SAARLAND UNIVERSITY
COMPUTER SCIENCE

C/C++ Programmierung

Grundlagen: Der Präprozessor

Sebastian Hack
Christoph Mallon

(hack|mallon)@cs.uni-sb.de

Fachbereich Informatik
Universität des Saarlandes

Wintersemester 2009/2010

Der Präprozessor

§6.10

- Einfaches Textersetzungssystem
- Läuft vor eigentlicher Übersetzung
- Präprozessordirektiven beginnen mit #
  → Muss erster nicht-Leerraum in der Zeile sein
- Enden mit dem Zeilenende
  → Können mit \ als letztes Zeichen der Zeile auf nächste Zeile ausgedehnt werden
- Einsatzzwecke: Konfiguration (z.B. Debugausgaben) und Anpassungen an Plattformen

Nulldirektive

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
#
```

- Tut nichts

#include

§6.10.2

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
#include <name>
#include "name"
```

- Ersetzt Direktive durch Inhalt der benannten Datei
- Wo die Dateien gesucht werden in beiden Varianten implementierungsabhängig
  → <> üblicherweise Systemheader
  → "" üblicherweise Header des Projekts
- Ist rekursiv anwendbar

#error

§6.10.5

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
#error Beliebige Nachricht
```

- Gibt die Meldung auf der Fehlerausgabe aus
- Beendet die Übersetzung mit Fehler
- Häufige Erweiterung: `#warn`

#if

§6.10.1

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
#if expression
#else
#endif
```

- Rechnen nur mit Zahlen mit den üblichen Operatoren
- Kein `sizeof`, Kommaoperator, Zuweisungen, Funktionsaufrufe
- Neuer Operator: `defined X`
  $\longrightarrow$ Ist genau dann `1`, wenn das Makro `X` definiert ist, sonst `0`
  $\longrightarrow$ Optional mit Klammern: `defined(X)`
- `#ifdef X` Abkürzung für `#if defined X`
- `#ifndef X` Abkürzung für `#if !defined X`
- `#else` optional
- `#elif` Abkürzung für `#else` gefolgt von `#if`

Beispiel

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
#if defined __FreeBSD__
#  include <stdlib.h>
#elif defined __linux__ || defined _WIN32
#  include <malloc.h>
#else
#  error Sorry, your platform is not supported
#endif

#if defined DUTCH  + defined ENGLISH + \
    defined FRENCH + defined GERMAN != 1
#  error You must choose exactly one language
#endif
```

#define: Objektartige Makros

§6.10.3

```
#define NAME Ersetzung
```

- Ersetzt alle Vorkommen von `NAME` (ein Bezeichner) durch `Ersetzung` (beliebig viele Token)
  → insbesondere auch leere Eresetzung
- Ersetzung kann wiederum Makros enthalten, die ausgewertet werden
- Werden bei der Verwendung ausgewertet
- Ersetzung nicht selbstrekursiv
- Konvention: Makronamen komplett in GROßBUCHSTABEN mit Unterstrichen
- Früher: Zur symbolischen Benennung von Konstanten
  → Wird bei globaler Initialisierung immernoch benötigt (nicht C++)

#define: Funktionsartige Makros

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
#define NAME(arg0, ...) Ersetzung
```

- Kein Leerzeichen zwischen `NAME` und `(`
  $\longrightarrow$ sonst objektartiges Makro
- Ersetzt `NAME(...)` durch `Ersetzung`, alle Argumente werden eingesetzt
- Früher: Offener Einbau von Programmstücken

Funktionsartige Makro und Seiteneffekte

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
#define MIN(a, b) ((a) < (b) ? (a) : (b))

// Verwendung:
MIN(++i, 0)
// Ersetzung:
((++i) < (0) ? (++i) : (0))
```

- Falle: `++i` wird (evtl.) zweimal ausgewertet!

#undef


```
#undef NAME
```

- Löscht das Makro `NAME`

Vordefinierte Makros

- `__DATE__`: Datum der Übersetzung
- `__TIME__`: Zeitstempel der Übersetzung
- `__STDC__`: Wert `1`, um eine konforme Implementierung anzuzeigen
- `__FILE__`: Name der aktuellen Datei
- `__LINE__`: Aktuelle Zeilennunmmer
- Datei und Zeile mit `#line` manipulierbar:

```
#line 666 "nirgendwo"
```

→ Kommt bei Programmgeneratoren zum Einsatz

#-Operator

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
#define STRINGIFY(x) #x

// Verwendung:
M(hello world)
// Ersetzung:
"hello␣world"
```

- Wandelt Makroargument in Zeichenkettte um

Beispiel: assert()

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
#undef assert
#ifdef NDEBUG
#  define assert(x) ((void)0)
#else
#  define assert(x) ((x) ? ((void)0) : \
  __assert(#x, __FILE__, __LINE__))
#endif
```

- `/usr/include/assert.h`
- `__assert()` ist externe Funktion, die die Bedingung, Datei und Zeile ausgibt sowie den Programmlauf abbricht

Falle: Klammerung

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
#define SIX     1 + 5
#define NINE    8 + 1
#define ANSWER SIX * NINE
```

Falle: Klammerung

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
#define SIX     1 + 5
#define NINE    8 + 1
#define ANSWER  SIX * NINE
```

- Antwort: 42