

Beispiel zur Auflösung von Makros durch den C - Präprozessor

(vgl. Script 11 – 4)

Seien folgende Makros definiert:

```
#define PB(format, vari) printf("%" #format "\n", vari)

#define NL putchar('\n')

#define PC(form, v) { PB(form, v); NL; }
```

Das Makro **PB** mit den Parametern **format** und **vari** erzeugt einen Aufruf der Funktion **printf**, bei dem die „Variable“ **vari** unter Verwendung des Kennbuchstabens **format** sowie anschließend ein „newline“-Zeichen (**'\n'**) ausgegeben werden.

Achtung: **format** würde hier nicht in einem String (z.B. **"%format\n"**) durch den Parameter ersetzt!

NL ist eine symbolische Konstante, sie wird im Quelltext einfach durch **putchar('\n')** ersetzt, d.h. es wird das „newline“-Zeichen (**'\n'**) ausgegeben.

Der vom Makro **PC** mit den Parametern **form** und **v** erzeugte Quelltext enthält einen „Aufruf“ des Makros **PB** sowie die symbolische Konstante **NL**, wobei die erhaltenen Parameter an **PB** weitergereicht werden (s.u.).

PC erweitert **PB** also um die Ausgabe eines newlines und damit einer Leerzeile unter dem ausgegebenen Wert.

Wir können demnach

```
PC(c, 65);
```

wie folgt auflösen:

Zunächst betrachten wir die Definition **PC** s

```
{ PB(form, v); NL; };
```

und ersetzen die Parameter durch ihre tatsächlichen Werte:

```
{ PB(c, 65); NL; };
```

Diese Schritte wiederholen wir für **PB** und **NL**:

```
{ printf("%" #format "\n", vari); putchar('\n'); };
```

```
{ printf("%c\n", 65); putchar('\n'); };
```

Der Präprozessor ersetzt also den Ausdruck `PC(c, 65);` im Quelltext vor der Kompilierung durch `{ printf("%c\n", 65); putchar('\n'); };` .

Anmerkung:

Die Umwandlung von

```
{ printf("%" #format "\n", vari); putchar('\n'); };
```

erfolgt über den Zwischenschritt

```
{ printf("%" "65" "\n", 65); putchar('\n'); };
```

und führt dank Stringkonkatenation zu unserem Ergebnis

```
{ printf("%c\n", 65); putchar('\n'); }
```

Wie wir sehen schließt `#` in einem Makro vor einem Parameter diesen beim Ersetzen in `"` ein.

Ich möchte an dieser Stelle an die Möglichkeit erinnern, einen String (z.B. bei der Ausgabe) auf diese Weise zu trennen. So sind beispielsweise

```
printf("Das ist ein sehr langer String");
```

und

```
printf("Das ist ein sehr"  
      " langer String");
```

äquivalent.