Verlässliche Echtzeitsysteme

Übungen zur Vorlesung

Codereplikation

Phillip Raffeck, Tim Rheinfels, Simon Schuster, Peter Wägemann

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

Wintersemester 2022

Code-Replikation: Stringification

Stringification von CPP

```
1 #define CMP_FUNC(pre, repl, type, op) \
2     type pre##repl(type a, type b) {  \
3         return a op b ? a : b;        \
4     }
5
6 CMP_FUNC(max, 1, int, >); // Funktion ?
7 CMP_FUNC(max, 2, int, >); // Funktion ?
8 ...
9 CMP_FUNC(min, 1, int, <); // Funktion ?
```

- Verwendung des C-Präprozessors (CPP)
- ##: „Token Pasting Operator“
- Konkatenieren zweier Token zu einem
- Aufruf & Deklaration müssen erstellt werden

☞ Es geht eleganter ...

Code-Replikation: Stringification

Stringification von CPP

```
1  #define CMP_FUNC(pre, repl, type, op) \
2      type pre##repl(type a, type b) {  \
3          return a op b ? a : b;        \
4      }
5
6  CMP_FUNC(max, 1, int, >); // Funktion max1
7  CMP_FUNC(max, 2, int, >); // Funktion max2
8  ...
9  CMP_FUNC(min, 1, int, <); // Funktion min1
```

- Verwendung des C-Präprozessors (CPP)
- ##: „Token Pasting Operator“
- Konkatenieren zweier Token zu einem
- Aufruf & Deklaration müssen erstellt werden

☞ Es geht eleganter ...

C++ Templates

C++ Template

```
1  template <typename T>
2  T max(T x, T y) {
3    T value;
4    if (x < y) value = y;
5    else value = x;
6    return value;
7  }
8  ...
9  double md = max<double>(2.3, 4.2);
10 auto   mi = max<int>(23U, 42);
```

- Templates ermöglichen generische Programmierung
- Wiederverwendung durch **Parametrisierung**
- Unterscheidung von Funktions- & Klassen-Templates
- Expansion zur Compilezeit $\rightsquigarrow$ Quelltext muss verfügbar sein (im Header)
- „Code Bloat” beim Compilieren $\rightarrow$ **nutzbar für Replikation von Code**

C++ Templates

- Explizite Typen als Templateparameter möglich
- Nutzbar zum „Zählen” von Templates

Indizierte Template-Spezialisierung

```
1  template <typename T, unsigned INDEX>
2  T max(T x, T y) {
3    T value;
4    if (x < y)
5      value = y;
6    else
7      value = x;
8    return value;
9  }
10 ...
11 auto m0 = max<int, 0>(23, 42);
12 auto m1 = max<int, 1>(23, 42);
```

Einbinden von C++-Code in C

C Linkage

```
1  // C++ code
2  extern "C" void f(int); // one way
3  extern "C" {            // another way
4    int g(double);
5    double h();
6  };
7  void code(int i, double d)
8  {
9    f(i);
10   int ii = g(d);
11   double dd = h();
12   // ...
13 }
```

- C++ betreibt name mangling

  $\Rightarrow$ `int test(int)` $\mapsto$ `_Z4testi`

- Name mangling verhindern

  $\Rightarrow$ C++-Code aus C-Code aufrufbar