Subject: Acceptflags and ".FLAG" syntax
Posted by dolik.rce on Wed, 02 Mar 2011 21:49:24 GMT
View Forum Message <> Reply to Message

Hi

Recent discussion with Lionel (chickenk) about how theide handles flags lead us to quite surprising finding, that I wasn't aware of at all. Actually it was mostly Lionel who figured this out and I just looked at the code to make sure it is really true

Consider a very simple GUI package, let's call it SimpleApp. This main package includes only CtrlLib. If you unfold the package dependencies into a tree, it looks like this:SimpleApp

```
 +--CtrlLib
    +--PdfDraw
    |  +--Draw
    |     +--Core
    |        +--Plugin/z
    +--CtrlCore
       +--Draw
       |   +--Core
       |      +--Plugin/z
       +--Plugin/bmp
       |  +--Draw
       |     +--Core
       |        +--Plugin/z
       +--RichText
          +--Draw
          |  +--Core
          |     +--Plugin/z
          +--Plugin/png
             +--Draw
             |  +--Core
             |     +--Plugin/z
             +--Core
                +--Plugin/z
```

Now the interesting part comes: If you compile this app with flag ".NOGTK", theide will search the dependencies for accepted flags, and add NOGTK to each package that has at least on child with acceptflags = NOGTK. In our SimpleApp this actually means that all the packages with exception of Core and Plugin/z will have NOGTK flag.

Until now I always thought that the behavior is to set flag prefixed with a dot only to main package and to packages which explicitly accept the flag. It would be IMHO much more efficient in terms of recompiling as little code as possible (that was the rationale behind this right?), to limit it this way. For our example package only CtrlLib, CtrlCore and the main package should need to know whether .NOGTK is set.

The only reason for this "reverse inheritance" I can imagine is that the packages might have different interfaces based on the flag, so that packages depending on it would need to know which

interface to use. But AFAIK for the two flags most commonly used with dotted syntax (NOGTK and USEMALLOC), it is not the case. And even if this conditional interface was used somewhere, I think it would be perfectly fair to require filling in the accept field in package manager for any package using such interface.

Also currently, Draw accepts NOGTK even though it doesn't use it at all (that is probably some artifact from past refactoring of Draw). And oppositely, CtrlCore uses flagNOGTK and doesn't accept it, but due to Draw including it and the inheritance it is not noticeable. Fixing these two packages reduces the amount of packages compiled with NOGTK to only 3 in our example above (CtrlCore,CtrlLib and SimpleApp), so the problem would be invisible for this particular case, but in more complex example it would still cause inefficiency...

So, what do you think about this? It is probably question aimed primarily to Mirek, but I wonder if it is surprising to the rest of you as much as it was to me

Best regards,
Honza

Subject: Re: Acceptflags and ".FLAG" syntax
Posted by chickenk on Thu, 03 Mar 2011 08:52:58 GMT
View Forum Message <> Reply to Message

I personally agree with Honza, in that accept flags are intuitively expected to be non-heritable, and that dotted flags should only be 'accepted' by packages explicitely accepting them.

The good point is that it would make the parsing code easier and not more complex...

Another related question: Is the dotted flag and accept flags usage currently documented somewhere? That would be a nice addition if not already done. If already present, we should also update it to reflect the behaviour we will eventually choose.

Lionel
