K 9. Übung

K.1 Überblick

- Besprechung 5. Aufgabe (trsh)
- Terminaltreiber

K.2 Eingabeverarbeitung des Terminals

- Zeilenmodus (“canonical mode”):
  Tastatur-Eingabe wird vom Terminaltreiber vorverarbeitet
  - ➤ Pufferung von Zeilen
  - ➤ Einfache Editier-Funktion (Zeichen oder Zeile löschen)
  - ➤ Erkennen von Sonderzeichen (Signale, EOF)
- Einzelzeichenmodus (“non-canonical mode”):
  Tastatur-Eingabe wird “sofort” an das Programm weitergeleitet
  - ➤ keine Zeilenpufferung
  - ➤ keine Eingabebearbeitung möglich
- Verhalten des Terminaltreibers ist konfigurierbar

K.3 Terminal-Parameter auslesen

```
#include <termios.h>
int tcgetattr(int fildes, struct termios *termios_p);
```

- Argumente
  - **filedes**: Filedeskriptor der mit einem Terminal verknüpft ist (z.B. STDIN_FILENO)
  - **termios_p**: Puffer für Terminal-Informationen
- Rückgabewert: 0 wenn OK, -1 wenn Fehler (siehe errno-Variable)

K.4 Terminal-Parameter setzen

```
#include <termios.h>
int tcsetattr(int fildes, int optional_actions,
                          const struct termios *termios_p);
```

- Argumente
  - **filedes**: Filedeskriptor der mit einem Terminal verknüpft ist (z.B. STDIN_FILENO)
  - **optional_actions**: Modus, wann die Änderungen wirksam werden
    - **TCSANOW**: sofort
    - **TCSADRAIN**: nachdem alle Ausgaben übermittelt wurden
    - **TCSAFLUSH**: wie TCSADRAIN, jedoch werden zusätzlich alle nicht verarbeitete Eingaben verworfen
  - **termios_p**: Puffer für Terminal-Informationen
- Rückgabewert: 0 wenn mindestens ein Wert gesetzt werden konnte,
  -1 bei Fehlern (siehe errno-Variable)

K.5 Die Struktur termios

```
typedef unsigned int tcflag_t;
typedef unsigned char cc_t;
typedef unsigned int speed_t;

struct termios {
        tcflag_t     c_iflag;      /* input modes */
        tcflag_t     c_oflag;      /* output modes */
        tcflag_t     c_cflag;      /* control modes */
        tcflag_t     c_lflag;      /* line discipline modes */
        cc_t         c_cc[NCCS];   /* control chars */
};
```

- **c_iflag**: Eingabekontrolle (z.B. Behandlung von CR und LF)
- **c_oflag**: Ausgabekontrolle (z.B. Umsetzung des ’\n’-Zeichens)
- **c_cflag**: Einstellungen der Hardware des Terminals (Baud-Rate)
- **c_lflag**: allgemeine Terminalfunktionen (z.B. canonical mode)
- **c_cc**: Kontroll-Zeichen bzw. Terminal-Variablen

K.6 Die Einzelzeicheneingabe

- Aktivieren des “non-canonical” Modus durch Löschen von **ICANON** in **c_lflag**:

```
settings.c_lflag &= ~ICANON;
```

- Eine Lese-Anforderung des Programms wird jedoch erst erfüllt wenn:
  - mindestens *MIN* Zeichen eingegeben wurden oder
  - die Zeit zwischen der Eingabe von zwei Zeichen länger ist als *TIME*.
- Der Wert *MIN* wird im Element **VMIN** des Feldes **c_cc** gespeichert, der Wert *TIME* im Element **VTIME**:

```
settings.c_cc[VMIN]  = MIN;
settings.c_cc[VTIME] = TIME;
```

K.7 Beispiel

```
#include <termio.h>
struct termios settings;

if ( tcgetattr(STDIN_FILENO, &settings) == -1 ){
  perror("Fehler bei tcgetattr: "); exit(EXIT_FAILURE);
}

settings.c_lflag &= ~ICANON;
settings.c_cc[VMIN] = 1;
settings.c_cc[VTIME] = 0;

if ( tcsetattr(STDIN_FILENO, TCSANOW, &settings) == -1 ) {
  perror("Fehler bei tcsetattr: "); exit(EXIT_FAILURE);
}

/* ...... */
```