Subject: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Tue, 11 Sep 2007 22:50:12 GMT
View Forum Message <> Reply to Message

I'm testing U++ and TheIDE since less than one week, so I will put some (newbie) first impressions and what I'd like to see in next releases

-The concept of packages, nests and assemblies is very good. It forces some order in code, besides of other advantages. I'd prefere a three view on left pane, indeed. I like also how output files are organized.

-The editor is nice, and quite fast, but compared to scintilla found in some other IDEs (codeblocks, for example) it miss many confortable commands, like rectangular blocks, comment blocks of code, and so on. Why not scintilla inside TheIDE ?

-Too many toolbars are missing; I know thet in some IDEs they're too much, but I really think that a SAVE button is a must!. Ok, CTRL-S does the job, but when you have a hand on mouse, it's more confortable to hit the toolbar.

-I like the layout editor besides 2 points : the first is the way to add a widget... I'd prefere a bin 'a-la-borland', for example. The second is the location of 'anchors' (springs, ecc...); I think they're part of widget, so they should be put in widget property panel. But I really like the layout editor, I tested some wxwidgets edtors, and they were all quite unconfortable.

-Help system should be customizable, so users can add his/her own help files and topics. I think this would help also to have more tutorials from users.

Now, some 'small' things that are missing :

- Draggable and dockable toolbars and panels. If you write a customizable app with many many commands (a cad, for example) they are a must. Those or command line... menus are too slow, and fixed toolbars are difficult to customize by end user. In such apps user normally select a (small) subset of command and put in toolbars, and the less used commands are done with menu or keyboard.

- MDI interface and DOC/VIEW support. I know MDI is old, but many people likes it.

- The parser. I've read in another thread that his problem are macros. So, why not to feed the source in CPP preprocessor before parsing ? I know it's time consuming, but result could be cached on diskfile and updated only on changes... maybe in background.

a last word.... I find U++ code style one of the best I've ever seen. That's a good usage of C++, not a bunch of #defines !

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]

Posted by unodgs on Wed, 12 Sep 2007 07:07:25 GMT
View Forum Message <> Reply to Message

Quote:-The editor is nice, and quite fast, but compared to scintilla found in some other IDEs (codeblocks, for example) it miss many confortable commands, like rectangular blocks, comment blocks of code, and so on. Why not scintilla inside TheIDE ?
Because of a degree of control I know scintilla is a very advanced editor control, but it's easier to extend something that was built by us, something that perfectly fit in upp architecture. IMO editor needs:
1. rectangular blocks
2. ability to define syntax highlighing rules by user
3. default highlighting for xml, html (see point 2)
Commenting of blocks of code is available now. Just select block of code and press / or shift-8 or ctrl-/. Key / works if editor enclose selection is set.
Quote:-Too many toolbars are missing; I know thet in some IDEs they're too much, but I really think that a SAVE button is a must!. Ok, CTRL-S does the job, but when you have a hand on mouse, it's more confortable to hit the toolbar.
Frankly you don't need save button as theide saves the file each time you switch to another one or when you compile or close theide.
Quote:-I like the layout editor besides 2 points : the first is the way to add a widget... I'd prefere a bin 'a-la-borland', for example. The second is the location of 'anchors' (springs, ecc...);
You mean each widget should have icon on toolbar? I vote for it, but only for the most common ones. Anchor system is ok as it is now IMO.
Quote:- Draggable and dockable toolbars and panels. If you write a customizable app with many many commands (a cad, for example) they are a must.
Absolutly, it was discussed before and it will show up soon or later.
Quote:and fixed toolbars are difficult to customize by end user.rd.

It's on todo

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mirek on Wed, 12 Sep 2007 09:03:22 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Tue, 11 September 2007 18:50
-The concept of packages, nests and assemblies is very good. It forces some order in code, besides of other advantages. I'd prefere a three view on left pane, indeed.

That would work for small projects. The current solutions allows you to reach any file in 700 files project (TheIDE) using two mouse clicks.

Quote:
like rectangular blocks, comment blocks of code, and so on. Why not scintilla inside TheIDE ?

Comment blocks are there, you just need to activate it in preferences.

Rectangular blocks are in todo list.

Quote:
-Help system should be customizable, so users can add his/her own help files and topics. I think this would help also to have more tutorials from users.

Actually, help show documentation of packages, which is part of packages. Just click those .tpp things in e.g. CtrlLib.

Quote:
- The parser. I've read in another thread that his problem are macros. So, why not to feed the source in CPP preprocessor before parsing ? I know it's time consuming, but result could be cached on diskfile and updated only on changes... maybe in background.

