Contents

1 Introduction 2

2 Comments 2
   2.1 Clearifying 2
   2.2 Documenting Bugs 2
      2.2.1 Gotcha Keywords 2
      2.2.2 Gotcha Formatting 3

3 Statements - Bracing, Parenthesis, and Spacing 3

4 Variables 4

5 The Preprocesor 4
   5.1 #defines 4
   5.2 Macros 4
1 Introduction

This is meant to serve as a way to make our code look prettier, more readable, and more correct. The better we can understand what we were thinking when we wrote the code, the better chance we have of creating a successful project.

2 Comments

Comments need to be used to clearify code and document bugs that have not been fixed.

2.1 Clearifying

2.2 Documenting Bugs

When documenting bugs, there needs to be a standardized way to search through the code in order to identify what needs to be fixed. Several projects have a system whereby “Gotcha Keywords” are used. The “Gotcha” system is described below.

2.2.1 Gotcha Keywords

- :TODO: topic
  Means there’s more to do here, don’t forget.

- :BUG: [bugid] topic
  means there’s a Known bug here, explain it and optionally give a bug ID.

- :KLUDGE:
  When you’ve done something ugly say so and explain how you would do it differently next time if you had more time.

- :TRICKY:
  Tells somebody that the following code is very tricky so don’t go changing it without thinking.

- :WARNING:
  Beware of something.

- :COMPILER:
  Sometimes you need to work around a compiler problem. Document it. The problem may go away eventually.

- :ATTRIBUTE: value
  The general form of an attribute embedded in a comment. You can make up your own attributes and they’ll be extracted.
2.2.2 Gotcha Formatting

Make the gotcha keyword the first symbol in the comment. Comments may consist of multiple lines, but the first line should be a self-containing, meaningful summary. The writer’s name and the date of the remark should be part of the comment. This information is in the source repository, but it can take quite a while to find out when and by whom it was added. Often gotchas stick around longer than they should. Embedding date information allows other programmer to make this decision. Embedding who information lets us know who to ask. Example

    // :TODO: tmh 960810: possible performance problem
    // We should really use a hash table here but for now we’ll
    // use a linear search.

    // :KLUDGE: tmh 960810: possible unsafe type cast
    // We need a cast here to recover the derived type. It should
    // probably use a virtual method or template.

3 Statements - Bracing, Parenthesis, and Spacing

It is hard to deal with non-standardized statements such as the following examples:

    if (collision==BETA_NOT_COLLIDE) {
            . . .
    }

or

    while( collision == BETA_NOT_COLLIDE ){
            . . .
    }

or

    for(collision == BETA_NOT_COLLIDE)
    {
            . . .
    }

Each one of these statements was written a little differently and each style has its pros and cons. However, the first of these styles is decidedly best for our project.
4 Variables

5 The Preprocessor

5.1 #defines

5.2 Macros