Compilation and debugging

Gianluca Campanella

Compilation

Compilation steps

1. Preprocessing
2. Compilation
3. Linking

Preprocessing

- Transforms lines starting with `#`
- Mostly used to `#include` files
- Another common usage are `#include` guards
- Can also be used to `#define` macros

Compilation

- Transforms code to machine code
- Itself composed of multiple steps:
  1. Parsing and type checking
  2. Non-specific optimisation (e.g. dead code elimination)
  3. CPU-specific optimisation (e.g. instruction scheduling)

Linking

- Combines multiple files into a final executable
- Organises a program's address space (e.g. through relocation)
- Two main strategies:

  Dynamic (Some) libraries are loaded at runtime

  Static All libraries are included

Debugging

Let’s use a debugger!

Other options

- Assertions
- Logging
- Unit tests