Well, but pressing '.' is a change, is not it? The thing is, is this is about to be useful, you really need to reparse the file up to the point you want the information.

Mirek

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Wed, 12 Sep 2007 10:26:52 GMT

View Forum Message <> Reply to Message

unodgs wrote on Wed, 12 September 2007 09:07
Because of a degree of control I know scintilla is a very advanced editor control, but it's easier to extend something that was built by us....
....Commenting of blocks of code is available now. Just select block of code and press / or shift-8 or ctrl-/. Key / works if editor enclose selection is set.

Thanx for the comment hint !
About scintilla, it is quite customizable.... I don't know if it fits well in theide, but I've seen scintilla inside so many ide/editors/tools that it should not be a problem. If I remember well they've separated the low level interface from the rest, so to put inside theide should not be too difficult. I really like theide editor, but imho makes few sense to duplicate all scintilla functions, it's a lot of work !

Quote:
Frankly you don't need save button as theide saves the file each time you switch to another one or when you compile or close theide.

hmmmmm.... besides of users comfort of having a 'save' button (I really feel bad if I don't save my work every 20 seconds, if you work with autocad on windows you understand what I mean !), the autosave feature is good up to an ide crash... working on svn version is not as rare as it may

seem!
Quote:
You mean each widget should have icon on toolbar? I vote for it, but only for the most common ones.

I was thinking at a tabbed control, like borland ones, for example. You could have the most common controls in first tab and access to other ones with no effort, without clobbering the interface (I guess is this one your problem)
Quote:
Anchor system is ok as it is now IMO.

that is of course a matter of taste. I used working on borland tools and I felt confortable to have all widget properties on the same place, but this can be different for you.

Quote:
Absolutly, it was discussed before and it will show up soon or later.

perfect, I can't wait for it !   btw, here I have a suggestion that comes from autocad... it would be nice to have the ability o customization on the fly of toolbars, for example, right clicking a toolbar should allow user to add/remove buttons that can send, maybe, a string or a code to the application. That would allow customization whithout the hassle of dealing with callbacks on the fly.

Quote:
Actually, help show documentation of packages, which is part of packages. Just click those .tpp things in e.g. CtrlLib.

I missed this feature, I'll try it. But what I meant is the ability to insert external html help files on help menu. Maybe with indexing and word search. You could for example add boost or opengl help on the ide. I know that you can open the help out of the ide, but an integration is far more confortable. Codeblocks did it with their help plugin and it's really a nice thing.

Quote:
Well, but pressing '.' is a change, is not it? Smile The thing is, is this is about to be useful, you really need to reparse the file up to the point you want the information.

Yes, the '.' does already a great job   And I like the fact that the parameter name and type are copied on the editor, unlike many other ides that give you a popup hint that mostly disappears when you start typing ! What I meant, and I don't knoow if it's possible, is to feed the code to cpp and then to your parser, to solve macro problems. this could be done in background... the only problem that can arise is if cpp can deal with partial files. I'll make an example of what I mean :
1-you start typing your code, and your parser does his job as usual.
2-when you type a #define or a #include, all the stuff is fed to cpp and then to the parser, in background.
3-from then, if you type things again, your new code is append to cpp output (without a need of another cpp slow run...) and then fed to your parser.
4-when you do some other preprocessor stuffs, the process is repeated.
All that could be done in a background thread and you could switch to preprocessed stuff when is

ready.

That should make a good compromise between speed and effectiveness.

Again, thanx for your great job !

Ciao

Max

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mirek on Wed, 12 Sep 2007 12:40:04 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Tue, 11 September 2007 18:50
-I like the layout editor besides 2 points : the first is the way to add a widget... I'd prefere a bin 'a-la-borland', for example. The second is the location of 'anchors' (springs, ecc...); I think they're part of widget, so they should be put in widget property panel.

You can click on those anchor lines too... I guess this is exactly what you want

---

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mirek on Wed, 12 Sep 2007 12:46:15 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Wed, 12 September 2007 06:26
I missed this feature, I'll try it. But what I meant is the ability to insert external html help files on help menu. Maybe with indexing and word search. You could for example add boost or opengl help on the ide. I know that you can open the help out of the ide, but an integration is far more confortable. Codeblocks did it with their help plugin and it's really a nice thing.

You can, as long as you will make boost a package...

You cannot use HTML directly (yet?), but it should be trivial to copy&paste the help as RTF to Topic++.

