

Tipps zur Quellcodeformatierung

Praktikum mobile Roboter

Marcus Obst

Professur für Prozessautomatisierung
TU Chemnitz

16. Januar 2007

*Most people's [...] programs should be indented
six feet downward and covered with dirt.*

BLAIR P. HOUGHTON

- 1 Formatierung
 - Einrückung
 - Leerzeichen
 - Leerzeilen
 - Klammern
- 2 Variablen
- 3 Funktionen
- 4 Inlinedokumentation
- 5 Häufige Fehler
- 6 Fazit

Einrückung (Indentation)

- schnelles Erkennen von logischen Blöcken
- Tiefe: 2, 4 oder 8?

schlecht

```
do{
  //Motor starten
  motor_set(LINKS,vlinks);
  motor_set(RECHTS,vrechts);
  links_alt=links;
  rechts_alt=rechts;
  if (MIN(links,rechts) > 1000)
  warte(100);
  else
  warte(20);
} ...
```

besser

```
do {
  // Motor starten
  motor_set(LINKS,vlinks);
  motor_set(RECHTS,vrechts);
  links_alt=links;
  rechts_alt=rechts;
  if (MIN(links,rechts) > 1000) {
    warte(100);
  } else {
    warte(20);
  }
} ...
```

- Leerzeichen statt Tabulatoren
- Zeilenlänge max. 78 Zeichen (Terminal, Email)

Leerzeichen

- Hinter Kontrollschlüsselwörtern (if, while, for,...) sollte immer ein Leerzeichen stehen
- ein Funktionsname sollte nicht von der folgenden Klammer getrennt werden
- **Operatoren** (+, -, =, Komma, ...) hingegen sollten mit Leerzeichen abgetrennt werden (Erleichterung für die Augen)

richtig schlecht

```
do{  
  ...  
}while((!taster_get (TASTER_ROT))&&(abs (rechts)<anz));  
  
motor_set (LINKS|RECHTS,0);  
  
verh=-((float)vlinks)/((float)vrechts);
```

Leerzeichen (Fortsetzung)

schon besser

```
do {  
  ...  
} while ( (! taster_get(TASTER_ROT) ) && ( abs(rechts) < anz ) );  
  
motor_set(LINKS | RECHTS, 0);  
  
verh = -( (float) vlinks ) / ( (float) vrechts );
```

Segmentierung (Chunking)

- Logische Abschnitte durch Leerzeilen abgrenzen. (analog zum Absatz im Aufsatz)
- gute Stelle für Kommentare!

Klammern

Bevorzugt K&R-Stil verwenden.

Example

```
if (x is true) {  
    we do y  
}
```

Example

```
/*  
 * Funktionen sind  
 * eine Ausnahme  
 */  
  
int function(int x)  
{  
    body of function  
}
```

- öffnende Klammer ans Ende der Zeile
- schließende Klammer steht alleine

Funktionen sind anders:

- hier steht die öffnende Klammer am Beginn der neuen Zeile

Klammern (Fortsetzung)

Alternative sind auch andere Möglichkeiten okay.

Example

BSD-Stil

```
if (condition)
{
    printf("BSD-Stil\n");
}
else
{
    printf("Mir ist schlecht...\n");
}
```

Example

GNU-Stil

```
if (condition)
{
    printf("GNU-Stil\n");
}
else
{
    printf("Mir auch...\n");
}
```

Allerdings verschwendet man hier unnötig Zeilen.
Nutzt diese lieber für Kommentare!

Variablen

- globale Variablen sollten eine aussagekräftigen Namen bekommen (Unterstriche nutzen, z.B `ich_bin_eine_globale_variable`)
- globale Variable soweit wie möglich vermeiden
- lokale Variablen sollten kurz bekannt werden (Eindeutigkeit!)
- Defines immer groß schreiben!

Example

```
#define PI 3.148

int ...

void calculate_parameters(void)
{
    int i;
    for (i = 0; i < 100; ++i) {
        ...
    }
}
```

Funktionen

- Funktionen sollten kurz und klar sein
- immer nur eine Sache tun
- Länge einer Funktion ist umgekehrt proportional zur Komplexität

Inlinedokumentation

Vorteil

Wenn man sowieso gerade im Quellcode ist, ist die Chance auch groß, dass man gleich noch den Kommentar mit ändert.

Ein guter Tipp: doxygen (PDF, HTML, Text)

Example

Doxygen-Beispiel

```
/** Abfrage einer Motorgeschwindigkeit
    \param welcher \c LINKS oder \c RECHTS
    \return Die Geschwindigkeit von \a -100
            bis \a 100 in Prozent
 */
signed byte motor_get(unsigned byte welcher);
...
```

Die komplette Bibliothek ist mit Doxygen dokumentiert!

Häufige Fehler

- Vorsicht bei Vergleichen = anstatt von ==

- Ganzzahldivision!

Wie lautet das Ergebnis von `float x = 5 / 2`?

Lösung:

```
float x = 5.0 / 2
```

```
float x = (float) 5/2
```

- „kurzes if“ vermeiden → immer Klammern setzen

```
if (ostern) {  
    eier_farbe = "bunt";  
} else {  
    eier_farbe = "braun";  
}
```

8 Gebote

- 1 Du sollst Einrücken!
- 2 Nutze Leerzeichen und Leerzeilen
- 3 Dokumentiere bzw. Kommentiere
- 4 Begrenze die Anzahl der Variablen
- 5 Mehr als vier bis fünf Einrückungsebenen könnten bedeuten, dass dein Code schlecht ist!
- 6 Funktionen sollten max. 1 bis 2 Bildschirmseiten lang sein
- 7 Kommentiere den **Zweck**, nicht die Aktion!
- 8 Sei konsistent!

Referenzen



Linux CodingStyle

<http://lxr.linux.no/source/Documentation/CodingStyle?v=2.6.18>



Perl Best Practices,
DAMIAN CONWAY,
O'Reilly 2006



Inlinedokumentation mit Doxygen,
<http://www.doxygen.org>