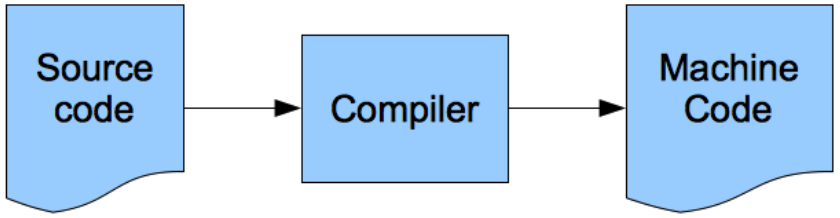
Lesson 4: Compilers, Names, & Interfaces

- Optimizing compilation
- Name lookup
- Interface design



4.1 Minimize Compile Dependencies

- Goal: Speed up compilation
- Analyze code
  - Minimize compile-time dependencies
- Guidelines
- Optimized code

4.2 Minimize Compile Dependencies 2

- Goal: Speed up compilation even more
- Analyze code
  - Hide private class implementation
- The Pimpl Idiom
  - Yes you read that right!
- Guidelines

4.3 Minimize Compile Dependencies 3

- What other changes can we make?
  - Replace inheritance with composition
- Final optimized code

4.4 Compilation Firewalls

- All users of class must recompile
  - …when any member in definition changes
- Pimpl idiom solves this
- What should go in Ximpl?

4.5 Name Lookup

- Analyze code
  - Which functions are called and why?
- Koenig lookup
- Conclusion
  - Namespaces not as independent as originally thought
  - Koenig lookup makes things work like we’d expect

4.6 Interface Principle

- What is “part of” a class?
- Can free function be part of a class?
- Interface principle
  - How does it interact with Koenig lookup?

4.7 How to Best Implement operator<<

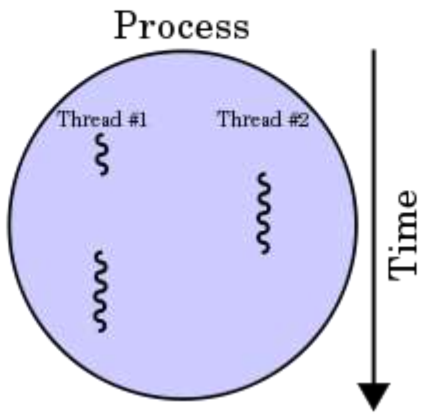
- Two options:
  - Free function using usual class interface
  - Free function calling virtual print()
- Traditional analysis
  - Flawed
- Correct analysis

4.8 Name Hiding

- Analyze code sample
  - Which names are visible where?
- How to work around unwanted name hiding
- Name hiding in nested namespaces
- Guidelines

4.9 Portable Multithreading

- Terminology
- Why multithreading
- Multithreading in C++


Process
Thread #1
Thread #2
Time