Quote:
What I meant, and I don't knoow if it's possible, is to feed the code to cpp and then to your parser, to solve macro problems. this could be done in background... the only problem that can arise is if

cpp can deal with partial files. I'll make an example of what I mean :
1-you start typing your code, and your parser does his job as usual.
2-when you type a #define or a #include, all the stuff is fed to cpp and then to the parser, in background.
3-from then, if you type things again, your new code is append to cpp output (without a need of another cpp slow run...) and then fed to your parser.
4-when you do some other preprocessor stuffs, the process is repeated.

The problem is that macros are in your file too.... You really would need to run .cpp each time you invoke the parser.

Solution is coming, but it really really is far from trivial...
(And it will include partial preprocessor; unfortunately normal preprocessor is not partial...).

Quote:
That should make a good compromise between speed and effectiveness.

We do not need compromise. We need to dedicate one month to solve it right

Mirek

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Wed, 12 Sep 2007 14:18:41 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 12 September 2007 14:46
You can, as long as you will make boost a package...

You cannot use HTML directly (yet?), but it should be trivial to copy&paste the help as RTF to Topic++.

Of course, no problem to make an rtf from html, but then I'd to keep in sync with the html, and that is an hassle.
I guess that in U++ is missing an HTML viewer, isn't it ? I had no time yet to look in gui classes.
Maybe grabbing some html widget eslewhere...

Quote:
The problem is that macros are in your file too.... You really would need to run .cpp each time you invoke the parser.

Not really, if you store a preprocessed version of the file
you should reprocess it ONLY if you change some macros inside.
For example, you start with

```
int main(int argc, char **argv)
{
```

here your default parser does the job. Then you add a macro definition on top :

```
#define blahbla(x) dosomething(x)

int main(int argc, char **argv)
{
```

Here you feed all this on cpp (thread in background) and when it's finished you feed to your parser. All in background, when a task isn't finished, you stay with the previous data.
When you write normal code, macro stuff are a minimal part of the job, so the preprocessor passes should be minimal!

Quote:
Solution is coming, but it really really is far from trivial...
(And it will include partial preprocessor; unfortunately normal preprocessor is not partial...).

uhmmmm ... I'm curious, so I just tried to feed a small sample, unfinished and with open #if/#else/#endif stuff, the preprocessor complained about the missing #endif but did the job quite well.
All macros and includes where preprocessed and sent to output.
Maybe windoze m$ compiler behaves different ?

Quote:
We do not need compromise. We need to dedicate one month to solve it right

Well, if you want to write a full preprocessor plus complete c++ parser, I think you'll need a month just for planning it, and 1 year to code
I was following some threads on codeblocks forum some time ago, they're putting much effort on it too, but it's far from finished.
But well, for my needs, your code completion is more than enough like it is !!!

Just a last question.... I've seen that svn development version is far from being as stable as latest release. It's a matter of missing commits or are you making big changes on the app ? Or maybe it's only the linux/unix side that must be polished ? In that case, I'll be glad to submit bug reports, if you need them.

Best Regards

Max

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]

Posted by mirek on Wed, 12 Sep 2007 15:48:04 GMT
View Forum Message <> Reply to Message

Well, actually, we already have C++ parser

All we need is that partial preprocessor.

The problem of your solution is that you really need to invoke the preprocessor each time you want the info. macros present or not, you need the complete sources up to the point of cursor.

There is another problem too: After preprocessing, you would have to parse it too...

Mirek

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Wed, 12 Sep 2007 16:12:19 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 12 September 2007 17:48Well, actually, we already have C++ parser

All we need is that partial preprocessor.

The problem of your solution is that you really need to invoke the preprocessor each time you want the info. macros present or not, you need the complete sources up to the point of cursor.

No, you must invoke the preprocessor only when you make some changes in source code that need it, i.e. change a #define or an #ifdef or so. If you don't fiddle with preprocessor stuff, the previous preproc run is enough !
And if you think that in a normal source code file the preprocessor stuff is about 1-5% max of the file, you must agree that preprocessor is not called too often ! Another task would be if you edit some wxwidget file...ehehehehe

Quote:
There is another problem too: After preprocessing, you would have to parse it too...

Of course, but again, the complete reparse should be needed ONLY when you change some preprocessor stuff in code.

I'm not telling you that's an easy task, but I think it's much easier than to write a complete c++ preprocessor.

Ciao

Max

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mirek on Wed, 12 Sep 2007 16:40:42 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Wed, 12 September 2007 12:12luzr wrote on Wed, 12 September 2007 17:48Well, actually, we already have C++ parser

All we need is that partial preprocessor.

The problem of your solution is that you really need to invoke the preprocessor each time you want the info. macros present or not, you need the complete sources up to the point of cursor.

