9.6 GTK/Parameter an die Callback-Funktion übergeben

9.6.1 Ziel

Wenn man eine Box benutzt, kann man ganz einfache mehrere Widgets, also z. B. Etiketten und Schaltflächen in einem Programm benutzen.

Nun ist es aber so, dass die Callback-Funktion als Parameter nur einen Zeiger auf *das* Widget bekommt, das das Ereignis ausgelöst hat. Beim Signal "clicked" bekommt man in der zugehörigen Callback-Funktion nur den Zeiger auf den Knopf. Auf das Etikett hat man keinen Zugriff. Da wäre es doch schön, wenn man in der Funktion auch auf bestimmte andere (oder alle anderen) Widgets Zugriff hätte.

9.6.2 Prinzip

Genau das ist möglich mit einem zusätzlichen Parameter der Callback-Funktion. Hierzu einen Blick auf das folgende Beispiel (parameter1.c):

```

1 #include <gtk/gtk.h>
2 void kg(GtkWidget *knopf, gpointer daten)
3 {
4     int *pzahl;
5     pzahl=(int *) daten;
6     g_print("Es_war_Knopf_%d\n", *pzahl);
7 }
8 int main(int argc, char *argv[])
9 {
10    int a=1, b=2, c=3, d=4;
11    GtkWidget *fenster;
12    GtkWidget *schachtel;
13    GtkWidget *knopfa, *knopfb, *knopfc, *knopfd;
14
15    gtk_init(&argc, &argv);
16    fenster=gtk_window_new(GTK_WINDOW_TOPLEVEL);
17    schachtel=gtk_box_new(GTK_ORIENTATION_HORIZONTAL, 2);
18    knopfa=gtk_button_new_with_label("Knopf_A");
19    knopfb=gtk_button_new_with_label("Knopf_B");
20    knopfc=gtk_button_new_with_label("Knopf_C");
21    knopfd=gtk_button_new_with_label("Knopf_D");
22
23    gtk_container_add(GTK_CONTAINER(fenster), schachtel);
24    gtk_box_pack_start(GTK_BOX(schachtel), knopfa, TRUE, FALSE, 2);
25    gtk_box_pack_start(GTK_BOX(schachtel), knopfb, TRUE, FALSE, 2);
26    gtk_box_pack_start(GTK_BOX(schachtel), knopfc, TRUE, FALSE, 2);
27    gtk_box_pack_start(GTK_BOX(schachtel), knopfd, TRUE, FALSE, 2);
28
29    g_signal_connect(G_OBJECT(fenster), "delete_event", gtk_main_quit, NULL);
30    g_signal_connect(G_OBJECT(knopfa), "clicked", G_CALLBACK(kg), &a);
31    g_signal_connect(G_OBJECT(knopfb), "clicked", G_CALLBACK(kg), &b);
32    g_signal_connect(G_OBJECT(knopfc), "clicked", G_CALLBACK(kg), &c);
33    g_signal_connect(G_OBJECT(knopfd), "clicked", G_CALLBACK(kg), &d);
34    gtk_widget_show_all(fenster);
35    gtk_main();
36    return 0;
37 }

```

Zeile 30 Der letzte Parameter der Funktion bekommt als Inhalt einen Zeiger auf `a`.

Zeile 2 Die Callback-Funktion bekommt diesen Zeiger unter dem Namen `daten`.

Zeile 5 Dieser Zeiger wird nach `pzahl` kopiert.

Zeile 6 Der Inhalt der Variablen `a`, auf die `pzahl` zeigt, wird auf die Konsole ausgegeben.

Was macht das Programm eigentlich? Hier können die Knöpfe 1, 2, 3 und 4 gedrückt werden. In jedem Fall wird dieselbe Callback-Funktion aufgerufen. Sie soll aber wissen, wer sie aufgerufen hat, um mit `g_print()` die richtige Zahl schreiben zu können.

Das Geheimnis liegt im bisher unbenutzten dritten Parameter in `g_signal_connect()`: Was hierin steht, wird im letzten Parameter an die Callback-Funktion übergeben und kann darin benutzt werden. Der Parameter ist vom Typ `void*`, also ein beliebiger Zeiger.

Diese Methode funktioniert leider nicht bei GDK-Ereignissen. Ein GDK-Ereignis ist allerdings leicht erkennbar an seinem Namen, der auf `event` endet, wie z. B. `delete-event`, `focus-in-event` usw.

9.6.3 Nutzen: Zeiger auf ein Widget

Der letzte Parameter der Callback-Funktion kann auch so genutzt werden, dass man den Merker eines Widgets übergibt. Das zeigt das Beispiel (`parameter2.c`):

```

1 #include <gtk/gtk.h>
2 void knopfgeklickt(GtkWidget *knopf, gpointer e)
3 {
4     gtk_label_set_text(GTK_LABEL(e), "Danke!");
5 }
6 int main(int argc, char *argv[])
7 {
8     GtkWidget *fenster;
9     GtkWidget *schachtel;
10    GtkWidget *knopf;
11    GtkWidget *etikett;
12    gtk_init(&argc, &argv);
13    fenster=gtk_window_new(GTK_WINDOW_TOPLEVEL);
14    schachtel=gtk_box_new(GTK_ORIENTATION_VERTICAL,2);
15    knopf=gtk_button_new_with_mnemonic("Bitte_klicken");
16    etikett=gtk_label_new("...");
17    gtk_container_add(GTK_CONTAINER(fenster), schachtel);
18    gtk_box_pack_start(GTK_BOX(schachtel), etikett, TRUE, FALSE, 2);
19    gtk_box_pack_start(GTK_BOX(schachtel), knopf, TRUE, FALSE, 2);
20    g_signal_connect(G_OBJECT(fenster),"delete_event",gtk_main_quit,NULL);
21    g_signal_connect(G_OBJECT(knopf), "clicked",
22                    G_CALLBACK(knopfgeklickt), etikett);
23    gtk_widget_show_all(fenster);
24    gtk_main();
25    return 0;
26 }

```

Zeile 22 Der letzte Parameter der Funktion bekommt als Inhalt den Merker des Etiketts.

