Übungen zu Systemnahe Programmierung in C

Abschnitt 13.2: Hands-On (Stoppuhr)

20.07.2020

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

Hands-on: Stoppuhr

```
$ ./stoppuhr
Press Ctrl+C (SIGINT) to start and stop
^CStarted...
1 sec
2 sec
3 sec
4 sec
^CStopped.
Duration: 4 sec 132 msec
```

- Ablauf:
  - Stoppuhr startet durch `SIGINT` Signal
    - → Gibt jede Sekunde die bisherige Dauer aus (Format: “3 sec”)
  - Stoppuhr stoppt bei weiterem `SIGINT` und gibt Dauer aus
    - → Gibt Gesamtdauer inkl. Millisekunden aus (Format: “4 sec 132 msec”)
    - → Beendet sich anschließend
- Verwendet intern `SIGALRM` und `alarm(2)`
- Schutz kritischer Abschnitte beachten

Wiederholung Signale

1. Signalhandler installieren: `sigaction(2)`

```
01 struct sigaction act;
02 act.sa_handler = SIG_DFL; // Handlersignatur: void f(int signum)
03 act.sa_flags = SA_RESTART;
04 sigemptyset(&act.sa_mask);
05 sigaction(SIGINT, &act, NULL);
```

2. Signale blockieren/deblockieren: `sigprocmask(2)`

```
01 sigset_t set;
02 sigemptyset(&set);
03 sigaddset(&set, SIGUSR1);
04 sigprocmask(SIG_BLOCK, &set, NULL); /* Blockiert SIGUSR1 */
05 // kritischer Abschnitt
06 sigprocmask(SIG_UNBLOCK, &set, NULL); /* Deblockiert SIGUSR1 */
```

Wiederholung Signale

3. Auf Signale warten: `sigsuspend(2)`

```
01 sigprocmask(SIG_BLOCK, &set, &old); /* Blockiert Signale */
02 while(event == 0){
03   sigsuspend(&old); /* Wartet auf Signale */
04 }
05 sigprocmask(SIG_SETMASK, &old, NULL); /* Deblockiert Signale */
```

Alarme mit `setitimer` (1)

- Zeitgeber mittels `setitimer(2)` konfigurieren

```
01 #include <sys/time.h>
02
03 int setitimer(int which, const struct itimerval *new_value,
04               struct itimerval *old_value);
```

- Parameter:

  `which` Hier: `ITIMER_REAL` (Physikalische Zeit)
  `new_value` Zu setzende Konfiguration
  `old_value` Zum Auslesen der vorherigen Konfiguration

- `SIGALRM`: Timer ist abgelaufen bzw. Alarm eingetreten
  → Standardbehandlung: Programm beenden
  → Eigenen Signalhandler installieren

Alarme mit `setitimer` (2)

- Strukturen zur Konfiguration

```
01 struct timeval {
02     time_t      tv_sec;         /* seconds */
03     suseconds_t tv_usec;        /* microseconds */
04 };
```

Beschreibt Zeitintervall mit `tv_sec` s und `tv_usec` μs

```
01 struct itimerval {
02     struct timeval it_interval; /* Interval for periodic timer */
03     struct timeval it_value;    /* Time until next expiration */
04 };
```

Erster Alarm nach Intervall `it_value`
danach periodischer Alarm mit Intervall `it_interval`

- Besondere Werte

`it_interval = {0, 0}` Singleshot Alarm
`it_value = {0, 0}` Alarm abbrechen