No, you must invoke the preprocessor only when you make some changes in source code that need it, i.e. change a #define or an #ifdef or so. If you don't fiddle with preprocessor stuff, the previous preproc run is enough !

What file do you want to parse? Buffered file from preprocessor? Actual file you are working on?

Quote:
I'm not telling you that's an easy task, but I think it's much easier than to write a complete c++ preprocessor.

The preprocessor is the simple part... The hard part is caching logic.

Mirek

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Wed, 12 Sep 2007 18:59:08 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 12 September 2007 18:40
What file do you want to parse? Buffered file from preprocessor? Actual file you are working on?

buffered file from preprocessor kept in sync with the file I'm workin on.... This is the hardest (and fastest) way. But there's a simpler way :
I've not looked at your parser code, but I guess it does a first scan on file when ide opens, and then it keeps scanning only the changes you make to file, am I wrong ?
So, at ide opening you add the preprocessor pass before the full scan pass. Then, when you work on file, there are 2 possibilities :
1- you make changes not related to # preprocessing directives, so your code work as usual.
2- you make changes that involves preprocessing directives. So you start a cpp preprocess in background, then a full scan in background too. While this (long) process is not finished, you keep working as usual on previous buffered file; when preprocess-scan thread finishes, you switch to the new buffer.

Quote:
The preprocessor is the simple part... The hard part is caching logic.

Of course, but that is already done. It should only a matter of keeping a second buffer (aka symbol table in memory) and switch to it when the long job is done. It should be not too difficult to insert in current code. I'd say that the hard part is the scanner, but this one seems to me very well done yet.

Ciao

Max

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mirek on Wed, 12 Sep 2007 19:52:16 GMT

View Forum Message <> Reply to Message

[quote title=mdelfede wrote on Wed, 12 September 2007 14:59]luzr wrote on Wed, 12 September 2007 18:40

file from preprocessor kept in sync with the file I'm workin on....

How do you want to keep it in sync?

Note that rescanning after #define is typed is not enough.

Any change in file changes potential preprocessor output. This is not even related to macros, even if there are no macros, any change in file changes preprocessor output.

Quote:
I've not looked at your parser code, but I guess it does a first scan on file when ide opens, and then it keeps scanning only the changes you make to file, am I wrong ?

Yes, you are wrong

All project files are scanned at ide startup(*). Then modified files are scanned before they are leaved or if the information is needed.

(*) in fact, all information is cached, so in reality only changed files are scanned at startup, unchanged file information is just loaded.

Quote:
So, at ide opening you add the preprocessor pass before the full scan pass. Then, when you work

on file, there are 2 possibilities :
1- you make changes not related to # preprocessing directives, so your code work as usual.

Any change in file is related to preprocessor.

That is the problem you seem to miss (or maybe I do not understand your method).

Quote:
Quote:
The preprocessor is the simple part... The hard part is caching logic.

Of course, but that is already done. It should only a matter of keeping a second buffer (aka symbol table in memory) and switch to it when the long job is done. It should be not too difficult to insert in current code.

If only things would be so simple

Note:

What you suggest might partially work, if preprocessor would be capable of exporting all defined macros together with preprocessor output. In that case, you could split the processing to processing of headers, which would yield two files (preprossed files and set of macros) and the main file, then you could e.g. include only macros for processing of main file, which would be much faster. But hey, this is what I call "partial preprocessing".

Not that there is another problem: You are not only interested in information contained in the .cpp file, but also in things defined in headers...

Mirek

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Wed, 12 Sep 2007 22:15:52 GMT
View Forum Message <> Reply to Message

---

luzr wrote on Wed, 12 September 2007 21:52
How do you want to keep it in sync?

Note that rescanning after #define is typed is not enough.

Any change in file changes potential preprocessor output. This is not even related to macros, even if there are no macros, any change in file changes preprocessor output.

I don't understand your point, I think. You're saying that you scan the file on startup (or get from last saved cache), then only on demand. Let's start with an example :

```
#include <stdio.h>

int funca(int z)
{
  return 2*z;
}

int main(int argc, char *argv[])
{
  int x = 5;
  int y;

  y = funca(x);

}
```

the file here is saved.
When you reload, you add:

```
#include <stdio.h>

int funca(int z)
{
  return 2*z;
}

int funcb(int k)
{
  return k*k;
}

int main(int argc, char *argv[])
{
  int x = 5;
  int y;

  y = funca(x);
  x = funcb(  <-- here you have parameter suggestion

}
```

