C-Präprozessor - Übungsaufgaben

Lukas Stabe

Aufgabe 1

Nehmen Sie an Sie arbeiten gerade an einem Projekt. Dieses Projekt verwendet die Library `awesome`. Die Library deklariert in ihrem Header neben einigen Funktionen auch das Makro `__awesome_ver`, das die Version der Library enthält.

Den aktuelle Stand des Quellcodes Ihres Projektes sehen Sie hier:

main.c

```c
#include "include/awesome.h"

int main(int argc, char **argv) {
    printf("starting\n");
    something_awesome(42);
    printf("done\n");
    return 0;
}
```

a) In der gerade neu erschienen Version 3 der `awesome`-Library wurde der Name einiger Funktionen geändert. `something_awesome(int a)` heisst nun `do_something_awesome(int a)`.

   Es sollen künftig sowohl die neue, als auch alte Versionen der Library unterstützt werden. Passen Sie den Code so an, dass er mit beiden Versionen der Library kompiliert.

b) In ihrem Projekt soll in Zukunft ein Feature der Library verwendet werden, das erst ab Version 2 zur Verfügung steht.

   Ergänzen Sie den Code so, dass das Übersetzen mit einer aussagekräftigen Fehlermeldung abgebrochen wird, sollte die Library in einer Version kleiner als 2 vorliegen.

c) Immer wieder beschweren sich Benutzer, dass das Programm zu viel Debug-Text auf die Kommandozeile ausgebe.

   Sorgen Sie dafür, dass dies nur noch geschieht, wenn beim Kompilieren

das Makro `DEBUG` definiert war. Umstellen Sie dabei *nicht* die einzelnen `printf(...)`-Anweisungen mit `#if ... #endif`.

d) Zusatzaufgabe: Finden Sie heraus, wie sie dem GCC-Compiler Makros als Kommandozeilenoptionen übergeben können.

Aufgabe 2

In den folgenden Makro-Definitionen sind Fehler versteckt, die bei der Benutzung der Makros zu unerwarteten Ergebnissen führen. Geben Sie je ein Beispiel für die Auswirkungen der Fehler und korrigieren, wenn möglich, die die Definition.

a)
```
#define 2_PI 3.14 + 3.14
```

b)
```
#define MAX(a, b) ( a > b ? a : b )
```

c)
```
#define MANY_THINGS(a) something(a); something_else(a)
```

Aufgabe 3

Geben Sie an, wie folgender Quellcode nach Verarbeitung durch den Präprozessor aussehen würde.

```
%:include <stdio.h>
#define MAIN_RETURN int
#define MAIN_ARGS int argc, char** argv
MAIN_RETURN ma\
in(MAIN_ARGS) ??<
printf("He\
llo world!??/n");
%>
```

Verwendete Di- und Trigraphen:
`'%:' = '#', '??<' = '{', '??/' = '\', '%>' = '}'`