Extended Scope

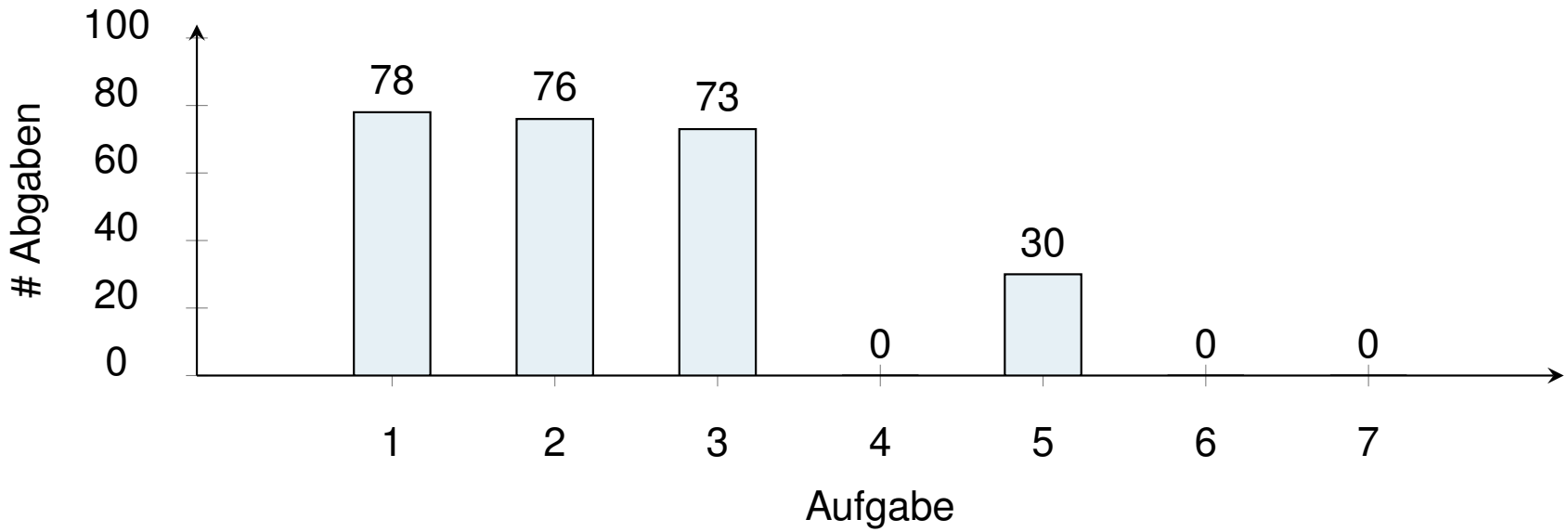
Events und Mailboxen

Florian Franzmann Tobias Klaus Florian Korschin
Florian Schmaus Peter Wägemann

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
https://www4.cs.fau.de

13. Januar 2016

Ein wenig Statistik

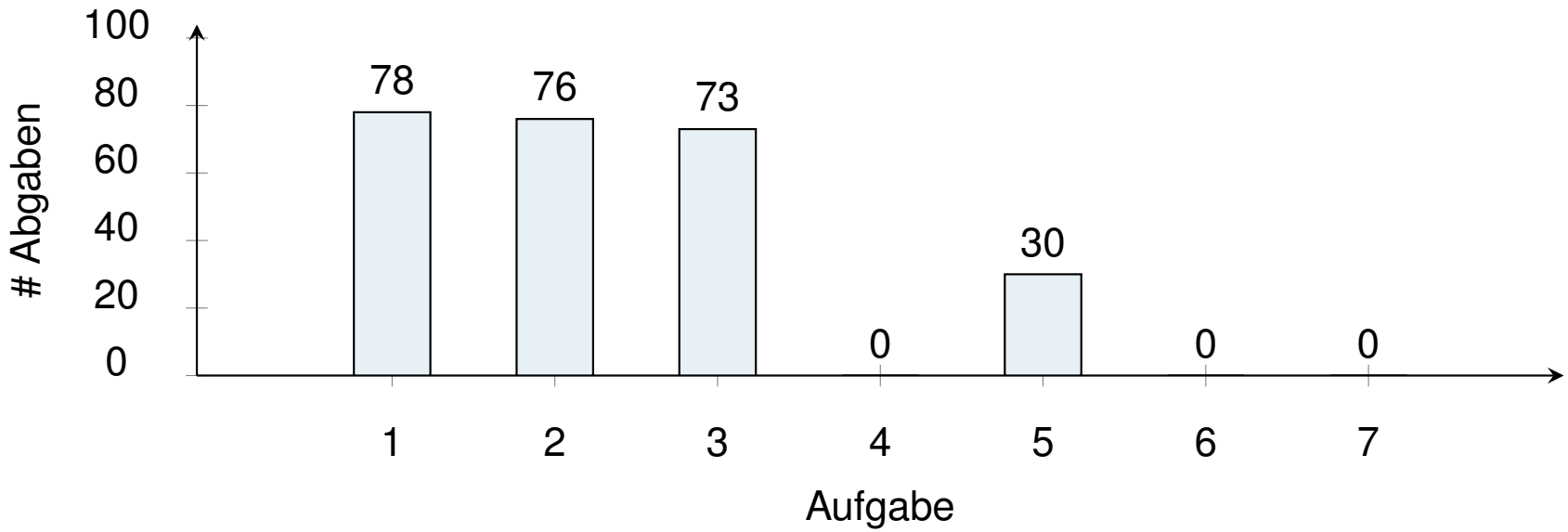


Teilnehmerstatistik

- **104** zu Beginn des Semesters angemeldet, **76** unter dem Semester
- Mehr als doppelt so viele Teilnehmer wie letztes Jahr!

↝ **Über 73 % sind dabei geblieben! ☺**

Ein wenig Statistik



Teilnehmerstatistik

- **104** zu Beginn des Semesters angemeldet, **76** unter dem Semester
- Mehr als doppelt so viele Teilnehmer wie letztes Jahr!

⇝ **Über 73 % sind dabei geblieben!** ☺

Organisatorisches

Evaluation der Veranstaltung

- Eure Meinung (Lob/Kritik) ist uns wichtig!
- Eure Rückmeldung hat Konsequenzen (z.B. Folien-Redesign)

→ Bitte evaluiert Vorlesung und Übungen

⚠ Typische Rückläuferquote $\mapsto 2-10\%$

- Zu wenig für eine sinnvolle Einschätzung
- Aber: Typische Rückläuferquote in EZS $\mapsto 60-80\%$

Motivationsanreiz zur Evaluation

- **Traditionell**: Kaffee und Kekse in der letzten Vorlesung
- **Feste Bedingung**: $\geq 60\%$ der ausgegebenen TANs werden evaluiert!

Organisatorisches

Evaluation der Veranstaltung

- Eure Meinung (Lob/Kritik) ist uns wichtig!
- Eure Rückmeldung hat Konsequenzen (z.B. Folien-Redesign)

→ Bitte evaluiert Vorlesung und Übungen

⚠ Typische Rückläuferquote $\mapsto 2-10\%$

- Zu wenig für eine sinnvolle Einschätzung
- Aber: Typische Rückläuferquote in EZS $\mapsto 60-80\%$

Motivationsanreiz zur Evaluation

