

Echtzeitsysteme - Übungen

Betriebsmittelprotokolle

Sommersemester 2024

Eva Dengler Peter Wägemann

Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)

Lehrstuhl Informatik 4 (Systemsoftware)

<https://sys.cs.fau.de>



Lehrstuhl für Informatik 4
Systemsoftware



Friedrich-Alexander-Universität
Technische Fakultät

Zugriffskontrolle

Zugriffskontrolle in eCos

Hinweise zu Aufgabe 7

Konkurrenz und Koordination

- Betriebsmittelarten \rightsquigarrow einseitige/mehrseitige Synchronisation
- Konkurrenz \rightsquigarrow Vergabe/Freigabe (P/V)
- Konflikt \rightsquigarrow Streit um begrenzte bzw. unteilbare Betriebsmittel (BM)

Synchronisation

\rightsquigarrow Nichtfunktionale Eigenschaft

- Prioritätsumkehr \rightsquigarrow kontrolliert vs. unkontrolliert
- Verklemmung (Deadlock)

Synchronisationsprotokolle

- Verdrängungssteuerung
- Prioritätsvererbung & Prioritätsobergrenzen
- Blockadezeit \rightsquigarrow direkt vs. durch Vererbung

Verdrängungssteuerung (NPCS)

- Unterbindet Verdrängung im kritischen Abschnitt
- Blockadezeit $\rightsquigarrow \max(cs)$
- + Deadlock Prevention \rightsquigarrow Kein „hold and wait“
- + Kein à priori Wissen nötig
- + Einfach; gut für wenige BM
- Verzögerung höher priorer Jobs ohne Konflikt

Prioritätsvererbung (Priority Inheritance)

- Priorität zeitweise erhöhen (von höherprioreren Fäden erben)
- Blockadezeit $\rightsquigarrow \min(n, k) \cdot \max(cs)$
- + Verbessert Verzögerung von Jobs ohne Konflikt
- Transitive Blockierung möglich; Deadlocks möglich

Prioritätsbergrenzen (Priority Ceiling Protocol)

- Variante der PV mit Prioritätsbergrenzen
- BM-Obergrenze $\rightsquigarrow \max(p_i)$ aller Jobs die das BM nutzen
- Systemobergrenze \rightsquigarrow höchstprioreres, belegtes BM (zur *Laufzeit*)
- Betriebsmittelvergabe \rightsquigarrow BM-Graph (lineare Ordnung)
- Blockadezeit $\rightsquigarrow \max(cs)$ (wie NPCCS)
- + Deadlock Avoidance \rightsquigarrow Kein „cyclic wait“
- + Vermeidet transitive Blockierung
- à priori Wissen nötig; aufwendig; avoidance blocking

Stackbasierte Prioritätsbergrenzen

- Vereinfachung des klassischen PCP \rightsquigarrow Stack-based PCP
- Implementiert z. B. in OSEK; Keine Selbstsuspendierung erlaubt!

Zugriffskontrolle

Zugriffskontrolle in eCos

Hinweise zu Aufgabe 7

Kerneldatenstrukturen durch Sperren des Schedulers geschützt

→ **Big Kernel Lock (BKL)**

- **Sperre:** `void cyg_scheduler_lock(void);`
 - Sofortiges Anhalten des Scheduling
 - Verzögerung der DSR-Ausführungen
 - **ISRs werden weiterhin zugestellt!**
- **Freigabe:** `void cyg_scheduler_unlock(void);`
 - Sofortige Abarbeitung angelaufener DSRs
- Alle Systemaufrufe werden per NPCS synchronisiert
- Anwendungen sollten Mutexe, Semaphore, etc. nutzen

¹<http://ecos.sourceware.org/docs-latest/ref/kernel-schedcontrol.html>

■ Initialisierung

```
void cyg_mutex_init(cyg_mutex_t* mutex);
```

■ Protokoll auswählen:

```
void cyg_mutex_set_protocol(cyg_mutex_t* mutex,  
                           enum cyg_mutex_protocol protocol);
```

- CYG_MUTEX_NONE keine Prioritätsvererbung
- CYG_MUTEX_INHERIT erbe Priorität des aktuellen Inhabers
- CYG_MUTEX_CEILING erbe Prioritätsobergrenze

■ nur bei CYG_MUTEX_CEILING: Prioritätsobergrenze setzen

```
void cyg_mutex_set_ceiling(cyg_mutex_t* mutex,  
                           cyg_priority_t priority);
```

²<http://ecos.sourceware.org/docs-latest/ref/kernel-mutexes.html>

■ Mutex belegen

```
cyg_bool_t cyg_mutex_lock(cyg_mutex_t* mutex);
```

Rückgabewert

- true falls Belegen erfolgreich
- false sonst

■ Mutex freigeben:

```
void cyg_mutex_unlock(cyg_mutex_t* mutex);
```

```
1  static cyg_mutex_t s_mutex;
2
3  void cyg_user_start(void) {
4      // Mutex initialisieren
5      cyg_mutex_init(&s_mutex);
6
7      // Protokoll auswaehlen
8      cyg_mutex_set_protocol(&s_mutex, CYG_MUTEX_CEILING);
9
10     // Prioritaetsobergrenze festlegen
11     cyg_mutex_set_ceiling(&s_mutex, 3);
12
13     // Tasks, Alarme etc.
14 }
15
16 void task_entry(cyg_addrword_t data) {
17     cyg_mutex_lock(&s_mutex); // auf Freigabe warten
18     // kritischer Abschnitt
19     cyg_mutex_unlock(&s_mutex); // Mutex freigeben
20 }
```

Zugriffskontrolle

Zugriffskontrolle in eCos

Hinweise zu Aufgabe 7

Aufgabensysteme

1. 3 Aufgaben, 1 Betriebsmittel
2. 4 Aufgaben, 3 Betriebsmittel
3. 3 Aufgaben, 2 Betriebsmittel

Problematische Konstellationen für einzelne Vergabeprotokolle

- Deadlocks
- Pathfinder-Beispiel
- Transitive Blockierung

Implementierung von 1–3, Basisaufgabe:

- `aufgabe_1.c` \rightsquigarrow Verdrängungssteuerung
- `aufgabe_2.c` \rightsquigarrow Prioritätsvererbung
- `aufgabe_3.c` \rightsquigarrow Prioritätsobergrenzen

(R, 3, 4)

- R: verwendetes Betriebsmittel
- 3: relativer Anforderungszeitpunkt in ms
- 4: Zeit, die Betriebsmittel gehalten wird in ms

Beispiel: Task mit WCET von 9 ms

```
1 void task() {  
2     non_critical_work(); // 3 ms  
3  
4     cyg_mutex_lock(&R); // acquire  
5     critical_work(); // 4 ms  
6     cyg_mutex_unlock(&R); // release  
7  
8     non_critical_work(); // 2 ms  
9 }
```