

## 1 Präprozessor

### 1.1 Include und Macro-Expansion

Demonstrieren Sie anhand eines einfachen Programmes wie der Präprozessor (cpp) arbeitet. Verwenden Sie in ihrem Programm zumindest zwei verschiedene Standard-Includes und ein eigens definiertes Funktionsartiges Makro.

### 1.2 Standards

Verschiedene Implementationen der C Library bieten z.T. zusätzliche Komfortfunktionen. Vergleichen Sie die Auswirkung auf verschiedene Header der C Library in dem Sie die Ausgaben des Präprozessors vergleichen, zum Beispiel:

```
#define _GNU_SOURCE
#include <fcntl.h>
```

```
#define _POSIX_C_SOURCE
#include <fcntl.h>
```

Konsultieren Sie dazu auch die Manpage zu `feature_test_macros` für alternativen:

```
1 man 7 feature_test_macros
```

## 2 Compile, Assemble, Link

In der Vorlesung haben Sie mehrere Schritte kennengelernt, die der GCC Compiler hinter den Kulissen ausführt. Führen Sie die einzelnen Schritte manuell durch und erzeugen Sie ein funktionsfähiges "Hello World!". Machen Sie sich zunächst mit den Schritten vertraut die der GCC (oder ein anderer Compiler) auf ihrem System ausführt. Benutzen Sie dazu den Verbose-Mode ihres Compilers. Für GCC z.B.:

```
1 gcc -v -o hello hello.c
```

## 3 Automatische Optimierung

Moderne Compiler verfügen über verschiedene Strategien, um ihren Code automatisch zu optimieren.

### 3.1 Vergleich der Ausgabe

Vergleichen Sie den Effekt von verschiedenen Optimierungsleveln und Compiler-Flags auf den generierten Maschinencode. Benutzen Sie zunächst die Option `-O0` um ohne Optimierungen zu kompilieren. Mögliche Optimierungslevel sind z.B. `-O1`, `-O2`, `-O3` oder `-O3 -funroll-loops`.

```

1 #include <stdio.h>
2
3 double compute(double d, unsigned n)
4 {
5     double x = 1.0;
6     unsigned j;
7
8     for (j = 1; j <= n; j++) {
9         x *= d;
10    }
11
12    return x;
13 }
14
15 int main (void)
16 {
17     double result = 0.0;
18     unsigned i;
19
20     for (i = 1; i < 999999999; i++) {
21         result += compute(i, i % 7);
22     }
23
24     printf ("result = %g\n", result);
25
26     return 0;
27 }

```

### 3.2 Vergleich der Performance

Benutzen Sie das `time` Kommando um die Performance ihrer Varianten zu testen. Die Ausgabe könnte z.B. wie folgt aussehen:

```

1 $ time ./00
2 result = 2.04082e+61
3
4 real    0m13.142s
5 user    0m13.102s
6 sys     0m0.004s
7
8 $ time ./01
9 result = 2.04082e+61
10
11 real    0m3.088s
12 user    0m3.081s
13 sys     0m0.002s

```

Dokumentieren Sie auch auf was für einem Prozessor Sie gemessen haben (zum Beispiel mithilfe des `lscpu` Kommandos).