In this case, I don't know if you're parsing all the file again or only the added code... But let imagine that your program is very smart and chooses the hard and faster way. So you find the code difference from the previous parse, and add funcb() definition. Doing so makes things hard when you delete a function or rename it, but can be made, I think...
Let's change things, and add a #define and a #include ;

```
#include <stdio.h>

int funca(int z)
{
  return 2*z;
}

#define macroc(x, y) (x) + (y)

#include "mystuff.h"

int funcb(int k)
{
  return k*k;
}

int main(int argc, char *argv[])
{
  int x = 5;
  int y;

  y = funca(x);
  x = funcb(  <-- here you have parameter suggestion

}
```

Here you have 2 ways :
get a partial preprocessor and feed it with the #define and #include, but.... it's not enough, you miss the #include at the beginning. So, you MUST feed preprocessor with this also.
But it's not eno"ugh yet, if for example your "mystuff.h" test for 'funca' definition it fails, because you didn't feed the preprocessor with 'funca' definition.

To make it simple, you MUST feed the preprocessor with ALL code from the beginning of file, to be sure to catch all cases. No choice. So, it makes no sense to have a partial preprocessor; you need only a preprocessor that accept a truncated file as input. CPP does the job. Whathever you do, you must process the file from the beginning when you add # stuffs.

Quote:
All project files are scanned at ide startup(*). Then modified files are scanned before they are leaved or if the information is needed.

(*) in fact, all information is cached, so in reality only changed files are scanned at startup, unchanged file information is just loaded.

so, let's say, before scanning and caching the file, you add the preprocess pass.
On the former example (before macro stuff), you have the funca defined and main code. You add the funcb, then start type in main() the x = funcb(...) and here you must call the scanner, I think, or you'll miss funcb() parameter suggestion.

Simple way : ignore it, up to the next file reload.
Slow way, rescan all file from the beginning on the fly.
Smart way, scan only the diff.

Wathever you're doing, let's add the preprocess stuff now :

Simple way : ignore ecc ecc
Slow way, repreprocess and rescan from the beginning
Smart way : too complicated
Smartest way : Take simple way and do the slow way in a background thread. When the latter is done, swap the parsed threes!
The point is : the user don't notice a delay, he can continue typing AND have the completion with THE OLD stuff; when the new scan is done, the system switch to new stuff. As normally a programmer types (and think) much more slowly than a computer, he will feel like you made a Slow scan but with the speed of simple way.

Quote:
Any change in file is related to preprocessor.

That is the problem you seem to miss (or maybe I do not understand your method).

Maybe we're speaking about the same thing

Quote:
Note:

What you suggest might partially work, if preprocessor would be capable of exporting all defined macros together with preprocessor output. In that case, you could split the processing to processing of headers, which would yield two files (preprossed files and set of macros) and the main file, then you could e.g. include only macros for processing of main file, which would be much faster. But hey, this is what I call "partial preprocessing".

Not that there is another problem: You are not only interested in information contained in the .cpp file, but also in things defined in headers...

No, here you're right. If you want a full code completion, you MUST preprocess all stuffs, and that from begnning. It's of no use feed the preprocessor only with macros; some includes depends on previous macros, and viceversa.
The only way I see, wherever you use cpp or your preprocessor, is to smart preprocess/scan all stuff in background and keep a shadow copy of previous job while the new one is done.

I really don't see it too hard, the most is done in current parser... but maybe I'm wrong somewhere...

Ciao

Max

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mirek on Thu, 13 Sep 2007 12:31:44 GMT
View Forum Message <> Reply to Message

You 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

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.

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mirek on Thu, 13 Sep 2007 17:37:40 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Thu, 13 September 2007 10:13
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.

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).

Quote:
On second thought, here more than a full preprocessor, you should only process the #includes and gather #define stuff for code

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).

Mirek

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

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mirek on Fri, 14 Sep 2007 07:40:16 GMT
View Forum Message <> Reply to Message

You still do not get it, but never mind

The issue is that you HAVE TO PREPROCESS THE FILE YOU ARE WORKING ON AT THE VERY MOMENT YOU NEED THE INFO end of story, otherwise the results from all this are only worse than what we have now - actually, how are you going to detect that you are in main? What if "main" is a macro?

Mirek

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mirek on Fri, 14 Sep 2007 07:43:20 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Thu, 13 September 2007 14:49
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.

Actually, I never said that and in fact, that is exactly what I am going to do.

The fact that results for an header depends on previous one is just an complication.... (this is the hard part I was speaking about).

