

Echtzeitbetriebssysteme

Weiter geht's mit Zeitdiensten...



Echtzeitbetriebssysteme

Zeitdienste

Zeiten spielen in einem Echtzeitsystem eine wesentliche Rolle

Aufgaben von Zeitdiensten

- Zyklische Interruptgenerierung
- Zeitmessung
- Watchdog (Zeitüberwachung)
- Zeitsteuerung für Dienste

Werkzeuge zur Zeitbestimmung (Zeitgeber)

- Realzeituhren
- Timer



Echtzeitbetriebssysteme

Zeitdienste Zyklische Interruptgenerierung

Beispiel 1: Ableitung der Systemzeit (Ticks) in UNIX, OS9:

Timer im System generiert zyklisch (z.B. alle 10 ms) einen Interrupt

Zugehörige Interrupt Service Routine:

- **Scheduler aufrufen**
- **zeitabhängige Systemdienste (*Weckaufrufe*) bearbeiten**
- **Softwaretimer**

Genauigkeit der zeitabhängigen Systemdienste hängt von dieser Zeitbasis ab.

Beispiel 2:

Zyklische Bedienung von „dummen“ Interface-Karten (meist wird Takt von Bus-Controller-HW generiert)



Echtzeitbetriebssysteme

Zeitdienste Zeitmessung

Oft vorkommende Aufgabe in der Automatisierungstechnik

Beispiele:

- **Zuordnung hochgenauer Zeitstempel zu Ereignissen**
 - z.B. in Sensoren bei Teilchenbeschleuniger-Experimenten
- **Berechnen von Geschwindigkeiten über Differenzzeitmessung (Geschwindigkeitsmessung von Fahrzeugen mit Lichtschranken)**



Echtzeitbetriebssysteme

Zeitdienste Watchdog (Zeitüberwachung)

**Sicherheitsrelevante Funktionalität in Echtzeitsystemen:
Überwachung der Einhaltung von Echtzeitbedingungen**

- **Zeitüberwachung einfacher Dienste (z.B. Ausgabe von Daten an eine Prozessperipherie, etwa Roboterarm über Bus)**
- **Zeitüberwachung einzelner Systemkomponenten (z.B. Gelenkregler bei Roboterarmen)**
- **Zeitüberwachung des gesamten Systems**
- **Überwachung kritischer User-Interaktion**
 - **„Totmann“-Taste in Lokomotiven**



Echtzeitbetriebssysteme

Zeitdienste Watchdog (Zeitüberwachung)

Mechanismus: Watchdog

- System oder Komponente muss im regelmäßigen Abstand einen Rückwärtszähler (Timer) zurücksetzen (d.h. Startwert neu setzen)
- Ist das System in einem undefinierten Zustand und kommt nicht mehr dazu, den Zähler zurückzusetzen, zählt dieser bis Null
- Zähler löst bei Zählerstand 0 Interrupt aus
 - Alarm/Fehlermeldung
 - Systemreset (ISR oder HW-Signal löst CPU-Reset aus)



Echtzeitbetriebssysteme

Zeitdienste Watchdog (Zeitüberwachung), Realisierung OS9 API

- Watchdogumgebung vorbereiten

```
error_code os_intercept ( void (*signal_empfang) () , _glob_data) ;
```

- Watchdog löschen bzw. abbrechen

```
error_code _os_alarm_delete( alarm_id alarm_id) ;
```

- Watchdog erzeugen und starten

```
error_code _os_alarm_set(  
alarm_id *alarm_id,      /*out: ID of watchdog alarm */  
signal_code mysignal, /*in: desired watchdog signal number >255*/  
u_int32 time);          /*in: watchdog time in time slices */
```

- Watchdog Aktion bei Auslöse

```
void signal_empfang (signal_code signal)  
{  
    if(signal == mysignal) /*watchdog hit!!! */ ;  
    _os_rte();  
}
```



Echtzeitbetriebssysteme

Zeitdienste Watchdog (Zeitüberwachung), OS9 API

```
volatile int watch_mark=0;
void mysig (signal_code sig){
if(sig == 300) watch_mark=1 /*Watchdog hit: just mark it happened 10
                             TICs are over */;

_os_rte();
}

int main (void) {
alarm_id alm_id;
_os_intercept(msig,_glob_data);
_os_alarm_set( &alm_id, 300, 10);/* Watchdog: signal 300
                                   after 10 TICS*/
maybeSlowRoutine (); /* zu überwachende Funktion */
_os_alarm_delete(alm_id); /* safe to delete always */
if(watch_mark == 1) printf("Watchdog hit! too late\n");
exit(0);
}
```



