Übungen zu Systemnahe Programmierung in C

Abschnitt WS.5: Aufgabe (trac)

21.12.2020

Tim Rheinfels
Benedict Herzog
Bernhard Heinloth

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg

Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Aufgabe: trac

- TRAC: TRanslate Arbitrary Characters

```
$ ./trac <FIND> <REPLACE>

$ ./trac aie bei     # a -> b;  i -> e;  e -> i
Dies ist ein Test    # Eingabe
Deis est ien Tist    # Ausgabe
```

- Ähnlich wie das Kommando `tr(1)` in Unix-artigen Betriebssystemen
- Programmablauf
  - Kommandozeilenparameter prüfen
    - Gegebenenfalls Fehlermeldung ausgeben und Programm beenden
  - Zeichen von `stdin` bis EOF oder Fehler einlesen
    - Zeichen entsprechend Abbildung ersetzen
    - Zeichen ausgeben
  - Fehler erkennen und passende Fehlermeldung ausgeben
  - Programm (mit entsprechendem exit-Status) beenden

Nützliche Bibliotheksfunktionen (1)

- `int getchar(void);`
  - Einzelnes Zeichen von `stdin` einlesen

  → getchar(3)
- `int putchar(int c);`
  - Einzelnes Zeichen auf `stdout` ausgeben

  → putchar(3)
- `size_t strlen(const char *s);`
  - Länge einer Zeichenfolge bestimmen

  → strlen(3)
- `int fprintf(FILE *stream, const char *format, ...);`
  - Formatierte Ausgaben auf einem Dateikanal (z.B. `stderr`)

  → fprintf(3)

Nützliche Bibliotheksfunktionen (2)

- `int ferror(FILE *stream);`
  - Fehlerzustand eines Dateikanals abfragen

  → ferror(3)

- `void perror(const char *s);`
  - Gibt die übergebene Zeichenfolge gefolgt von einer Beschreibung des letzten Fehlers auf `stderr` aus
  - Nutzt die globale `errno` Variable (⇒ nur für Bibliotheksfehler)

  → perror(3)

- `void exit(int status);`
  - Beenden des Programms mit übergebenem exit-Status

  → exit(3)

Mögliche Fehler

```
01 $ ./trac <FIND> <REPLACE>
```

- Bedienungsfehler
  - Falsche Anzahl an Argumenten
  - Unterschiedliche Länge von `FIND` und `REPLACE`
  - Selbes Zeichen kommt mehrfach in `FIND` vor

  ⇒ Erklärung der Verwendung auf `stderr` ausgeben und Programm beenden

- Ein-/Ausgabefehler
  - Zugriff auf eine Datei ist nicht möglich (umgeleiteter Dateikanal)

  ⇒ Fehler auf `stderr` ausgeben und Programm beenden

Fehlerbehandlung putchar()

```
01 if(putchar('A') == EOF) {
02   ...
03 }
```

- „fputc(), putc() and putchar() return the character written as an unsigned char cast to an int **or EOF on error.**”

Fehlerbehandlung getchar()

```
01 int c;
02 while ((c=getchar()) != EOF) {
03   ...
04 }
05
06 /* EOF oder Fehler? */
07 if(ferror(stdin)) {
08   /* Fehler */
09   ...
10 }
```

- „fgetc(), getc() and getchar() return the character read as an unsigned char cast to an int **or EOF on end of file or error.**”
- Wie kann man den Fehlerfall von EOF unterscheiden?

  $\Rightarrow$ `ferror(3)`