```
1  /*****************************************************************************
2  *
3  * Signale                                        Filename: "signal_c.cc"
4  *
5  * Programmbeschreibung:
6  *
7  * Es wird ein Vater- und ein Kindprozess erzeugt. Der Vaterprozess steuert
8  * mit verschiedenen Signalen den Ablauf des Kindprozesses.
9  *
10 *****************************************************************************
11 *
12 * Projekt      : Linux IPC Praktikum
13 *
14 * Datum/Name   : 25-Mai-98  durch M. Rueesch und D. Eisenegger
15 *
16 * Aenderungen  : 6-11-00 Markus Thaler: Signale mit sigaction()
17 *
18 *****************************************************************************/
19
20 //---------------------------------------------------------------------------
21 // Include Files
22 //---------------------------------------------------------------------------
23 #include <stdio.h>
24 #include <stdlib.h>
25 #include <signal.h>                            // Signalfunktionen
26 #include <unistd.h>
27
28 //---------------------------------------------------------------------------
29 // Lokale Funktionen
30 //---------------------------------------------------------------------------
31 void sig_handler (int sig);
32
33 //---------------------------------------------------------------------------
34 // Globale Variabeln
35 //---------------------------------------------------------------------------
36 char exit_flag = 1;
37
38
39 //****************************************************************************
40 // FUNKTION: main ()                           Kindprozess
41 //****************************************************************************
42
43 int main (void){
44
45    int i = 0;
46    struct sigaction neuAction;
47
48    neuAction.sa_handler = sig_handler;
49    neuAction.sa_flags = SA_RESTART;
50    sigemptyset(&neuAction.sa_mask);
51    sigaction (SIGINT, &neuAction, NULL);
52    sigaction (SIGUSR1, &neuAction, NULL);
53
54    printf (" Kind geboren (PID: %d, PPID: %d)\n", getpid(), getppid());
55
56    while (exit_flag){                          // bleibt hier bis SIGUSR1 oder
57       printf (" KIND: erste Schlaufe %d\n", i); // SIGINT empfangen wurde.
58       i++;
59       sleep(1);
60    }
61
62    i = 0;
```

```
63    while (1){                                    // bleibt hier bis SIGINT
64       printf (" KIND: zweite Schlaufe %d\n", i);// empfangen wurde.
65       i++;
66       sleep(1);
67    }
68    printf (" Kind normal terminiert\n");
69    exit (0);
70 }
71
72 //*************************************************************************
73 // HANDLER handler ()
74 //*************************************************************************
75
76 void sig_handler (int sig){
77
78    switch (sig){
79       case SIGINT:{                              // SIGINT terminiert das Kind
80          printf ("KIND: Signal SIGINT empfangen\n");
81          sleep(2);
82          printf ("*** Kind terminiert. ***\n");
83          exit(5);                                // exit mit einem Statuswert
84          break;
85       }
86       case SIGUSR1:{                             // SIGUSR1 bringt das Kind in
87          printf ("KIND: Verlasse erste Schlaufe\n");  // die zweite Schlaufe
88          exit_flag = 0;
89          break;
90       }
91       default:
92          break;
93    }
94 }
```