Echtzeitbetriebssysteme

Zeitdienste Zeitsteuerung für Dienste

Ausführen spezifischer Aufgaben in regelmäßigen oder unregelmäßigen Abständen

Beispiele

- Backups
- Meßwerte erfassen
- Abarbeiten von Timelines (z.B. bei Weltraummissionen)
- Protokolldienste
 - Meist: Proprietäres Protokollieren des Applikationszustands (`println`, `cout`, `printf`,...; ggf. Verbosity Level, selten Systemzeit)
 - BS stellen meist einheitliche, bei RT-OS echtzeitfähige Loggingdienste (z.B. `syslog`) auch für Applikationen/Dienste zur Verfügung, da BS-Abläufe sowieso protokolliert werden
- Zeitabhängiges Starten von Tasks / Threads (z.B. `cron-jobs`)
 - „ Speicher ressourcenschonend“ Start/Stop



Echtzeitbetriebssysteme

Zeitdienste Werkzeuge zur Zeitbestimmung - Zeitgeber

Varianten von Zeitgebern (allgemein)

- **Absolut-Zeitgeber (Uhren, *clocks*)**
- **Relativ-Zeitgeber (Timer, *timer*)**

Uhren und Timer lassen sich auf zwei Arten realisieren:

1. in Hardware und
2. in Software

in Form von

- **Vorwärtszähler**
- **Rückwärtszähler**
 - **repetitive Zähler: Vorgegebener Zählwert wird nach Ablauf (Zählerstand 0 erreicht) selbständig neu geladen**
 - **single shot Zähler: expliziter Neustart erforderlich**



Echtzeitbetriebssysteme

Zeitdienste Zugriffs-Ebenen in einem Realzeitsystem

- **Hardware-Level**
- **Kernel-Level**
- **User-Level**



Echtzeitbetriebssysteme

Zeitdienste, Zugriffs-Ebenen in einem Realzeitsystem

Hardwarelevel → „Single Hardware“

Zeit-bezogene Hardware:

- Echtzeituhr(en) (Absolutzeitgeber)
- Frequenzteiler
- Vor- und (meist)Rückwärtszähler
- Watchdogtimer (mit Reset-Signal bei Ablauf, vermeidet ISR)

Absolutzeitgeber:

- Stellen Absolutzeit (Tag, Monat, Jahr, Stunde, Minute, Sekunde, Millisekunde) bereit (z.B. UNIX: in Sekunden seit 1.1.1970)
- Meist batteriegepuffert, Weiterzählen der Uhrzeit nach Abschalten des Stroms
- Absolutzeiten sind bei eingebetteten Systemen nicht immer notwendig
- Wichtig: Genaue Absolutzeitbasis bei verteilten Realzeitsystemen
 - Probleme: Zeitzonen, Sommer/Winterzeit, Zeitdrift



Echtzeitbetriebssysteme

Zeitdienste

Kernellevel „Multi-Hardware“ → „virtuelle Zeitgeber“

Zeit-Basis: zyklische Timerinterrupts

Zeitgesteuerte Dienste im Betriebssystemkern:

- **Gerätetreiber können Tasks für wählbare Zeitdauer in den Zustand „wartend“ versetzen**
- **Module können zyklisch Funktionen aufrufen lassen**
- **Module können einmalig (zu einem Relativzeitpunkt) Funktionen aufrufen lassen**

Vorsicht bei „Real“-Zeit im Kernel (Sekunden statt Ticks):

- **Der Abstand zwischen zwei Timerticks kann von System zu System variieren**

Problem Zählerüberlauf (in Linux nach 417 Tagen: $55\text{Bit} * 1\text{ns}$).

- **Zählerüberlauf muss in allen Schichten geprüft werden**
- **Massnahme in aktuellen Linux-Kerneln: Start mit Zeit kurz vor Überlauf!!!**



Echtzeitbetriebssysteme

Zeitdienste

Userlevel → „private Zeitgeber“

- Absolutzeit abfragen mit z.B. `gettimeofday`
- Prozess oder Thread für eine definierte Zeit in den Zustand “wartend” versetzen: `sleep` und `select`
- Dienstschnittstelle: Prozess starten
 - periodisch: `cron`
 - bestimmter (Absolut-) Zeitpunkt: `at`
- In Realzeitsystemen: (z.B. Proprietäre oder POSIX.1-2017 -) Interfaces für (Hardware-)Timer
 - zeitgenauere (bzgl. Auflösung) Steuerung von Prozessen (Micro-Nanosekunden)



Echtzeitbetriebssysteme

Zeitdienste

Kritische Punkte

Kernelebene:

- **Relativzeiten wegen der möglichen Zählerüberläufe**

User-Ebene:

Absolutzeiten, insbesondere in verteilten Echtzeitsystemen, z.B.

