# Instructions for Developers #

## Submitting Patches ##

We welcome patches and rely on your contributions to make IWYU smarter.

Coding and testing guidelines are available in the [IWYU Coding Style](docs/IWYUStyle.md) guide.

Use GitHub's [pull request system](https://github.com/include-what-you-use/include-what-you-use/pulls) to submit change requests to the `include-what-you-use/include-what-you-use` repo.

It's usually a good idea to run ideas by the [IWYU mailing list](http://groups.google.com/group/include-what-you-use) to get general agreement on directions before you start hacking.

## Running the Tests ##

If fixing a bug in IWYU, please add a test to the test suite! You can create a file called `whatever.cc` (\_not\_ .cpp), and, if necessary, `whatever.h`, and `whatever-<extension>.h`. You may be able to get away without adding any `.h` files, and just including `direct.h` -- see, for instance, `tests/remove\_fwd\_decl\_when\_including.cc`.

To run the IWYU tests, run

python run\_iwyu\_tests.py

It runs one test for each `.cc` file in the `tests/` directory. (We have additional tests in `more\_tests/`, but have not yet gotten the testing framework set up for those tests.) The test runner searches for IWYU in the system `PATH` by default.

The output can be a bit hard to read, but if a test fails, the reason why will be listed after the `ERROR:root:Test failed for xxx` line.

You can select individual tests by listing their filename without extension as arguments

python run\_iwyu\_tests.py array macro\_location

If you don't want to modify your `PATH` you can specify which IWYU executable to use for testing

python run\_iwyu\_tests.py -- ./include-what-you-use

(put any test names before '--' and the IWYU path after.)

When fixing `fix\_includes.py`, add a test case to `fix\_includes\_test.py` and run

python fix\_includes\_test.py

## Debugging ##

It's possible to run include-what-you-use in `gdb`, to debug that way. Another useful tool -- especially in combination with `gdb` -- is to get the verbose include-what-you-use output. See `iwyu\_output.h` for a description of the verbose levels. Level 7 is very verbose -- it dumps basically the entire AST as it's being traversed, along with IWYU decisions made as it goes -- but very useful for that:

env IWYU\_VERBOSE=7 make -k CXX=/path/to/llvm/Debug+Asserts/bin/include-what-you-use 2>&1 > /tmp/iwyu.verbose

## A Quick Tour of the Codebase ##

The codebase is strewn with TODOs of known problems, and also language constructs that aren't adequately tested yet. So there's plenty to do! Here's a brief guide through the codebase:

\* `iwyu.cc`: the main file, it includes the logic for deciding when a symbol has been 'used', and whether it's a full use (definition required) or forward-declare use (only a declaration required). It also includes the logic for following uses through template instantiations.

\* `iwyu\_driver.cc`: responsible for creating and configuring a Clang compiler from command-line arguments.

\* `iwyu\_output.cc`: the file that translates from 'uses' into IWYU violations. This has the logic for deciding if a use is covered by an existing `#include` (or is a built-in). It also, as the name suggests, prints the IWYU output.

\* `iwyu\_preprocessor.cc`: handles the preprocessor directives, the `#includes` and `#ifdefs`, to construct the existing include-tree. This is obviously essential for include-what-you-use analysis. This file also handles the IWYU pragma-comments.

\* `iwyu\_include\_picker.cc`: this finds canonical `#includes`, handling private->public mappings (like `bits/stl\_vector.h` -> `vector`) and symbols with multiple possible #includes (like `NULL`). Additional mappings are maintained in a set of .imp files separately, for easier per-platform/-toolchain customization.

\* `iwyu\_cache.cc`: holds the cache of instantiated templates (may hold other cached info later). This is data that is expensive to compute and may be used more than once.

\* `iwyu\_globals.cc`: holds various global variables. We used to think globals were bad, until we saw how much having this file simplified the code...

\* `iwyu\_\*\_util(s).h` and `.cc`: utility functions of various types. The most interesting, perhaps, is `iwyu\_ast\_util.h`, which has routines that make it easier to navigate and analyze the clang AST. There are also some STL helpers, string helpers, filesystem helpers, etc.

\* `iwyu\_verrs.cc`: debug logging for IWYU.

\* `port.h`: shim header for various non-portable constructs.

\* `iwyu\_getopt.cc`: portability shim for GNU `getopt(\_long)`. Custom `getopt(\_long)` implementation for Windows.

\* `fix\_includes.py`: the helper script that edits a file based on the IWYU recommendations.