Subject: Why each package should reside in its own directory?
Posted by Novo on Sun, 13 Apr 2008 17:12:29 GMT
View Forum Message <> Reply to Message

Directory per project seems to be a strong constraint.

Why not just use a name of an UPP project file to name a project if there is more than one project file in a directory?

Otherwise it is hard to adapt old projects to TheIDE, especially when they are third-party projects and TheIDE is used as a build tool.

Please correct me if I'm wrong.

Subject: Re: Why each package should reside in its own directory?
Posted by Mindtraveller on Mon, 14 Apr 2008 06:17:58 GMT
View Forum Message <> Reply to Message

One dir per project came from years of experience with environments like MS Visual Studio or Borland C++ Builder, and complex projects developed under them. The main idea of IDE is creation of "real" (complex enough to be commercial) applications, not etudes or simple tests. So this way one dir per project (package) seems very natural. If you`ve developed commercial apps, you should know that it`s rather obvious solution.

Subject: Re: Why each package should reside in its own directory?
Posted by Novo on Mon, 14 Apr 2008 15:01:11 GMT
View Forum Message <> Reply to Message

I do have some experience in commercial software development

The problem is that the package system in its current state cannot be used to build non

projects, sometimes I need to improve these third-party libraries. That means that I cannot change structure of somebodies code because I want to keep that code up-to-date using different version control systems, and I cannot use TheIDE to build that code because it is non package-organized.

I also have approximately 50 my own libraries. Many of them are package-organized, but several reside in one directory. Restructuring that bunch of code just to be able to build it under TheIDE is quite painful.

TIA

Subject: Re: Why each package should reside in its own directory?
Posted by masu on Tue, 15 Apr 2008 09:13:16 GMT
View Forum Message <> Reply to Message

I would keep them separately and add includes and libs as needed.
Then you can update the external libs as you want and also use it within TheIDE.
It is equivalent of handling for example OpenGL libs within TheIDE.

You could even use TheIDE's command line feature to automate your building tasks (update external libs, rebuild them, recompile TheIDE project).

I do not see any disadvantages in doing so.

Matthias

Subject: Re: Why each package should reside in its own directory?
Posted by Novo on Tue, 15 Apr 2008 20:27:35 GMT
View Forum Message <> Reply to Message

masu wrote on Tue, 15 April 2008 05:13I would keep them separately and add includes and libs as needed.
Then you can update the external libs as you want and also use it within TheIDE.
It is equivalent of handling for example OpenGL libs within TheIDE.

GLCtrl is using a prebuilt version of OpenGL.

changing of one line of code can cause 15-30 minutes of recompilation. UPP files are easy to create and they work on Windows and Linux.

B.T.W. Creating of a package, which includes only one file, seems to be overkill. (I mean one cpp file per directory.)

Quote:
You could even use TheIDE's command line feature to automate your building tasks (update external libs, rebuild them, recompile TheIDE project).

I do not see any disadvantages in doing so.

Matthias

Subject: Re: Why each package should reside in its own directory?
Posted by Novo on Tue, 15 Apr 2008 21:39:05 GMT
View Forum Message <> Reply to Message

Novo wrote on Tue, 15 April 2008 16:27
Quote:
You could even use TheIDE's command line feature to automate your building tasks (update external libs, rebuild them, recompile TheIDE project).

I do not see any disadvantages in doing so.

Matthias

I found that in the manual.

An -h command-line key for umk would make that process much easier.

Putting command-line keys into first position would make umk even easier to use, because the very first thing I (and probably everybody else) would do is to call "umk -h". It is somewhat complicated to figure out that you have to call "umk assembly package build_method -h" to get a help screen.

Thanks for pointing out.

Subject: Re: Why each package should reside in its own directory?
Posted by Novo on Thu, 17 Apr 2008 04:16:37 GMT
View Forum Message <> Reply to Message

Novo wrote on Sun, 13 April 2008 13:12
Otherwise it is hard to adapt old projects to TheIDE, especially when they are third-party projects and TheIDE is used as a build tool.

I figured out that myself

The answer is:

Do not keep your C++ and project files in one basket (directory).

Putting project files somewhere else doesn't force you to create a bunch of subdirectories in your project and follow good design principles.

There is always a way to avoid good design ...

Subject: Re: Why each package should reside in its own directory?
Posted by mirek on Sun, 21 Sep 2008 18:34:44 GMT
View Forum Message <> Reply to Message

Novo wrote on Sun, 13 April 2008 13:12Directory per project seems to be a strong constraint.

