Blatt 04

Besprechung: 24. Mai 2012

Morse-Code

Beim Morse-Code werden Buchstaben- und Zahlzeichen in eine Folge von kurzen und langen Signalen (notiert als '.' bzw. '-', gesprochen als 'di(t)' bzw. 'dah') uebersetzt. So entspricht etwa die Signalfolge '.-' ('di-dah' bzw. 'kurz lang') dem Buchstaben 'A' und die Signalfolge '.-.' ('di-dah-dit' bzw. 'kurz lang kurz') dem Buchstaben 'R'. Klein- und Grossbuchstaben werden nicht unterschieden. Bis zu sechs Signale koennen notwendig sein, um ein Zeichen zu kodieren. Das Ende eines Zeichens ist an einer kurzen Uebertragungspause, notiert als Leerzeichen, zu erkennen. Beispielsweise steht ''-- .- .-. -.- ..- ...'' fuer ''MARKUS''.

Die Dateien `morse.h` und `morse.c`, die auf der Homepage der Vorlesung zur Verfuegung stehen, definieren bereits folgende zwei Funktionen:

`char *encode(unsigned char c);`

> Diese Funktion uebersetzt das Zeichen `c` in den zugehoergen Morse-Code. Zurueckgegeben wird ein Zeiger auf einen unveraenderlichen String, der nur '.' und '-' beinhaltet und nicht freigegeben werden muss. Wenn fuer das angegebene Zeichen kein Morse-Code definiert ist, wird ein NULL-Zeiger zurueckgegeben.

`unsigned char decode(char *str);`

> Diese Funktion leistet das Umgekehrte: Der String `cp`, der nur '.' und '-' beinhalten darf, wird als Morse-Code interpretiert. Zurueckgegeben wird der zu diesem Morse-Code gehoerende Buchstabe. Falls der Morse-Code ungueltig war, wird 0 zurueckgeliefert.

Das kurze Beispielprogramm `mtest.c` zeigt, wie die beiden Funktionen verwendet werden und gibt nebenbei eine Liste aller Zeichen mit ihrem Morse-Code aus.

Der Morse-Code beinhaltet keine Leerzeichen; obige Funktionen bilden jedoch die leere Signalfolge darauf ab.

1. Morse Uebertragung (mrecv.c und msend.c) (10P)

Ihr sollt zwei Programme schreiben, eines das Text in Morse-Code uebersetzt und diesen in Form von Unix-Signalen an einen anderen Prozess uebertraegt, und ein zweites, das die Signale empfaengt und den dekodierten Text ausgibt.

Dabei soll das Unix-Signal SIGUSR1 einem dit (kurzes Morse-Signal) und das Unix-Signal SIGUSR2 einem dah (langes Morse-Signal) entsprechen. Statt der Uebertragungspause wird das Signal SIGTERM verwendet.

Schreibt zuerst das argumentlose Empfangsprogramm *mrecv* und testet es, indem ihr ihm mit dem Kommando *kill* Signale schickt. Zu beachten ist, dass ggf. *fflush()* verwendet werden muss, damit ein ausgegebener Buchstabe sofort auf dem Bildschirm erscheint.

Schreibt danach das Sendeprogramm *msend*, das als Argument eine Prozess-ID bekommt und Text von der Standardeingabe liest. Der Text soll dann als Morse-Code kodiert und wie oben beschrieben an den angegebenen Prozess uebertragen werden. Verwendet dazu *kill()*.

Ein erster Versuch wird wahrscheinlich zeigen, dass der Text nicht richtig ankommt. Warum?

Synchronisiert eure Programme so, dass der Sender nach jedem gesendeten Signal eine Empfangsbestaetigung abwartet bevor er das naechste Signal sendet. Da bietet sich `sa_sigaction` der *sigaction()*-Schnittstelle an. Verwendet als Bestaetigungssignal SIGINT, da dies von der Shell ignoriert wird, was das Testen mit dem Befehl *kill* vereinfacht.

Die Programme `pingpong.c` und `strikeback.c` aus den Vorlesungsfolien bieten Ansatzpunkte.

2. Verbesserte Mine (5P)

Jetzt sollt ihr noch euer Minenprogramm von Blatt 2 verbessern. Falls einer der Arbeiter erfolgreich war und eine Loesung gefunden hat, dann musstet ihr bisher noch warten bis alle anderen Arbeiter zurueckgekehrt waren. Aendert euer Minenprogramm so ab, dass ihr die anderen Arbeiter in diesem Fall einfach per Signal toetet. Ihr muesst dazu nur `mine.c` anpassen, nicht `miner.c`.

Zum Einreichen eurer Loesung:

```
submit ss2 4 [team] [notes] mrecv.c msend.c mine.c
```

Fragen zur Selbstkontrolle

- Nenne drei verschiedene Einsatzzwecke fuer Signale mit Beispielen.
- Wie kann ein Prozess auf ein Signal reagieren?
- Welche Einschraenkungen gelten fuer den Code in Signalbehandlern?
- Was sind die Unterschiede zwischen *exit()*, *_exit()* und *_Exit()*?
- Was bedeutet *reentrant*-faehig zu sein? Wann ist eine Funktion das nicht?
- Nenne einen Anwendungsfall fuer SIGALRM?
- Wie unterscheidet man bei *read()* ob man am Ende der Datei angelangt ist oder ein Signal eingegangen ist? Wie unterscheidet man es bei *getchar()*?

- Wie versendet man Signale in C und wie in der Shell?
- Wie unterscheiden sich *signal()* und *sigaction()*? Welches ist die bessere Wahl? Warum?