ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing Group

Research in Computer Science

Code Formatting for TinyOS in Eclipse

Jianyuan Li

Advisor

Benjamin Sigg

Supervisor

Prof. Dr. Roger Wattenhofer

Distributed Computing Group

ETH Zurich

December 2009

1 Introduction

TinyOS [1] is a wide spread operating system designed for wireless sensor networks. *nesC* [2] is a modified version of the C programming language used to build applications for the TinyOS platform. *Yeti 2* [3] provides a nesC editor implemented as a plugin for Eclipse [4].

Developers always follow a particular programming style when writing source code. A programming style is a set of code formatting rules that help programmers to read and understand code, and help to avoid errors. To make sure the formatting rules are always applied, code formatting tools are used.

This work adds code formatting capabilities for nesC to Eclipse. It offers programmers a fast way to modify the source code so that the formatting rules are fulfilled.

2 Design and Implementations

Generally, code formatting is about proper spaces between tokens, proper line-feeds between statements, and proper indentation for each line. The code formatting for nesC works in multiple steps, which are shown in Figure 2.1.

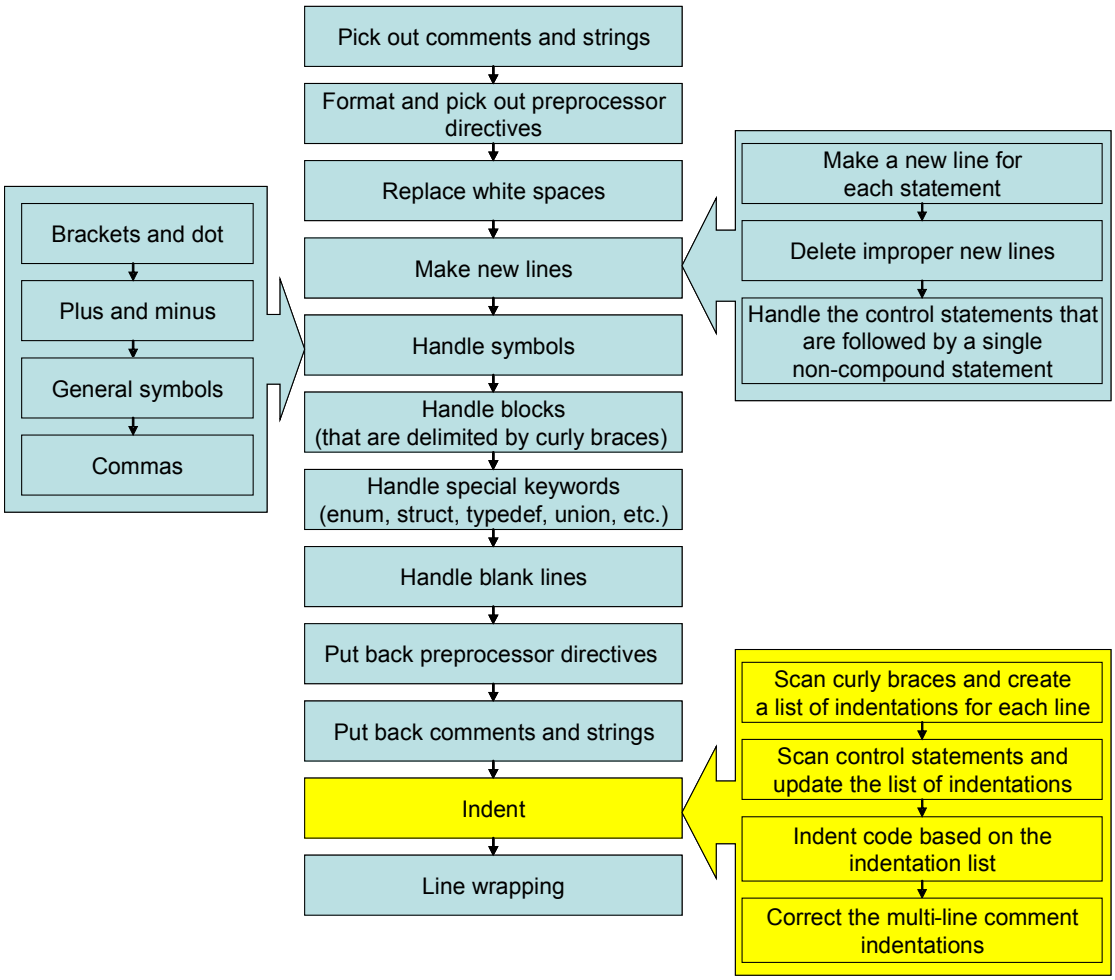

Figure 2.1 Code formatting flow for nesC

2.1 Code Formatting

To avoid the confusion caused by comments and strings, for example, comments may include keyword text, all comments and strings are temporarily removed. Since preprocessor directives are separated from nesC code, they are temporarily removed as well. Each whitespace region, which could include spaces, tabs and line-feeds, is replaced with a single space. After this, the whole source code is in a single line.

The strategy to make new lines works in three steps. First, add a new line after each single statement that ends with a semicolon. Second, correct the improper line-feeds from the first step. For example:

```
for(int i = 0; i < 10; i++) {
    …
}
```

In this *for* loop case, there should not be any line-feed after a semicolon. Third, handle the control statements that are followed by a single non-compound statement, which

does not use braces:

```
if(i < 0)
    i++;
```

Here, the statement after *if* should be in a new line.

Next, symbols are handled group by group according to their different formatting features. Parentheses, square brackets and dots do not need spaces before and after. Plus and minus are processed case by case, since they may be in one of the following forms: *a* - *b*, *a*++ + ++*b*, *a* + -*b*, etc. The other symbols, including '<', '>', '|', '^', '!', '?', ':', '%', '=', '&', '/', and '*', will appear as one of these symbol sequences: one single symbol ('<'), double symbols ('<<'), a symbol followed by an equal ('<='), or double symbols followed by an equal ("<<="). In most cases, there should be a space before and after such a symbol sequence, but no space in it (e.g., a && b). Special cases, like "+=", "->", "&p", "**p", are handled specially. A comma needs no space before, but one space after it (e.g., a, b).