Why not just use a name of an UPP project file to name a project if there is more than one project file in a directory?

Otherwise it is hard to adapt old projects to TheIDE, especially when they are third-party projects and TheIDE is used as a build tool.

Please correct me if I'm wrong.

Actually, it is a directory per package.

E.g. "ide" project has definitely much more than single directory...

And there is even more flexibility to it: Package directory can have subdirectories, which can either be packages too or sources from them can be directly referenced in package.

Also, sometimes when adopting 3rdparty sources, you can use approach demonstrated in plugin/png...

Mirek

Subject: Re: Why each package should reside in its own directory?
Posted by Novo on Sun, 21 Sep 2008 20:37:36 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 21 September 2008 14:34
Also, sometimes when adopting 3rdparty sources, you can use approach demonstrated in plugin/png...

Mirek

In case of plugin/png and other plugins you copy files into plugin/<pakage> directory. I do not want to do that because I want to keep original source code structure, and I want that because I want to be able to update third-party source code from original VCS (CVS, SVN, Mercurial, Monotone, etc).

Another issue with UPP-project files is that project-files use absolute absolute paths. That makes impossible to move project files between computers and different OS. Actually, it is possible, but you need to fix directory names all the time. It is complicated if you have a hundred of them.

Subject: Re: Why each package should reside in its own directory?
Posted by mirek on Sun, 21 Sep 2008 22:23:22 GMT
View Forum Message <> Reply to Message

Novo wrote on Sun, 21 September 2008 16:37luzr wrote on Sun, 21 September 2008 14:34
Also, sometimes when adopting 3rdparty sources, you can use approach demonstrated in plugin/png...

Mirek

In case of plugin/png and other plugins you copy files into plugin/<pakage> directory. I do not want to do that because I want to keep original source code structure, and I want that because I want to be able to update third-party source code from original VCS (CVS, SVN, Mercurial, Monotone, etc).

Why cannot your original source code structure start at package folder? (with U++ wrapper if needed).

Quote:
Another issue with UPP-project files is that project-files use absolute absolute paths. That makes impossible to move project files between computers and different OS.

Huh? The main purpose is to AVOID the need for absolute paths. All package file paths are package directory relative, UNLESS you use "Add any file" (which is only for the use in very specific situations).

Mirek

Subject: Re: Why each package should reside in its own directory?
Posted by Novo on Mon, 22 Sep 2008 04:25:54 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 21 September 2008 18:23

Why cannot your original source code structure start at package folder? (with U++ wrapper if needed).

Because they are big and have complicated structure.
Actually, as I wrote previously, I found a solution.
I can create a fake package structure with no source code in it. The source code resides somewhere else. Absolute and relative path seem to work with project files, but the ability to use environment variables in project files (like SOMETHING_ROOT) would make them even more flexible. (Probably, it is already possible, and I just do not know how to do that.)

Quote:
Quote:
Another issue with UPP-project files is that project-files use absolute absolute paths. That makes impossible to move project files between computers and different OS.

Huh? The main purpose is to AVOID the need for absolute paths. All package file paths are package directory relative, UNLESS you use "Add any file" (which is only for the use in very specific situations).

Mirek

Sorry, it passed a long time since I opened this discussion, so I forgot the details . I meant var-files. I believe they are assembly-files.

Basically, everything is fine with the assembly-package-etc structure. I like BLITZ and I'd like to convert all my projects to TheIDE. It is just not that easy because I have more than fifty of them. I cannot afford to wait for an hour till they all rebuild. So, I'm spending my time developing parsers and project-file generators.

Subject: Re: Why each package should reside in its own directory?
Posted by mirek on Mon, 22 Sep 2008 05:45:43 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 22 September 2008 00:25
Sorry, it passed a long time since I opened this discussion, so I forgot the details . I meant var-files. I believe they are assembly-files.

Well, but you need to specify some path in the end... With assemblies, you specify as little as possible...

Quote:
Basically, everything is fine with the assembly-package-etc structure. I like BLITZ and I'd like to convert all my projects to TheIDE. It is just not that easy because I have more than fifty of them. I cannot afford to wait for an hour till they all rebuild. So, I'm spending my time developing parsers and project-file generators.

I see.

Well, of course not in all cases is source-based conversion desirable or possible. I remember that we gave-up with SSL - there is simply too many files and configuration in sources...

Mirek
