Events und Mailboxen

Florian Franzmann Martin Hoffmann Tobias Klaus
Peter Wägemann

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
www4.informatik.uni-erlangen.de

1. Dezember 2014

- Events
- Mailbox

Signalisierung von Ereignissen - eCos Event Flags[1]

Grundlagen

- Signale unterstützen Produzent-Konsument Muster
- Ein Thread/DSR kann ein Ereignis (z.B. Tastendruck) signalisieren auf das ein konsumierender Thread wartet
- Umsetzung: 32-bit Integer $\rightarrow$ 32 Einzelsignale pro Flag
  - Ein Flag erlaubt somit $2^{32} - 1$ Signalkombinationen
  - Threads können ein Signalmuster blockierend warten, bzw. pollen
- Achtung: Flags zählen keine Ereignisse! (vgl. HW-Interrupts)

[1] `http://ecos.sourceware.org/docs-latest/ref/kernel-flags.html`

Signalisierung von Ereignissen - eCos Event Flags[2]

API

- Produzenten/Konsumenten teilen sich eine Flagobjekt
- Dieses wird von der Anwendung bereitgestellt (vgl. Alarmobjekt)
- Flagobjekt muss initialisiert werden:

```
cyg_flag_init(cyg_flag_t* flag)
```

- Signal(e) im Flag setzen:

```
cyg_flag_setbits(cyg_flag_t* flag, cyg_flag_value_t value)
```

- Bzw. zurücksetzen:

```
cyg_flag_maskbits(cyg_flag_t* flag, cyg_flag_value_t value)
```

- Auf Signal warten/pollen:

```
cyg_flag_value_t cyg_flag_wait/poll(cyg_flag_t* flag,
cyg_flag_value_t pattern, cyg_flag_mode_t mode);
```

[2] http://ecos.sourceware.org/docs-latest/ref/kernel-flags.html

Signalisierung von Ereignissen - eCos Event Flags[3]

API - Konsumieren von Signalen

- `cyg_flag_value_t pattern` setzt gewünschte Signalkombination
- `cyg_flag_mode_t` legt Weckmuster fest

  CYG_FLAG_WAITMODE_AND Alle konfigurierten Signale müssen aktiv sein; Sie bleiben nach dem Aufwachen gesetzt.

  CYG_FLAG_WAITMODE_OR Mindestens eines der konfigurierten Signale muss aktiv sein; Alle Signale bleiben nach dem Aufwachen gesetzt.

  CYG_FLAG_WAITMODE_OR | CYG_FLAG_WAITMODE_CLR Mindestens eines der konfigurierten Signale muss aktiv sein; Alle gesetzten Signale werden nach dem Aufwachen gelöscht.

[3] `http://ecos.sourceware.org/docs-latest/ref/kernel-flags.html`

Signalisierung von Ereignissen - eCos Event Flags[4]

Beispiel

```
cyg_flag_t flag0;

void my_dsr(cyg_vector_t v, cyg_ucount32 c, cyg_addrword_t d){
  cyg_flag_setbits(&flag0, 0x02);
}

void user_thread(cyg_addr_t data){
  while(1){
    cyg_flag_wait(&flag0, 0x22,
      CYG_FLAG_WAITMODE_OR | CYG_FLAG_WAITMODE_CLR);
    printf("Event!\n");
  }
}

void cyg_user_start(){
...
  cyg_flag_init(&flag0);
...
}
```

[4] http://ecos.sourceware.org/docs-latest/ref/kernel-flags.html

Versenden von Nachrichten - eCos Mail Boxes[5]

Grundlagen

- Zwischen Threads können Nachrichten versendet werden
- Ein Konsument erzeugt einen Briefkasten (mailbox) einer gewissen Größe (Default: 10)
- Produzenten können Nachrichten dort ablegen
  - Inhalt: Zeiger auf beliebige Datenstruktur
  - Konsument kann auf Nachrichtenempfang blockieren
  - Produzent blockiert, falls Briefkasten voll
  - Aber auch nicht-blockierende Aufrufvarianten

[5] http://ecos.sourceware.org/docs-latest/ref/kernel-mail-boxes.html

Versenden von Nachrichten - eCos Mail Boxes[6]

API - Übersicht

- Mailbox anlegen:

```
cyg_mbox_create(cyg_handle_t* handle, cyg_mbox* mbox);
```

- Nachricht verschicken:

```
cyg_bool_t cyg_mbox_put(cyg_handle_t mbox, void* item);
```

- Nachricht empfangen: `void* cyg_mbox_get(cyg_handle_t mbox);`

- Empfang *und* Versand können blockieren.
- `*try*`-Versionen: Würde ich blockieren?
- `*timed*`-Versionen: Blockieren, aber nur für bestimmte Zeit.
- → Selbststudium!

[6] http://ecos.sourceware.org/docs-latest/ref/kernel-mail-boxes.html

Versenden von Nachrichten - Beispiel

- Initialisierung

```
static cyg_handle_t mailbox_handle;
static cyg_mbox     mailbox;

void cyg_user_start(void) {
  cyg_mbox_create(&mailbox_handle, &mailbox);
  ... }
```

- Produzent (Sender)

```
void producer_entry(cyg_addrword_t data) {
  ...
  cyg_mbox_put(mailbox_handle, &my_message);
  ... }
```

- Konsument (Empfänger)

```
void consumer_entry(cyg_addrword_t data) {
  ...
    void *message = cyg_mbox_get(mailbox_handle);
  ... }
```