Compilation and debugging

Gianluca Campanella

Compilation

Compilation steps

1. Preprocessing
2. Compilation
3. Linking

Preprocessing

The preprocessor…

- Transforms lines starting with `#`
- Is mostly used to `#include` files
- Can also be used to `#define` macros

Compilation

The compiler...

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

Linking

The linker...

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

Dynamic	(Some) libraries are loaded at runtime
Static	All libraries are included in the executable

Debugging

Let’s use a debugger!

Other options

- Assertions
- Logging
- Unit tests