- **„Jahr 2000 Problem“ (alle 100 Jahre), wenn eine Jahresangabe nicht in 4, sondern in nur 2 Stellen abgelegt wird**
- **Wahl der Zeitzone**
- **Umschaltung zwischen Winter- und Sommerzeit zum falschen Zeitpunkt.**
- **Falsche Berechnung von Schalttagen (29. Februar).**
- **Zeitdrift**



Echtzeitbetriebssysteme

Zeitdienste, Kritische Punkte

Zeitdrift

Ursachen: Uhren laufen ungenau, ungenaue (oder keine) Systemzeit beim Start

Zeitsynchronisation zwischen Rechnern bzw. der Korrektur von Absolutzeiten

- z.B. Zeitserver versendet Broadcastmessages
- Verarbeitung dieser Messages in Hardware ermöglicht Zeitsynchronisation im Mikrosekundenbereich

Standardisiert: Zeitsynchronisation via NTP (Network Time Protocol).

- Zeitserver stellen die Zeit zur Verfügung, hierarchisch (Stratum) klassifiziert
- Stratum-1 Server: bestimmt Uhrzeit selbst (z.B. von DCF77)
- Stratum-2 Server: bezieht Uhrzeit von Stratum-1 Server(n)
- Statistische Berechnung der Paketlaufzeiten bei der Verteilung der Uhrzeit
- Statistische Berechnung der Uhrzeit aus der Information mehrerer Server



Echtzeitbetriebssysteme

Zeitdienste, Kritische Punkte

Zeitkorrektur

Problematisch:

- **sprunghafte Korrektur von Absolutzeiten!!!**
 - Zeitpunkte entfallen oder sind doppelt!!!

Besser:

- **Abbremsen/Beschleunigen der Systemuhr**

NTP-Systemcall `int adjtimex(struct timex *buf);`

Ähnliches Problem: Rechnerstillstand (z.B. für Wartung)

- **Innerhalb der Downtime geplante Aktionen entfallen**

Abhilfe:

- **Protokollieren durchgeführter Aktionen**
- **Nachholen nicht durchgeführter Aktionen beim Neustart**



Echtzeitbetriebssysteme

Zeitdienste, POSIX.1-2017

Repräsentation von Zeit

Absolutzeit:

- `time_t`: Integer, Sekunden seit 1.1.1970 (27.11.06, 9:40 : 0x456aa3ef);
 - 32Bit Überlauf in ca. 105 Jahren...
 - ABER: z.B. in Millisekunden???
- Aufgelöst in Datumskomponenten

```
struct tm {
    int tm_sec    /* Seconds [0,60] */
    int tm_min    /* Minutes [0,59] */
    int tm_hour   /* Hour [0,23] */
    int tm_mday   /* Day of month [1,31] */
    int tm_mon    /* Month of year [0,11] */
    int tm_year   /* Years since 1900 */
    int tm_wday   /* Day of week [0,6] (Sunday =0) */
    int tm_yday   /* Day of year [0,365] */
    int tm_isdst  /* Daylight Savings flag */
}
```



Echtzeitbetriebssysteme

Zeitdienste, POSIX.1-2017

Repräsentation von Zeit

- **Hochaufgelöst (Nanosekunden) für Absoluttimer**

```
struct timespec {
    time_t tv_sec, /* Seconds */
    long   tv_nsec /* Nanoseconds */
}
```

- **Relativzeiten und Zeitpunkte für Timer**

```
struct itimerspec {
    struct timespec it_interval, /* Intervall */
    struct timespec it_value    /* (Absolut)-Zeitpunkt */
}
```



Echtzeitbetriebssysteme

Zeitdienste, POSIX.1-2017

API für Zeitdienste

- **Absolutzeit**

Wandelt Zeitbeschreibung in Klartext

```
char *asctime(const struct tm *);
```

Liefert aktuelle Zeit in Sekunden seit 1.1.1970

```
time_t time ( time_t *t);
```



Echtzeitbetriebssysteme

Zeitdienste, POSIX.1-2017

API für Zeitdienste

- **Warten (Prozess schlafenlegen)**

Prozess für n Sekunden schlafen legen

```
int sleep ( int secs );
```

Prozess für n Nano-Sekunden schlafen legen (<>OS9)

```
int nanosleep(const struct timespec *request,  
              struct timespec *remainder_after_signal);
```



Echtzeitbetriebssysteme

Zeitdienste, OS9

API für Zeitdienste

- **Warten relativ in TICS oder 1/256 Sekundenzeitraster**

```
error_code _os_sleep(
```

```
u_int32 *ticks, /* Generell: Wakeup durch Signalempfang (wenn  
erlaubt)
```

```
in: ticks = 0 unendlich warten
```

```
in: ticks > 0 warten von mindestens (ticks-
```

```
2) *TICS
```

```
= 1 kein Warten
```

```
= 2 bis zum next TIC warten
```

```
= 3 einen vollen TIC mind. warten
```

```
in: ticks = 0x80000000 + Anz_256th_of_Seconds
```

```
out: nach Wakeup enthält ticks die noch zu  
wartende Restzeit */
```

```
signal_code *signal); /* 0 oder Signalnummer, die wake-up  
verursachte*/
```