Mirek

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Fri, 14 Sep 2007 10:44:51 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 14 September 2007 09:40You still do not get it, but never mind

The issue is that you HAVE TO PREPROCESS THE FILE YOU ARE WORKING ON AT THE VERY MOMENT YOU NEED THE INFO end of story, otherwise the results from all this are only worse than what we have now - actually, how are you going to detect that you are in main? What if "main" is a macro?

I agree that you've to process ALL of the file up to cursor. In theory, to get the 'perfect' result, you should do it for each keypress of (or . or -> or :: (hope I'm not missing others....)
The problem is speed, and I see it very hard job.
What I'm saying (last time, I promise !   ) is that doing it in background and not in real time improves MUCH the 'speed' (on the user feeling side) with really very few disadvantages. And simplifies a lot your code, too.

Quote:
Actually, I never said that and in fact, that is exactly what I am going to do.

The fact that results for an header depends on previous one is just an complication.... (this is the hard part I was speaking about).

uhhmmm... so I did misunderstand it. But I still think that it will be too complicated to account for previous headers changes when you cache the current one. You have to detect those changes and rescan current header too.

Well.... you know the matter better than I ! It's a pity that I've got no spare time at the moment, I'd like to test my way of doing the job

Ciao

Max

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mirek on Fri, 14 Sep 2007 13:45:58 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Fri, 14 September 2007 06:44uhhmmm... so I did misunderstand it. But I still think that it will be too complicated to account for previous headers changes when you cache the current one.

You do not need to do that. What you really need is to make the current macro set part of the "caching key" (together with header file path and time).

Mirek

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Fri, 14 Sep 2007 17:46:44 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 14 September 2007 15:45

You do not need to do that. What you really need is to make the current macro set part of the "caching key" (together with header file path and time).

I see the point. But you must anyways rescan all headers coming after the changed or inserted one. Ok that'll be a rare occasion too.

Ciao

Max

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mirek on Tue, 25 Sep 2007 12:22:50 GMT
View Forum Message <> Reply to Message

---

mdelfede wrote on Fri, 14 September 2007 13:46luzr wrote on Fri, 14 September 2007 15:45

You do not need to do that. What you really need is to make the current macro set part of the "caching key" (together with header file path and time).

I see the point. But you must anyways rescan all headers coming after the changed or inserted one. Ok that'll be a rare occasion too.

Ciao

Max

Actually, maybe my working theory is wrong, but I think you do not have to rescan headers in this case, as long as the change in header does not change the macros...

Therefore this comes to quite simple scheme - each header has set of "input macros" as part of caching keys and produces a set of "output macros". As long as caching key matches, you can just take output macros of header instead of parsing it when you see #include.

Or that is the plan If you see any flaw, do not stay silent, you would save my time following a wrong approach...

(Well, the one possible flaw is that of course, some program constructs can start in one header and end in another. I decided simply to ignore this possibility for now...)

Mirek

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Tue, 25 Sep 2007 14:42:29 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 25 September 2007 14:22
Actually, maybe my working theory is wrong, but I think you do not have to rescan headers in this case, as long as the change in header does not change the macros...

I don't think you're wrong, I only think all that is VERY complicated !

Quote:
Therefore this comes to quite simple scheme - each header has set of "input macros" as part of caching keys and produces a set of "output macros". As long as caching key matches, you can just take output macros of header instead of parsing it when you see #include.

Or that is the plan If you see any flaw, do not stay silent, you would save my time following a wrong approach...

No flaws at all, it seems all ok, but really complicated to code... If I did understand, you do that :
1 - On first scan, completely scan all headers
2 - Starting from first one (no input macros), you get 'output macros' from it and get some sort of hash or magic code from those macros.
3 - You repeat all that for following headers, storing the 'magic' on input and tha cached macros.
4 - On next scans, if first header is changed you rescan it and get again the 'magic code'
5 - if 'magic code' didn't change even if the header changed, you don't need to rescan following headers.

The only (small) flaw I see is that you change a macro on first header you must indeed rescan all following headers, but that's rare.

Quote:
(Well, the one possible flaw is that of course, some program constructs can start in one header and end in another. I decided simply to ignore this possibility for now...)

well, if a programmer does such a construct, he deserves the bad behaviour of parser.... I'd call it "just force things to make them buggy"

What I don't see are 'big' advantages against a simpler (much simpler) background processing.
Ok, you're real time each key you press, but your code will become very complicated and, in the (rare) possibility that user change a simple macro on first header, he will notice a long delay on editor.
OTOH, your way is of course more 'technical' and 'funny' to code...

Ciao

Max

p.s. do you think also to make some sort of word completion (besides member function/variables) ? Something like "type-2-chars-press-a-key-and-get-a-list-of-words" ? That would be useful too...

