Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Thu, 13 Sep 2007 14:13:10 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 13 September 2007 14:31You have too simple view of the issue. In most cases, you cannot just parse the current line or something like that; that would lead to much worse situation than we have now.

Consider e.g:

foo.h:

```
struct A { Class1 x; void MyFun(); };

struct B { Class2 y; void MyFun(); }
```

foo.cpp:

```
#include "foo.h"

void A::MyFun()
{
   x.
```

Changeing void A::MyFun to B::MyFun completely changes the context.

Detecting such changes is more expensive than parsing the file each time '.' is pressed.

Mirek

No, I did explain myself bad.... I mean that you have to parse the file up to the cusor. This with or without preprocessor stuff. Well, you could detect some trivial changes that don't need it, but mostly you have to reparse.

Looking at the problem a bit more, I've some thoughts :

1- Reparsing on EACH change is too time expensive. Even if you reparse only on '.' or '->' code.
2- Preprocessor would add much more overhead, too.

so, the solution : a timed full preprocess/parse in background... let's say, each 2-3 minutes, for example, and then switch the completion context to the new parsed one once done.
Or, if you don't care slowing down the machine a bit, a continuous parse thread with 2 contexts kept in memory, the last one and the currently parsing one. Like a sort of double buffer.
You can miss some completion in the parse time, but I think this will be quite a rare possibility, depending only on time between parses. It should even be faster than now, if you're reparsing

real-time on each '.' keypress.
What do you think about ? That should also be easy to implement, I guess.

Ciao

Max

Edit

On second thought, here more than a full preprocessor, you should only process the #includes and gather #define stuff for code completion, so you can present macro, but with untyped parameters. CPP does the full job with macros, replacing them with the defined code, so it's of no use for help in macro completion. But the rest is still valid. And a preprocessor that gathers #includes only is a trivial task, too.