- **Traditionell**: Kaffee und Kekse in der letzten Vorlesung
- **Feste Bedingung**: $\geq 60\%$ der ausgegebenen TANs werden evaluiert!

Organisatorisches

Evaluation der Veranstaltung

- Eure Meinung (Lob/Kritik) ist uns wichtig!
- Eure Rückmeldung hat Konsequenzen (z.B. Folien-Redesign)

→ Bitte evaluiert Vorlesung und Übungen

Typische Rückläuferquote $\mapsto 2-10\%$

- Zu wenig für eine sinnvolle Einschätzung
- Aber: Typische Rückläuferquote in EZS $\mapsto 60-80\%$

Motivationsanreiz zur Evaluation

- **Traditionell**: Kaffee und Kekse in der letzten Vorlesung
- **Feste Bedingung**: $\geq 60\%$ der ausgegebenen TANs werden evaluiert!

Organisatorisches

Evaluation der Veranstaltung

- Eure Meinung (Lob/Kritik) ist uns wichtig!
- Eure Rückmeldung hat Konsequenzen (z.B. Folien-Redesign)

→ Bitte evaluiert Vorlesung und Übungen

Typische Rückläuferquote $\mapsto 2-10\%$

- Zu wenig für eine sinnvolle Einschätzung
- Aber: Typische Rückläuferquote in EZS $\mapsto 60-80\%$

Motivationsanreiz zur Evaluation

- **Traditionell**: Kaffee und Kekse in der letzten Vorlesung
- **Feste Bedingung**: $\geq 60\%$ der ausgegebenen TANs werden evaluiert!

Organisatorisches

Evaluation der Veranstaltung

- Eure Meinung (Lob/Kritik) ist uns wichtig!
- Eure Rückmeldung hat Konsequenzen (z.B. Folien-Redesign)

→ Bitte evaluiert Vorlesung und Übungen

Typische Rückläuferquote $\mapsto 2-10\%$

- Zu wenig für eine sinnvolle Einschätzung
- Aber: Typische Rückläuferquote in EZS $\mapsto 60-80\%$

Motivationsanreiz zur Evaluation

- **Traditionell**: Kaffee und Kekse in der letzten Vorlesung
- **Feste Bedingung**: $\geq 60\%$ der ausgegebenen TANs werden evaluiert!