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mirek on Tue, 25 Sep 2007 16:22:42 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Tue, 25 September 2007 10:42luzr wrote on Tue, 25 September 2007 14:22
Actually, maybe my working theory is wrong, but I think you do not have to rescan headers in this case, as long as the change in header does not change the macros...

I don't think you're wrong, I only think all that is VERY complicated !

Why? Well, there are some complicated details, but I think the basic principle is simple...

Quote:
The only (small) flaw I see is that you change a macro on first header you must indeed rescan all following headers, but that's rare.

Yes, but there is no other way...

Quote:
What I don't see are 'big' advantages against a simpler (much simpler) background processing.

Bacgkground processing really is not possible, I have shown you before. You do not know when it is supposed to stop. You cannot get a list of macros from preprocessor so that you can continue.

Quote:
p.s. do you think also to make some sort of word completion (besides member function/variables) ? Something like "type-2-chars-press-a-key-and-get-a-list-of-words" ? That would be useful too...

That is already there, ctrl+,

Mirek

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Tue, 25 Sep 2007 18:02:38 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 25 September 2007 18:22
Bacgkground processing really is not possible, I have shown you before. You do not know when it is supposed to stop. You cannot get a list of macros from preprocessor so that you can continue.

Why ? Doing it in background you don't have to stop. You can (if you want) keep parsing the file and includes in background and switch from 2 contexts on the fly... All you can notice with this way is an outdated context between 2 parse cycles but, if you don't type (and think) at light speed, that would be unnoticeable.

Just an example of what can happen :
1) you start with this file :

```
#include <stdio.h>

int dosomething(int a, int b)
{
  return a*b;
}

main()
{
  dosomething(  <-- CODE COMPLETION HERE)

}
```

After the first (background) scan, that works. Now, let add an include file 'inc.h' :

```
#define dosomething(x) (x)+2
```

So, you add an include in your file :

```
#include <stdio.h>

int dosomething(int a, int b)
{
  return a*b;
}

#include "inc.h"

main()
{
  dosomething(  <-- CODE COMPLETION HERE)

}
```

There are 2 possibilities :

a- when you're typing 'dosomething(' you've got the old context, because the new one is not ready, so you get the wrong completion. But parse times should be matter of 2-3 seconds, so it's a rare event! the time you change add the include line and move the cursor, parser usually does ts work.

b- the new parsed context is ready, the ide switches to it, and you'll get the correct completion.

I really don't see what can be wrong.... you don't have to cache macros, don't have to worry about changes of macros in includes... All you need is to continue parsing the source (and its includes) in background.

The only pitfall is to double the memory to keep 2 contexts and some cpu time lost on background parses but, for an ide those are minor problems.

ah, if you meant that you can't use CPP, you're right, I was wrong, you can't use it for macros.

Quote:
That is already there, ctrl+,

..... I keep learning new stuffs each day...
Thanx for the hint!

Ciao

Max

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mirek on Tue, 25 Sep 2007 21:01:49 GMT

View Forum Message <> Reply to Message

mdelfede wrote on Tue, 25 September 2007 14:02luzr wrote on Tue, 25 September 2007 18:22
Bacgkground processing really is not possible, I have shown you before. You do not know when it is supposed to stop. You cannot get a list of macros from preprocessor so that you can continue.

Why ? Doing it in background you don't have to stop. You can (if you want) keep parsing the file and includes in background and switch from 2 contexts on the fly... All you can notice with this way is an outdated context between 2 parse cycles but, if you don't type (and think) at light speed, that would be unnoticeable.

Just an example of what can happen :
1) you start with this file :

#include <stdio.h>

```
int dosomething(int a, int b)
{
  return a*b;
}

main()
{
  dosomething(  <-- CODE COMPLETION HERE)

}
```

After the first (background) scan, that works.

OK, stop right there.

How does background scan know where is your cursor?

Mirek

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Tue, 25 Sep 2007 21:59:49 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 25 September 2007 23:01

OK, stop right there.

How does background scan know where is your cursor?

He could know, but it isn't necessary.
He could know where the cursor is when you start the background scan... no problem. But for what ? When you store the parse context, you only add line numbers on scanned data, that's all. When you look for completion, you look ONLY on scanned data that ha s line number BEFORE the current one.

Ciao

Max

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]

Posted by mirek on Tue, 25 Sep 2007 22:24:28 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Tue, 25 September 2007 17:59luzr wrote on Tue, 25 September 2007 23:01

OK, stop right there.

How does background scan know where is your cursor?

