COMP 2400 UNIX Tools

Christian Grothoff
christian@grothoff.org

http://grothoff.org/christian/

Coding Standards

- Avoid bugs!
- Make your code easier to read/learn/understand
- Enable you to show/ship your code
- Reduce maintenance costs (curly wars!)
- Help broaden your customer base
- Avoid bugs!

Document Contributors

- Helps to identify who can be asked about a particular issue.
- Crucial for copyright reasons: who owns the code, legally speaking?
- Applies to code, comments and documentation files. For copyright law, comments and code are just text.
- Version control systems can help, but should not be relied upon as the only way to track contributions!
- State the license for your code clearly

Coding Style

- Avoid conditional compilation (#ifdef)
- Avoid system-specific extensions (MSC, gcc), stick to the language standard!
- Avoid esoteric languages, unless there is a huge benefit

Write Robust Programs

- Avoid arbitrary limits on the length or number of any data structure
- Check every system call for an error return, and include the system error message (`strerror`) in your program error message
- Call `abort()` if and only if error checks detect impossible conditions
- Write reentrant code wherever possible

Equality testing

If you want to test if variable x has a particular constant value, use

```
if (5 == x) { ... }
```

instead of

```
if (x == 5) { ... }
```

Development Priorities

- First write a text-mode (shell) interface for your program!
- Once that is working, you can consider a graphical interface.
- This will facilitate testing, profiling and broaden your userbase.

Memory Usage

- A few megabytes are hardly ever an issue
- But avoid memory use equivalent to the size of your inputs or outputs if your input/output sizes are unbounded!

Formatting

- Use a consistent way of formatting your code!
- Most editors provide formatting help.
- Do not have lines longer than 76 characters.
- Make meaningful use of whitespace to ease readability.
- Avoid too much whitespace to fit reasonable amounts of code onto the screen.

Commenting

- Good code needs few comments – good variable names, function names, types and code structure document most of it
- All names and comments should always be in English
- Brief comment at the start of each source file describes its overall purpose
- Each non-trivial function should have comment describing its purpose, including the meaning of the arguments and return value

Writing text

- No space before comma, dot, semicolon
- Two spaces after dot at the end of sentences
- Stick to the 76 character per-line limitation
- Good spelling and grammar are important, even for comments!

No warnings!

- Make sure that your code does not cause any warnings (`-Wall`) from the compiler
- If the compiler is really, really wrong, learn the syntax to disable the warning manually (`@nowarn`)

Try SSA!

- Only declare one varible per line (no `int a,b,c`)
- If possible, declare and define the variable in one line
- If possible, only have a single assignment per variable
- Avoid declaring temporaries that are only defined and used once, except if you need to break up long lines or use the variable name to document what is going on

Naming

- Look for names that give useful information
- The name should be longer if the symbol is visible to a larger scope
- The name can be shorter if the symbol is used very frequently
- Use `i,j,k` for local integer loop variables
- Use `d,f` for local floating point variables
- Use `n,m` for (array) sizes

Internationalization

- Learn about how to use GNU gettext for your language
- Mark all messages given to the user early on, even if you are not going to ship with translations initially.
- Stick to 7-bit ASCII text for your source code, even in Java

JavaDoc

- http://java.sun.com/j2se/javadoc/writingdoccomments/
- `javadoc -d /home/html -sourcepath /home/src -su java`
- `-exclude java.net:java.lang`
- `-windowtitle $WINDOWTITLE`

Doxygen

- Documentation tool like JavaDoc, just better
- Supports C, C++, Java, Python and others
- Generates HTML or LaTeX
- For Java, use JavaDoc-compatible syntax

Questions

?