After this, line-feeds are added before or / and after curly braces that delimit blocks. Then improper line-feeds made by the previous step in the cases of initialization for arrays, enum, struct, typedef, union and so on, are deleted. Next, add proper blank lines, put back preprocessor directives, comments and strings according to the marks made at the beginning of the algorithm.

By now, the code formatting is almost finished except indentation. The code indentation here can be used separately as a simple form of code formatting. Details will be shown in section 2.2.

The last step is line wrapping. If a line has more than 80 characters, it will be wrapped at a proper offset, which is normally before or after a specified symbol.

2.2 Indentation

In this work, a variant of K&R style [5] is applied to nesC code. The K&R style is one of the most popular indent styles for C. It keeps the first opening brace on the same line as the function or the control statement, indents the statements within the braces, and puts the closing brace on the same indentation level as the function or the control statement. For example:

```
int main() {
    if(exp)
        do_something();
    else {
        do_something_else();
```

```
        }
    }
```

To fulfill the indent rule, the code is scanned twice. First, the code is scanned for braces and a list of indentations is created for each line. During the scanning, indentations are increased for lines after an opening brace, and decreased for lines after a closing brace. Second, the code is scanned for control statements that are followed by a single statement without using braces. If the statement is not on the same line as the control statement, add indentations for the statement line. After the scanning, a list of indentations for every line has been created, and the code gets indented based on the list. The indentations of multi-line comments could be different from normal statements because of the star at the beginning of each line, so the last step for indentation is to correct them.

3 Future Work

This work provides code formatting by using a popular programming style. It does not support configurations for this tool. Following work could be providing a user interface where the user can specify which formatting rules to use.

In addition, code could be formatted in different ways based on the context of tokens. For example, the token "->" in "*BlinkC.Leds -> LedsC;*" and "*dataPtr->ID = ID;*" should have a different format. In the first case it means binding an interface to another, while in the second it means referring to a particular member (*ID*) of pointer *dataPtr*. Also, in the case of "message_t* msgPtr = &m;" and "a * b = c", whether the star (*) works for a pointer or works as a multiplication sign should be distinguished.

References

[1] TinyOS, http://www.tinyos.net/

[2] NesC, http://nescc.sourceforge.net/

[3] Yeti 2, http://tos-ide.ethz.ch

[4] Eclipse, http://www.eclipse.org/

[5] K&R style, The C Programming Language by Brian Kernighan and Dennis Ritchie, 1978