He could know, but it isn't necessary.
He could know where the cursor is when you start the background scan... no problem. But for what ? When you store the parse context

Ah, I see. You plan to store parse context for all lines. OK, that would work to some degree... Would be pretty unreliable for fast typers, but yes, not entirely insane.

Just for the record, current (and future) parser does not work this way. In fact, a pretty bad trick is used - the file is "ended" at the position of cursor and parser fails on "unexpected end of file" error, which is in fact expect behaviour Then the context is simply read from the parser. As simple as that:)

Mirek

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Wed, 26 Sep 2007 08:17:59 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 26 September 2007 00:24

Ah, I see. You plan to store parse context for all lines.

Not exactly, I'd store the parse context for all definitions, quite less amount of data.

Quote:
 OK, that would work to some degree... Would be pretty unreliable for fast typers, but yes, not entirely insane.

Well, if you find a programmer that is so fast typing to overcome the speed of a well done background parser, please send him to me, I've got some job for him

Quote:

Just for the record, current (and future) parser does not work this way. In fact, a pretty bad trick is used - the file is "ended" at the position of cursor and parser fails on "unexpected end of file" error, which is in fact expect behaviour Then the context is simply read from the parser. As simple as that:)

I understand this, but then you MUST do it in real time.
From my point of view, all you need is a data structure storing the function-classes-macro-ecc definitions along with line numbers where they're defined.
You keep 2 such structures, and swap them when the background task is done, an so forth.That's all. Then, all what you need to do in real time is to do a fast lookup in that data structure to find what you're typing. Quite simple and quick job, if data structure is well indexed.... much faster than to call the parser for each keypress and wait for him do complete the job.
BTW, the 'speed feeling' of users would be higher of an order of magnitude than full real-time approaches.

Well, again I've got no time to make some experiments with this idea... pity, because I find it promising.

Ciao

Max

p.s. The ctrl+ doesn't work for me.... do I miss something ?

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mirek on Wed, 26 Sep 2007 12:30:03 GMT

View Forum Message <> Reply to Message

mdelfede wrote on Wed, 26 September 2007 04:17luzr wrote on Wed, 26 September 2007 00:24

Ah, I see. You plan to store parse context for all lines.

Not exactly, I'd store the parse context for all definitions, quite less amount of data.

Quote:
 OK, that would work to some degree... Would be pretty unreliable for fast typers, but yes, not entirely insane.

Well, if you find a programmer that is so fast typing to overcome the speed of a well done background parser, please send him to me, I've got some job for him

He does not need to type that much fast to get false result. And the full parsing can take up to seconds.

Quote:
I understand this, but then you MUST do it in real time.
From my point of view, all you need is a data structure storing the function-classes-macro-ecc definitions along with line numbers where they're defined.

Not enough. You need to know in what function you are too and to have all local variables info. In fact, knowing in what expression and perhaps even statement you are is quite important too.

Quote:
You keep 2 such structures, and swap them when the background task is done, an so forth.

Hehe, sounds as seductive as all bad ideas in programming:)

Quote:
That's all. Then, all what you need to do in real time is to do a fast lookup in that data structure to find what you're typing.

Well, that is not enough. The more important is to find the context WHERE YOU ARE. In what method, what locals are currently defined.

Quote:
Quite simple and quick job, if data structure is well indexed.... much faster than to call the parser for each keypress and wait for him do complete the job.
BTW, the 'speed feeling' of users would be higher of an order of magnitude than full real-time approaches.

Well, do you feel that it is slow now?! What I am up to will be as fast...

Quote:
p.s. The ctrl+ doesn't work for me.... do I miss something ?

Ctrl+, - i mean key with comma... I mean, hold Ctrl and press comma key

Mirek

Subject: Re: Very first impressions and.... [FEATURE REQUESTS]
Posted by mdelfede on Wed, 26 Sep 2007 14:59:08 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 26 September 2007 14:30

Hehe, sounds as seductive as all bad ideas in programming:)

tsk !!! That's only 'cause I've got no time to try it...

Quote:

Well, that is not enough. The more important is to find the context WHERE YOU ARE. In what method, what locals are currently defined.

That's solved storing start-and-end lines of parsed context.

Quote:
Well, do you feel that it is slow now?! What I am up to will be as fast...

nope, I feel it VERY fast, NOW ! What I'm afraid about is that'll be no such fast AFTER !!! There's too much you want to add to your completion, and all that requires cpu time !

Quote:
Ctrl+, - i mean key with comma... I mean, hold Ctrl and press comma key

Ahhhhh ! I didn't notice the ',' !!!! I'll try it this night !

Well, I can't wait to see your results !

Ciao

Max