses

Kapitel 13

Autoren

Edits	User
32	Dirk Huenniger
1	Jonas Grote
49	Intruder

Kapitel 14

GNU Free Documentation License

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section “Entitled XYZ“ means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements“, “Dedications“, “Endorsements“, or “History“.) To “Preserve the Title“ of such a section when you modify the Document means that it remains a section “Entitled XYZ“ according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- Preserve the section Entitled “History“, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If the-

re is no section Entitled “History“ in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History“ section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- K. For any section Entitled “Acknowledgements“ or “Dedications“, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

- M. Delete any section Entitled “Endorsements“. Such a section may not be included in the Modified Version.

- N. Do not retitle any existing section to be Entitled “Endorsements“ or to conflict in title with any Invariant Section.

- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements“, provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History“ in the various original documents, forming one section Entitled “History“; likewise combine any sections Entitled “Acknowledgements“, and any sections Entitled “Dedications“. You must delete all sections Entitled “Endorsements.“

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate“ if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the

electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements“, “Dedications“, or “History“, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version“ applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Kapitel 15

Bildnachweis

In der nachfolgenden Tabelle sind alle Bilder mit ihren Autoren und Lizenen aufgelistet.

Für die Namen der Lizenzen wurden folgende Abkürzungen verwendet:

- GFDL: Gnu Free Documentation License. Der Text dieser Lizenz ist in einem Kapitel diese Buches vollständig angegeben.
- cc-by-sa-3.0: Creative Commons Attribution ShareAlike 3.0 License. Der Text dieser Lizenz kann auf der Webseite http://creativecommons.org/licenses/by-sa/3.0/ nachgelesen werden.
- cc-by-sa-2.5: Creative Commons Attribution ShareAlike 2.5 License. Der Text dieser Lizenz kann auf der Webseite http://creativecommons.org/licenses/by-sa/2.5/ nachgelesen werden.
- cc-by-sa-2.0: Creative Commons Attribution ShareAlike 2.0 License. Der Text der englischen Version dieser Lizenz kann auf der Webseite http://creativecommons.org/licenses/by-sa/2.0/ nachgelesen werden. Mit dieser Abkürzung sind jedoch auch die Versionen dieser Lizenz für andere Sprachen bezeichnet. Den an diesen Details interessierten Leser verweisen wir auf die Onlineversion dieses Buches.
- cc-by-sa-1.0: Creative Commons Attribution ShareAlike 1.0 License. Der Text dieser Lizenz kann auf der Webseite http://creativecommons.org/licenses/by-sa/1.0/ nachgelesen werden.
- cc-by-2.0: Creative Commons Attribution 2.0 License. Der Text der englischen Version dieser Lizenz kann auf der Webseite http://creativecommons.org/licenses/by/2.0/ nachgelesen werden. Mit dieser Abkürzung sind jedoch auch die Versionen dieser Lizenz für andere Sprachen bezeichnet. Den an diesen Details interessierten Leser verweisen wir auf die Onlineversion dieses Buches.
- GPL: GNU General Public License Version 2. Der Text dieser Lizenz kann auf der Webseite http://www.gnu.org/licenses/gpl-2.0.txt nachgelesen werden.
- PD: This image is in the public domain. Dieses Bild ist gemeinfrei.
- ATTR: The copyright holder of this file allows anyone to use it for any purpose, provided that the copyright holder is properly attributed. Redistribution, derivative work, commercial use, and all other use is permitted.

Bild	Autor	Lizenz
1	Haeber	GFDL
2	Russell Boltz	GPL
3	screenshot maked by Deutscher Friedensstifter	GPL
4	User:Cljk	GFDL
5		PD
6		PD
7		PD
8		PD
9		PD
10		PD
11		PD
12		PD
13		PD
14		PD
15		PD
16		PD
17		PD
18		PD
19		PD
20		PD
21		PD
22		PD
23		PD
24		PD
25		PD
26		PD
27		PD
28		PD
29		PD
30		PD
31		PD
32		PD
33		PD
34		PD
35		PD
36		PD
37		PD
38		PD
39		PD
40		PD

41		PD
42		PD
43		PD
44		PD
45		PD
46		PD
47		PD
48		PD
49		PD
50		PD