Übersicht

1. Organisatorisches
   - Semesterrückblick
   - Evaluierung
2. Rekapitulation
   - Rangfolgen
3. Ereignisse in eCos
   - Events
   - Mailbox

Rekapitulation der Vorlesung

Rangfolge

- Abhängigkeit von Kontrollfluss $\rightsquigarrow$ Reihenfolge
- oft in Datenabhängigkeiten begründet
  - Produzent/Konsument Verhältnis
  - Konsumierbare Betriebsmittel
  - begrenzte Puffer
- Hinweis auf unterschiedliche zeitliche Domänen!

Kausalordnung

- Relation: Ursache, Wirkung, Nebenläufigkeit
- Nebenläufigkeit vs. Gleichzeitigkeit
- Abhängigkeits- und Aufgabengraphen

Rekapitulation der Vorlesung

Rangfolge

- Abhängigkeit von Kontrollfluss $\rightsquigarrow$ Reihenfolge
- oft in Datenabhängigkeiten begründet
  - Produzent/Konsument Verhältnis
  - Konsumierbare Betriebsmittel
  - begrenzte Puffer
- Hinweis auf unterschiedliche zeitliche Domänen!

Kausalordnung

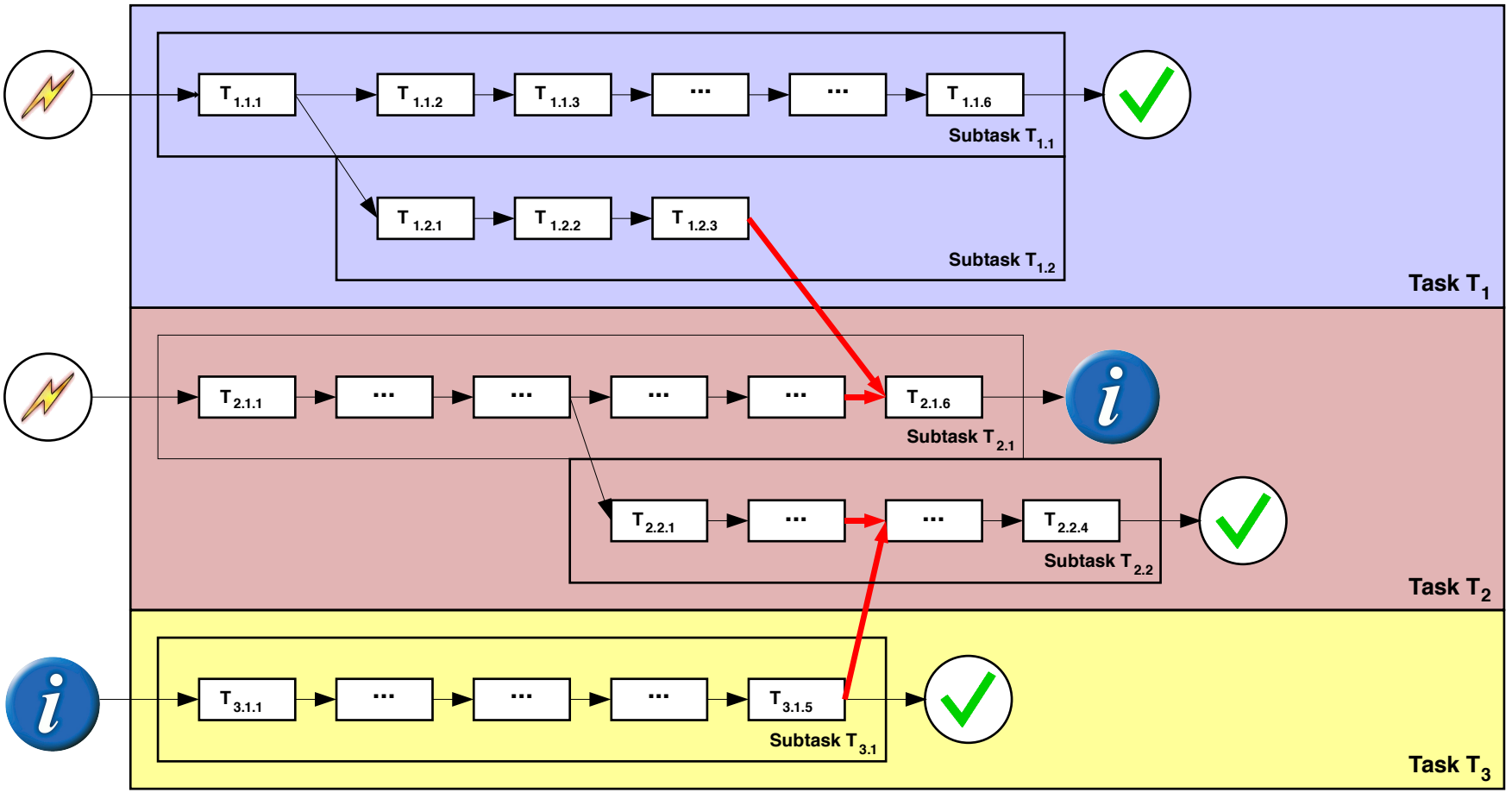
- Relation: Ursache, Wirkung, Nebenläufigkeit
- Nebenläufigkeit vs. Gleichzeitigkeit
- Abhängigkeits- und Aufgabengraphen

