Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Thu, 13 Sep 2007 18:49:37 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 13 September 2007 19:37
Once again, preprocessing in background, using "cpp", does not work. There is no result that can be used for anything useful (because the file has changed meanwhile).

Here I don't agree. The cpp is not useful because it makes macro substitution, so you can't use macro names in code completion, that's right.
But saying that a user is so fast typing code that he notices that the backgriund preprocess/parse has still old data, IMHO is wrong.

Quote:
Aha! You are starting to have a clue about the problem

So you will soon see that really the simplest solution is to do it right:) (Which +/- means caching the results of preprocessing at the end of each header).

No, I don't think so... why make things complicated ? Why try to make it real time stuff, very difficult task, when you can make it a background stuff sparing time and resources ?
More, if you cache the file at the end of an header you can fall n the problem we spoke about some posts ago :

test.h
#define amacro(x) x*x

test.cpp

#include "test.h"
int main()
{
  int z = amacro(.... <here you should get completion right.

but then :

test2.h
#undef amacro
#define amacro(x, y) x * y

test.cpp

#include "test.h"
#include "test2.h"  <--- ADDED LINE
int main()
{

int z = amacro(.... <what do you get here ?

In my example, if you want right completion, you should repreprocess/reparse file while typing, or at least detect the inclusion of test2.h and reparse the file. Long task. If you set the reparse to manual ("rescan file" button) you have the wrong completion if you don't hit the button.
And whorse, if you use the "hit the button" method, user must wait 1-4 seconds that the process is finished.
As you said before, you can't cache the results on a 'per header' basis, as results for an header may depend on previous ones.
Last but not least, you must cache a lot of data anyway. So I don't see the point of keeping it all in foreground.

again : whouldn't be better to have in memory a double buffered parse three, and swap on a timed basis with the one from the background job ?
Using background tasks have another advantage : you must have in memory ONLY the current file (and includes); when you switch to another source code, simply discard all data and start another background process. The user will maybe notice that he has no code completion for 3-4 seconds, at most.

Ciao

Max

hemm... I've another question, about layouts, but I'll start a new topic