Praktikum 3: Systemaufrufe, Zeit- und Ressourcenverbrauch

Die Lernziele in diesem Praktikum sind

- Verwendung von Systemaufrufen.
- Kommandoaufrufe aus einem Programm.
- Analyse von CPU-Zeiten unterschiedlicher Programme.

Aufgabe 1
Schreiben Sie ein C++-Programm namens `zeiten` mit dessen Hilfe sich die Laufzeit und die CPU-Zeit eines beliebigen Programms messen lässt. Bei der CPU-Zeit soll zudem unterschieden werden zwischen CPU-Zeit im Benutzermodus und CPU-Zeit im Systemmodus.

Der Aufruf des zu messenden Programms soll als Kommandozeilenargument des Mess-Programms angegeben werden.

Bemerkung: Das Messprogramm wird im Laufe der weiteren Praktikumsaufgaben noch einige Male verwendet.

Beispiel: Verwendung des Programms und Ausgabe
`./zeiten vi`

Das Programm `zeiten` startet nun den `vi` und misst die dabei anfallenden Zeiten. Nach Beendigung des `vi` gibt `zeiten` die gemessenen Werte wie folgt aus:

```
Kommando: vi
Laufzeit: 24.450000 sek
User-Zeit: 0.010000 sek
System-Zeit: 0.010000 sek
```

Hinweise zur schrittweisen Entwicklung des Programms

1. Mit Hilfe des Systemaufrufs `system()` kann ein beliebiges Kommando aus einem C++-Programm heraus gestartet werden.

   ```
   #include<cstdlib>
   int system(const char *string);
   ```

   `system()` startet einen Kind-Prozess mit der in *string* angegebenen Programmname-/Parameterkonstellation.

2. Alle Unixsysteme, unabhängig von der Zeitzone, verwenden intern die Unixzeit. Die Unixzeit ist die Zahl der vergangenen Sekunden seit Donnerstag, dem 1. Januar 1970, 00:00 Uhr, UTC (vormals GMT). Dieses Startdatum wird auch als The Epoch (siehe Epoche) bezeichnet.

```
#include <sys/time.h>
int gettimeofday (struct timeval *tv, struct timezone *tz);
```

Der Systemaufruf `gettimeofday()` liefert die Kalenderzeit in einem Struct vom Typ `timeval` auf den *tv* zeigt. Der Parameter *tz* sollte immer `NULL` sein.

Der Struct `timeval` ist wie folgt definiert:

```
struct timeval {
    time_t      tv_sec;     /* Seconds since 00:00:00,
                               1 Jan 1970 UTC */
    long        tv_usec;    /* Additional microseconds */
};
```

Sie können alternativ auch die C++ Bibliothek `<chrono>` verwenden, informieren Sie sich dazu in der C++-Referenz.

3. Um die Prozessorzeiten im Benutzer- und im Systemmodus zu erhalten, können Sie den Systemaufruf `getrusage()` verwenden, der über die benutzten Ressourcen Auskunft gibt.

   ```
   #include <sys/resource.h>
   int getrusage(int who, struct rusage *res_usage);
   ```

   Der Paramenter `who` spezifiziert für welchen Prozess die Information angefordert wird. Er hat einen der Werte `RUSAGE_SELF` (Information über den aufruffenden Prozess.)
   `RUSAGE_CHILDREN` (Informationen über Kind-Prozesse, die beendet sind und auf die gewartet wurde.)
   `RUSAGE_THREAD` (Linux-spezifisch für Informationen über den aufrufenden Thread.)

   Der Parameter `res_usage` ist ein Zeiger auf ein Struct vom Typ `rusage`, definiert als:

   ```
   struct rusage {
       struct timeval ru_utime;      /* User CPU time used */
       struct timeval ru_stime;      /* System CPU time used */
       // ...
   }
   ```

   In dem Struct finden Sie den bereits bekannten Struct `timeval` für die CPU-Zeiten im Benutzermodus und im Systemmodus.

Aufgabe 2

Testen Sie das Programm `zeiten` anhand der Messung eines CPU-intensiven Programms (z.B. ein Programm, welches 10.000.000 Gleitkommazahlen addiert), eines E/A-intensiven Programms (z.B. eines Editiervorgangs mittels vi) und einem Programm, das nichts tut (z.B. sleep-Aufruf). Sie finden entsprechende Beispielprogramme im `export`-Verzeichnis der Vorlesung.

Was ist Ihre Beobachtung bei der Zeiten-Messung?