Comments and Documentation

2501ICT/7421ICTNathan

René Hexel

School of Information and Communication Technology
Griffith University

Semester 1, 2012

Outline

Comments

- Plain C allows comments between `/*` and `*/`
  - `/* this is a valid C comment */`
- Comments may not be nested
  - `/* this /* is not a valid C comment */ */`
- C99 also allows double-slash `//` end-of-line comments
  - `// this is a valid comment`
  - no closing sequence needed – the comment ends at the end of the line

Comment Example

Example (Program with Comments)

```c
/*
* This program prints "j = 007".
* It does not take any parameters and returns 0 on success.
*/
int main(void)                    /* main function definition */
{
  int j;                          // our int variable to play with

  j = 7;                          // assign a value to be printed
  printf("j = %03.3d\n", j);      // print value with leading zeroes

  return 0;                       // everything is fine, exit program
}
```

Where to put comments?

- At the beginning of each file (module)
    - describe the name of the module, purpose, author, and dates when first created and last modified
- Before each function (method)
    - describe the purpose of the function or method,
    - input parameters (arguments),
    - return values (output parameters), and
    - pre- and postconditions (contract)
- At the beginning of each class
    - describe the purpose of the class, and
    - things to keep in mind when using this class

How to comment?

- Use comments to document important parts of your code
- Document key functionality
- Don’t re-iterate the obvious!

Example (Bad comment)

```
i = 7;  // assign 7 to i
```

Example (Better)

```
i = 7;  // seven iterations to go
```

Extracting Documentation from your Program

- Everybody hates writing documentation, right?
  - can be lots of work
  - duplicated efforts if all the information is already in the source code
- The good news: Tools that extract documentation from the source
  - JavaDoc (Java specific)
  - HeaderDoc (`http://developer.apple.com/opensource/tools/headerdoc.html`)
    - in the labs: can use JavaDoc syntax for C, C++, Objective-C
  - Doxygen (`http://www.stack.nl/~dimitri/doxygen/`)
    - similar, installed on `dwarf`
  - AutoGSDoc
    - part of the GNUstep environment on Linux and Windows

Automatic Documentation Example

Example

```c
/**
* The main() function of this program prints "Hello World" and
* then exits.  This function does not take any parameters and
* returns 0 to indicate success.
*/
int main(void)
{
  printf("Hello World!\n");
  return 0;
}
```

Using Doxygen

- Doxygen allows to automatically extract comments from source code
  - similar to JavaDoc
  - requires a configuration file
- How to come up with a configuration file?
  - create by hand
    - complex
    - ⇒ can be error-prone
  - automatically create a template, then modify to suit your project
    - log in to `dwarf`
    - go to your project (assignment) working directory
    - run `doxygen -g` to create a configuration file called Doxyfile
    - edit `Doxyfile` to suit your needs (set `PROJECT_NAME`, `OUTPUT_DIRECTORY`, etc.
- Run `doxygen Doxyfile` to generate documentation for your project
  - add a `Documentation` target to your `Makefile`