

Verlässliche Echtzeitsysteme

Übungen zur Vorlesung

Dynamische Stackbedarfsanalyse

Phillip Raffeck, Florian Schmaus, Simon Schuster

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)

<https://www4.cs.fau.de>

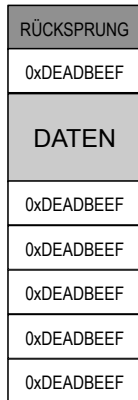
Wintersemester 2020





- **Harte, verlässliche Echtzeitsysteme**
 - Garantien über Ressourcenbedarf notwendig
 - ☞ statische Analyse unabdingbar
- Mögliche Ressourcen: Speicherbedarf, Laufzeit, etc.
- Übung: Analyse des Stackbedarfs einer Bibliothek
- Stack-Analyse
 1. Dynamisch: Wasserstandstechnik
 2. Statisch: „Eigenbau“ und aiT (Stack-Analyzer der a³ Suite)

- **Messung zur Laufzeit:** Wasserstandsmessung
- Grundidee: Einfügen von **Stack Canaries**
- Explizite Verwaltung des Stapelspeichers notwendig
- pthread-Bibliothek ermöglicht Verwaltung
- Mögliche Canaries
 - Lesbare Bitmuster: 0xDEADBEEF
 - Unwahrscheinliche Bitmuster: 0b101010101010...
 - Kleinere Bitmuster \rightsquigarrow größere Auflösung
- ⚠ Keine allgemeingültigen Aussagen
 - Liefert nur den konkreten Bedarf der Messungen
 - Vorsichtige Aussagen über Worst-Case-Verhalten
- Einsatz zur dynamischen Fehlererkennung





1. (Globalen) Stack anlegen:

```
1 static unsigned int g_data[DATA_SIZE];
```

2. Thread anlegen & starten:

```
1 pthread_t thread;  
2 pthread_attr_t attr;  
3 pthread_attr_init(&attr);  
4 pthread_attr_setstack(&attr, &g_stack, STACK_SIZE);  
5 // worker function: void *run(void *param)  
6 int status = pthread_create(&thread, &attr, run, NULL);  
7 if (status != 0) { ... // handle error
```

3. Auf Thread warten:

```
1 pthread_join(thread, &ret);
```



pthread Stack

