Universität Hamburg, Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen
E. Betke, N. Hübbe, M. Kuhn, J. Squar, J. Lüttgau

Übungsblatt 10 zum Praktikum
C-Programmierung
im WiSe 2019/2020
Abgabe: 22.12.2019 (23:59)

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>
```

Machen Sie den Vergleich für mindestens zwei weitere Standard-Header. Konsultieren Sie dazu auch die Manpage zu `feature_test_macros` für mögliche 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

Bewertete Aufgabe!
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 (Assembler-)Maschinencode. Benutzen Sie zunächst die Option `-O0` um ohne Optimierungen zu kompilieren. Mögliche Optimierungslevel sind z.B. `-O1`, `-O2`, `-O3` oder `-O3 -funroll-loops`.

```c
#include <stdio.h>

double compute(double d, unsigned n)
{
    double x = 1.0;
    unsigned j;

    for (j = 1; j <= n; j++) {
        x *= d;
    }

    return x;
}

int main (void)
{
    double result = 0.0;
    unsigned i;

    for (i = 1; i < 999999999; i++) {
        result += compute(i, i % 7);
    }

    printf ("result = %g\n", result);

    return 0;
}
```

3.2 Vergleich der Performance

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

```
$ time ./O0
result = 2.04082e+61

real    0m13.142s
user    0m13.102s
sys     0m0.004s

$ time ./O1
result = 2.04082e+61

```

```
real    0m3.088s
user    0m3.081s
sys     0m0.002s
```

Überlegen Sie sich für den Vergleich eine angemessene (visuelle) Darstellung. Dokumentieren Sie auch auf was für einem Prozessor Sie gemessen haben (zum Beispiel mithilfe des `lscpu` Kommandos).

Abgabe

Abzugeben ist ein gemäß den bekannten Richtlinien erstelltes und benanntes Archiv. Das enthaltene und gewohnt benannte Verzeichnis soll folgenden Inhalt haben:

- Alle Quellen, aus denen Ihre Evaluation besteht.

Senden Sie Ihre Abgabe an `cp-abgabe@wr.informatik.uni-hamburg.de`.