Programmieren C: Vordefinierte Strukturen: Datumsrechnen

Klaus Kusche

Wir wollen ein Programm schreiben, das zu einem Datum den Wochentag
und den Tag im Jahr ermittelt, also z.B. *"Der 31. 12. 2010 (365. Tag des Jahres) ist ein Freitag"*.
Wir rechnen aber diesmal nicht selbst, sondern verwenden *fertige Funktionen*.

Diese fertigen Funktionen verwenden (wie viele vordefinierte Systemfunktionen in C)
vordefinierte Strukturen.
Die Typdefinitionen solcher Strukturen muss man nicht selbst schreiben:
Der Header, der zur Funktion gehört, enthält auch die dafür nötige **`struct`**-Definition.

In unserem Fall ist das der Header **`time.h`**, der den folgenden Strukturtyp definiert:

```
struct tm {
    int tm_sec;         /* seconds, 0...59 */
    int tm_min;         /* minutes, 0...59 */
    int tm_hour;        /* hours, 0...23 */
    int tm_mday;        /* day of the month, 1...31 */
    int tm_mon;         /* month, 0...11 !!! */
    int tm_year;        /* year-1900 !!! */
    int tm_wday;        /* day of the week, 0...6, 0 = Sun */
    int tm_yday;        /* day in the year, 0...364(5) !!! */
    int tm_isdst;       /* daylight saving time (0/1/-1) */
};
```

(je nach System enthält die Struktur dahinter noch weitere Member)

Unser Programm geht wie folgt vor:

- Das Datum wird mit 3 Zahlen (Tag, Monat, Jahr) auf der *Befehlszeile* angegeben.
- Wir *deklarieren* eine solche **`struct tm`** -Strukturvariable
  und *befüllen* die Stunden, Minuten und Sekunden mit 12:00:00
  und Tag, Monat und Jahr mit dem eingegebenen Datum.

  `tm_wday` und **`tm_yday`** dürfen wir undefiniert lassen.
  `tm_isdst` setzen wir auf **`-1`**.

  Du könntest statt sieben einzelnen Zuweisungen
  auch eine *Initialisierung für die ganze Struktur* verwenden.

  Achtung:

  Beachte die oben im Kommentar angegebenen *Wertebereiche* der Strukt-Member:
  Beim Monat muss bei der Eingabe *1 abgezogen* und bei der Ausgabe wieder
  dazugezählt werden, das Jahr muss zweimal *um 1900 korrigiert* werden.
- Dann rufen wir die vordefinierte Funktion **`mktime`** auf:

  Hinein geht ein *Pointer* auf unsere Struktur, und als Returnwert
  kommt ein Wert vom Typ **`time_t`** heraus, den wir uns merken
  (der **`typedef`**-Typ **`time_t`** ist normalerweise ein **`int`** oder **`long`**
  und enthält die *Sekunden* seit 1.1.1970).

  Wir sollten das Ergebnis auf **`-1`** prüfen:
  In diesem Fall war die Zeit in unserer Struktur außerhalb des Bereiches,
  den **`time_t`** darstellen kann, und wir sollten mit einer Fehlermeldung aufhören.

Anmerkung:

Auf 64-Bit-Rechnern ist **`time_t`** normalerweise ein 64 Bit Integer.
Das reicht für einen viel größeren Datumsbereich (knapp +/- 300 Milliarden Jahre),
als das 32-Bit-Jahr **`tm_year`** in der **`struct tm`** darstellen kann
(rund +/- 2 Milliarden Jahre).

Auf 32-Bit-Rechnern ist **`time_t`** nur eine 32-Bit-Zahl,
und damit funktioniert unser Programm nur von 1901-12-13 bis 2038-01-19
(das berühmte "Jahr-2038-Problem").

- Mit der vordefinierten Funktion **`localtime`** rechnen wir zurück:

  Hinein geht ein *Pointer* auf unsere **`time_t`**-Variable,
  und als *Returnwert* kommt ein *Pointer* auf eine befüllte **`struct tm`** heraus
  (die Ergebnis-Struktur ist eine statische lokale Variable von **`localtime`**).

- Wir geben zur Kontrolle Tag, Monat und Jahr aus dieser Struktur,
  auf die der Pointer zeigt, aus (nicht die eingegebenen Werte!),
  und dann den *Wochentag* (**`tm_wday`**) und den *Tag im Jahr* (**`tm_yday`**).

  Achtung:

  `tm_yday` beginnt bei **`0`** statt **`1`**, also 1 dazuzählen!

  Tipp:

  Deklariere ein *initialisiertes Array von 7 String-Pointern* mit den Wochentags-Namen
  und verwende **`tm_wday`** als *Index* in dieses Array, um den Wochentag als Text
  zu bekommen. **`tm_wday`** gleich **`0`** ist der Sonntag, **`1`** der Montag usw..

Anmerkung:

Laut C-Standard sollte **`mktime`** in der als Parameter übergebenen Strukur
gleich die richtigen Werte für **`tm_wday`** und **`tm_yday`** einsetzen.
Wir könnten uns daher den "Rückweg" über **`localtime`** *sparen*
und nach **`mktime`** gleich die Werte aus der ursprünglichen Struktur ausgeben.

Da sich aber noch nicht alle Systeme richtig an den Standard halten,
programmieren wir vorsichtshalber den umständlicheren Weg.