Echtzeitbetriebssysteme

Zeitdienste, POSIX.1-2017

API für Zeitdienste

- **Hochauflösende (Absolutzeit-)Uhr (<>OS9)**

Aktuelle Uhrzeit erfragen

```
int clock_gettime(clockid_t clock_id,  
                 struct timespec *tp);
```

Uhrzeit setzen

```
int clock_settime(clockid_t clock_id,  
                 const struct timespec *tp);
```

Auflösung der Uhrzeit erfragen

```
int clock_getres(clockid_t clock_id, struct timespec *res);
```



Echtzeitbetriebssysteme

Zeitdienste, POSIX.1-2017

API für Zeitdienste

- **Hochauflösende (Relativzeit-)Timer (<>OS9)**

Timer erzeugen

```
int timer_create(clockid_t clockid,  
                struct sigevent *evp,  
                timer_t *timerid);
```

Timer löschen

```
int timer_delete(timer_t timerid);
```

Verbleibende Zeit im Timer abfragen

```
timer_gettime(timer_t timerid, struct itimerspec *value);
```

Zeit bis zum Ablauf setzen

```
timer_settime (timer_t timerid, int flags,  
              const struct itimerspec * value,  
              struct itimerspec * ovalue); /*(Restzeit)*/
```



Echtzeitbetriebssysteme

**Einschub: Aufruf von
(Zeit-)Dienstfunktionen
(Systemcall - Interface)
des BS...**



Echtzeitbetriebssysteme

Systemcall-Interface

Absolut unabhängig von Programmiersprachen

Meist gekapselt durch Bibliotheksfunktionen (komfortabler)

Systemcalls erwarten Argumente in Registern oder auf dem Stack

Aufruf per Softwareinterrupt: Assemblerbefehl **INT** bzw. **TRAP** mit einer Exceptionnummer (bei Linux beispielsweise 0x80)

```
write_hello_world:
movl $4,%eax ;           //code fuer "write" syscall
movl $1,%ebx ;           //file descriptor fd (1=stdout)
movl $message,%ecx ;     //Adresse des Textes (buffer)
movl $12,%edx ;          //Laenge des auszugebenden Textes
int $0x80 ;               //SW-Interrupt, Auftrag an das BS
ret
.data
message:
.ascii "Hallo World\n"
```



Echtzeitbetriebssysteme

Systemcall-Interface OS9

Ein C-Aufruf für alle Aufrufe

Gekapselt durch C-Bibliotheksfunktion (komfortabler)

```
error_code error;
i_write_pb pb; /* declare parameter block write*/
pb.cb.code = I_WRITE; /* write command */
pb.cb.param_size = sizeof i_write_pb;
pb.cb.edition = _OS_EDITION; /* fill edition number */
pb.path = 1; /* stdout*/
pb.buffer = "Hallo Welt\n"; /* Textoutput*/
pb.count = strlen("Hallo Welt\n");
if ((error = _oscall(&pb)) == SUCCESS) /*all ok */;
```



Echtzeitbetriebssysteme

Dienste (Services)

BS-Schicht auf User-Ebene: Dienste (Services oder Dämonen), je nach Applikation mehr oder weniger ausgeprägt

Dienste zur Konfiguration beim Systemstart, z.B.

- **Hardware initialisieren (PnP)**
- **Netzwerk konfigurieren (z.B. IP Adresse; DHCP)**
- **Booten (DHCP, TFTP, BOOTP)**
- **Treiber laden**

Protokolldienste

- **Meist: Proprietäres protokollieren des Applikationszustands (printf, cout, printf,...; ggf. Verbosity Level, selten Systemzeit)**
- **Dtrace: „cool“ -> mac os, FreeBSD, NetBSD, QNX**
- **BS stellen meist einheitliche, bei RT-OS echtzeitfähige Loggingdienste (z.B. syslog) auch für Applikationen/Dienste zur Verfügung, da BS-Abläufe sowieso protokolliert werden**



Echtzeitbetriebssysteme

Dienste (Services)

Netzwerkdienste

- telnet (Interaktive Sitzung)
- RPC Remote Procedure Call
- CORBA Common Object Request Broking Architecture
- FTP File Transfer
- HTTP HyperText Transfer Protocol
- NTP: Zeitsynchronisation in einem verteilten Echtzeitsystem
- NFS/SMB: Netzwerklaufwerke

Zeitsteuerung

- Tasks zyklisch oder zu vorgegebenen Zeiten starten („cron jobs“).
zentral verwalt-/steuerbar
 - Ressourcenschonend (Tasks belegen keine Ressourcen [Speicher], die sie belegen würden, wenn sie nur schliefen)

