What's New in CDT

Sergey Prigogin
Google

CDT committer, refactoring component lead
Areas of active development

- **C++11 support**
  
  Recently added support for:

  ```cpp
  using foo = std::array<int, 10>;
  
  constexpr int foo(int a, int b) { return a * b; }
  ```

  Current state in
  [http://bugs.eclipse.org/bugs/showdependencytree.cgi?id=395568](http://bugs.eclipse.org/bugs/showdependencytree.cgi?id=395568)

- **Preservation of typedefs**

- **Refactoring**
Brief history of refactoring in CDT
What’s next in refactoring?

- **Organize Includes**
  - Required for new refactorings (*Inline*, *Change Method Signature*)
  - Useful by itself. Bugzilla enhancement request was created in 2003 and has 38 comments
What needs to be organized?

- Includes
- Forward declarations
- Using declarations

```cpp
#include <string>
using std::string;
string concatenate(const string& pieces...);

void main(int argc, const char* argv[]) {
    string s = concatenate(argv[1], argv[2]);
    ... 
}
```
Include vs forward-declare

```c
A foo(A a) { // definition of A is required
    return a;
}

A* bar(A* a) { // definition of A is not required
    return a;
}

void baz(A* a) {
    a->f(); // definition of A is required
}

class C {
    D x; // definition of D is required
    static E y; // definition of E is not required
};
```
Who is responsible for inclusion?

MyString.h

```cpp
class MyString {
public:
    MyString(const char* s);
};
```

Compare.h

```cpp
class MyString; // is forward declaration enough?
int compare(const MyString& s1, const MyString& s2);
```

main.cpp

```cpp
#include "Compare.h"
int main(int argc, const char* argv[]) {
    return compare(argv[1], argv[2]);
}
```
Indirect inclusion

Font.h

```c
enum Font { TIMES_ROMAN, HELVETICA };
```

Graphics.h

```c
#include "Font.h"
void drawLine(int x1, int y1, int x2, int y2);
void setFont(Font font);
```

main.cpp

```c
#include "Graphics.h"
void main() {
    drawLine(0, 0, 1, 1);
    Font f = TIMES_ROMAN;
    ...
}
```
More about indirect inclusion

- **Include What You Use** principle
- Representative header files
  - `<vector>`, not `<bits/stl_vector.h>`
  - NULL is defined in 13 headers
Private and public headers

component/*.h can be included from anywhere

component/internal/*.h can be included only from component/.* and component/internal/.*
Flavors of include statements

- **Angle brackets or quotes**
  
  ```
  #include <vector>
  
  #include "my_vector"
  ```

- **Short or long path**
  
  ```
  #include "point.h"
  
  #include "graphics/primitives/point.h"
  ```
Grouping and ordering of includes

- With the same name but different extension
- In the same folder
- In subfolders
- “System” includes
- User-defined groups
Grouping example

/MyProject/src/time/DateTime.cpp

#include "time/DateTime.h"

#include <sys/time.h>
#include <time.h>
#include <cstdio>
#include <string>

#include "time/Duration.h"
#include "time/timezone/TimeZone.h"

#include "base/Types.h"
#include "strings/Format.h"
#include "util/Logging.h"

...
Preferences, preferences, preferences…

- Inclusion vs forward declaration
- Header file substitution
- Style of include statements
- Grouping and ordering
- What to do with unused includes
Demo