Systemprogrammierung: **popen** (und Signale)

Klaus Kusche

Gesucht ist ein Programm, das *einmal pro Sekunde*
die aktuelle CPU-Auslastung als Balken anzeigt.
Es soll dazu einen **`vmstat`**-Befehl verwenden, der einmal pro Sekunde
verschiedenste Performance-Werte als *Zahlen in Tabellenform* liefert.

Im Detail:

- Deklariere zuerst einen *Strukturtyp* mit drei **`int`**-Membern **`us`**, **`sy`** und **`wa`**
  (für User-Time, System-Time und I/O-Wait-Time).
- Schreib dann zwei kleine *Hilfsfunktionen*:
  - Die erste (mit **`char`**- und **`int`**-Parameter, ohne Returnwert) gibt
    das *angegebene Zeichen in der angegebenen Anzahl* aus (n mal nacheinander)
    (nimm **`putchar`** für die Ausgabe).
  - Die zweite (ebenfalls ohne Returnwert) bekommt eine solche *Strukturvariable*
    "by Reference" übergeben und gibt (durch drei Aufrufe der ersten Funktion)
    `us` viele **`U`**, **`sy`** viele **`S`** und **`wa`** viele **`W`** gefolgt von einem **`\n`** aus.
- Das Hauptprogramm startet zuerst mit **`popen`** den Befehl **`vmstat -nw 1`** so,
  dass es *den Output dieses Programms lesen* kann (prüfe auf Fehler,
  beende das Programm bei Fehlern mit einer schönen Fehlermeldung).
- In einer *Endlosschleife* liest es dann *zeilenweise* den Output dieses Befehls
  (verwende **`fgets`** zum Lesen):
  - Die *ersten 3 Zeilen* werden *überlesen* (ignoriert):
    Es handelt sich um 2 Zeilen mit der Spalten-Beschriftung
    und um eine Zeile mit den kumulativen Daten seit Systemstart.
  - Jede weitere Zeile enthält 17 **`int`**-Werte. Der Wert in der *13. Spalte* liefert das **`us`**
    unserer Struktur, die *14. Spalte* ist das **`sy`**, und die *16. Spalte* das **`wa`**, jeweils
    als *Prozent der Zeit* in der letzten Sekunde, d.h. als Zahl zwischen **`0`** und **`100`**.

    Verwandle den *gesamten Zeilentext auf einmal* mit **`sscanf`** in 17 **`int`**'s,
    speichere dabei die gesuchten Spalten in die *Struktur-Member*
    und alle anderen Spalten in eine *Dummy-Variable*.

    Prüfe, ob **`sscanf`** wirklich 17 Werte verwandelt hat,
    beende das Programm sonst mit einer Fehlermeldung.

    Ruf dann die *Hilfsfunktion* auf, um einen *Balken* mit diesen Werten auszugeben.
- Sollte **`fgets`** wider Erwarten einen *Fehler oder End-of-File* liefern,
  endet die Schleife (das sollte nicht passieren, denn **`vmstat`** sollte endlos laufen).

  Mach in diesem Fall ein **`pclose`**, analysiere dessen Ergebnis,
  und gib eine von drei möglichen *Fehlermeldungen* aus
  (du wirst wieder die Makros zum Decodieren des Exitstatus brauchen):

- **`pclose`** selbst schlug fehl: Fehlermeldung mit **`errno`**-*Fehlertext* ausgeben.
- **`vmstat`** endete normal: Fehlermeldung mit *Exitcode* ausgeben.
- **`vmstat`** starb an einem Signal: Fehlermeldung mit *Signal* ausgeben.

  Tipp: Falls Du das Signal nicht als Nummer, sondern *als Text* ausgeben willst, hilft dir **`strsignal`**. Dafür muss ganz oben im Programm
  `#define _POSIX_C_SOURCE 200809L`
  stehen, weil **`strsignal`** nur zum POSIX-Standard ab 2008 gehört, nicht zum normalen ISO-C-Standard.

Beende anschließend das Programm mit Fehlerstatus.

Zusatzaufgabe (Signale):

Wenn das Programm *mit Ctrl/C abgebrochen* wird,
soll es die *mittlere* User-, System- und Wait-Time ausgeben
(in Prozent als Kommazahlen, nicht als Balken).

Dazu ist Folgendes nötig:

- Eine *weitere Strukturvariable* für die CPU-Zeiten,
  in der die Werte im Hauptprogramm in jeder Sekunde *aufsummiert* werden,
  und ein *Zähler* für die *Anzahl* der aufsummierten Zeiten
  (zur Mittelwert-Berechnung).
- Ein *Signal-Handler*, der aus diesen Variablen die *Ausgabe* berechnet (Achtung: Zähler und Zeitwerte sind **`int`**, wie gibst du das Mittel *als Kommazahl* aus?) und dann das Programm *beendet*.
- Ein **`sigaction`**-Aufruf im Hauptprogramm vor der Endlosschleife,
  der diesen Signal-Handler für **`SIGINT`** einrichtet.