Zeile 2 Die Callback-Funktion bekommt diesen Zeiger unter dem Namen `e`.

Zeile 4 Dieser Zeiger wird mit dem Makro `GTK_LABEL()` überprüft und als Parameter an die Funktion `gtk_label_set_text()` weitergegeben.

9.6.4 Nutzen: Zeiger auf mehrere Objekte

Nun soll beim obigen Programm die Ausgabe auf zwei verschiedenen Etiketten erfolgen. Jetzt müsste man also zwei Zeiger übergeben. Man hat aber nur einen Zeiger zur Verfügung. Da bieten sich drei Lösungen an:

- Man legt alle Widgets in ein Record und übergibt einen Zeiger darauf; dann kann man alle Widgets mit ihrem Namen ansprechen
- Man legt alle Widgets in ein Array von Widgets und übergibt einen Zeiger darauf; mit dem richtigen Index kann man dann auf jedes zugreifen
- Man kehrt wieder dahin zurück, alle Widgets als globale Variablen anzulegen; diese Möglichkeit scheidet aus

Das Beispiel (parameter3.c) löst das Problem mit einem Record:

```

1 #include <gtk/gtk.h>
2 struct gui_t
3 {
4     GtkWidget *fenster;
5     GtkWidget *schachtel;
6     GtkWidget *knopf;
7     GtkWidget *etikett1;
8     GtkWidget *etikett2;
9 };
10 void knopfgeklickt(GtkWidget *knopf, gpointer daten)
11 {
12     struct gui_t *pgui;
13     pgui=daten;
14     gtk_label_set_text(GTK_LABEL((*pgui).etikett1), "Danke!");
15     gtk_label_set_text(GTK_LABEL((*pgui).etikett2), "Sehr_erfreut!");
16 }
17 int main(int argc, char *argv[])
18 {
19     struct gui_t gui;
20     gtk_init(&argc, &argv);
21     gui.fenster=gtk_window_new(GTK_WINDOW_TOPLEVEL);
22     gui.schachtel=gtk_box_new(GTK_ORIENTATION_VERTICAL,2);
23     gui.knopf=gtk_button_new_with_mnemonic("Bitte_klicken");
24     gui.etikett1=gtk_label_new("...");
25     gui.etikett2=gtk_label_new("");
26     gtk_container_add(GTK_CONTAINER(gui.fenster), gui.schachtel);
27     gtk_box_pack_start(GTK_BOX(gui.schachtel), gui.etikett1, TRUE, FALSE, 2);
28     gtk_box_pack_start(GTK_BOX(gui.schachtel), gui.knopf, TRUE, FALSE, 2);
29     gtk_box_pack_start(GTK_BOX(gui.schachtel), gui.etikett2, TRUE, FALSE, 2);
30     g_signal_connect(G_OBJECT(gui.fenster), "delete_event",
31                     gtk_main_quit, NULL);
32     g_signal_connect(G_OBJECT(gui.knopf), "clicked",
33                     G_CALLBACK(knopfgeklickt), &gui);
34     gtk_widget_show_all(gui.fenster);
35     gtk_main();
36     return 0;
37 }

```

Zeile 2-9 Hier ist der Record als Datentyp struct gui_t. Alle Widgets sind enthalten.

Zeile 19 Und hier wird die Variable `gui` in `main()` vereinbart. Keine Globale Variable!

Zeile 32 Der letzte Parameter der Funktion bekommt als Inhalt die Adresse der Variablen `gui`.

Zeile 10 Die Callback-Funktion bekommt Zeiger unter dem Namen `daten`.

Zeile 11 `pgui` ist ein Zeiger auf `struct gui_t`.

Zeile 13 `daten` wird kopiert in `pgui`; damit findet eine Typumwandlung statt.

Zeile 14 Das, worauf `pgui` zeigt, ist unser Record `*pgui`. Dessen Element `etikett1` wird nun mit dem Punkt-Operator ausgewählt.

Das Etikett `(*pgui).etikett` wird mit dem Makro `GTK_LABEL()` überprüft und an `gtk_label_set_text()` weitergegeben.

Zeile 15 Dasselbe passiert hier mit `etikett2`.

Je nach Oberfläche des Programms ist der Record beliebig erweiterbar. Man kann also mit Hilfe des Zeigers auf beliebig viele Elemente zugreifen.

Übrigens kann man die Zeilen 14 und 15 auch anders darstellen, wenn man den Pfeil-Operator benutzt:

```
1   gtk_label_set_text(GTK_LABEL(pgui->etikett1), "Danke!");  
2   gtk_label_set_text(GTK_LABEL(pgui->etikett2), "Sehr_erfreut!");
```

`x->y` bedeutet nämlich dasselbe wie `(*x).y`, sieht aber einfacher aus.