Rekapitulation der Vorlesung (Forts.)

Koordinierung

- Unnötig $\rightsquigarrow$ Rangfolge egal
  - Neuester Wert ist ausreichend
- Durch Einplanung $\rightsquigarrow$ analytische Verfahren
  - periodische Aufgaben $\rightsquigarrow$ Passende Raten!!!
  - Ablauftabelle, Phasenversatz
  - Keine Kontrolle zur Laufzeit
- Durch Kooperation $\rightsquigarrow$ konstruktive Verfahren
  - periodische und nicht-periodische Aufgaben
  - Synchronisation $\rightsquigarrow$ Vielzahl von Möglichkeiten
  - in zeitgesteuerten Systemen unmöglich!

Rekapitulation der Vorlesung (Forts.)



Gerichtete Abhängigkeiten: UND, ODER und zeitliche Abhängigkeiten

Übersicht

1 Organisatorisches
- Semesterrückblick
- Evaluierung

2 Rekapitulation
- Rangfolgen

3 Ereignisse in eCos
- Events
- Mailbox

eCos-Event-Flags[1]

Signalisieren von Ereignissen

- Signale unterstützen *Produzent-Konsument Muster*
- Thread/DSR *signalisiert* Ereignis (z. B. Tastendruck)
  . . . konsumierender Thread *wartet*
- Umsetzung: 32-bit Integer $\leadsto$ 32 *Einzelsignale* pro Flag
  - Ein Flag erlaubt somit $2^{32} - 1$ Signalkombinationen
  - Threads können auf ein Signalmuster blockierend warten oder pollen

Achtung:

Flags zählen keine Ereignisse! (vgl. HW-Interrupts)

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

eCos-Event-Flags

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

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

- Signal(e) im Flag setzen:

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

- Bzw. zurücksetzen:

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

- Auf Signal warten/pollen:

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

eCos-Event-Flags

- `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.

eCos-Event-Flags

```
1  static cyg_flag_t flag0;

2  void my_dsr(cyg_vector_t v,
3              cyg_ucount32 c,
4              cyg_addrword_t d){
5    cyg_flag_setbits(&flag0, 0x02);
6  }

7  void user_thread(cyg_addr_t data){
8    while(true) {
9      cyg_flag_wait(&flag0, 0x22,
10                   CYG_FLAG_WAITMODE_OR | CYG_FLAG_WAITMODE_CLR);
11     printf("Event!\n");
12   }
13 }

14 void cyg_user_start(void){
15   ...
16   cyg_flag_init(&flag0);
17   ...
18 }
```

eCos-Mailboxes[2]

- Zwischen Threads können *Nachrichten* versendet werden
- Konsument erzeugt einen Briefkasten (mailbox) fester Größe
- Produzenten legt Nachrichten dort ab
  - Inhalt: Zeiger auf beliebige Datenstruktur
  - Konsument kann auf *Nachrichtenempfang* blockieren
  - Produzent blockiert, falls Briefkasten *voll*
  - Aber auch *nicht-blockierende* Aufrufvarianten

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

Versenden von Nachrichten - eCos Mail Boxes[3] API - Übersicht

- Mailbox anlegen:

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

- Nachricht verschicken:

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

- Nachricht empfangen:

```
3 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!

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

Versenden von Nachrichten – Beispiel

- Initialisierung:

```
1  static cyg_handle_t mailbox_handle;
2  static cyg_mbox     mailbox;
3  void cyg_user_start(void) {
4          cyg_mbox_create(&mailbox_handle, &mailbox);
5          ...
6  }
```

- Produzent (Sender):

```
1  void producer_entry(cyg_addrword_t data) {
2          ...
3          cyg_mbox_put(mailbox_handle, &my_message);
4           ...
5  }
```

- Konsument (Empfänger):

```
1  void consumer_entry(cyg_addrword_t data) {
2          ...
3          void *message = cyg_mbox_get(mailbox_handle);
4          ...
5  }
```

Besprechung der Übungsaufgabe

„Extended Scope“

Fragen?

[3] https://commons.wikimedia.org/wiki/User:Pensiero~commonswiki