Subject: what technique does U++ use to reduce code bloat from too many template instantiations?
Posted by fudadmin on Mon, 12 Dec 2005 20:02:20 GMT
View Forum Message <> Reply to Message

what technique does U++ use to reduce code bloat from too many template instantiations?

Subject: Re: what technique does U++ use to reduce code bloat from too many template instantiations?
Posted by mirek on Mon, 12 Dec 2005 20:44:02 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Mon, 12 December 2005 15:02what technique does U++ use to reduce code bloat from too many template instantiations?

A good question Actually, some of that is reduced by smart linkers (they are able to merge the same function).

Anyway, the really great reduction is result of U++ containers, esp. two things: Array container has really low footprint (it needs about 200 bytes pre instantiation) and Map containers are composites of Index and Array or Vector, with quite a lot Index hashing code implemented in non-template .cpp.

To explain composition issue: VectorMap<String, int> and VectorMap<String, String> are actually compostions of Index<String> and either Vector<int> and Vector<String> and "composition" code is really minimal. Compare with STL that generates totally different code for map<string, int>, map<string, string>, vector<string> and vector<int>. With more types involved, advantage on U++ side increases....

Subject: Re: what technique does U++ use to reduce code bloat from too many template instantiations?
Posted by thierry on Sun, 10 Sep 2006 16:32:22 GMT
View Forum Message <> Reply to Message

Could it possible that theIDE can help still a little with encouraging the following practice in case of not so clever linker (relying only on weakreferences) that anyway can't avoid the inline code bloat:

T.h : no inline code for templates
(well just recommand and do it)

T.tpp: template function implementation

T_mytype.cpp: one for each explicit instanciation
#include "T.tpp"

template T<mytype>;

This is really annoying for a programmer to create one cpp per instanciation, but this is where an ide can help a lot to save developer to follow this guideline with generating the T_mytype.cpp for him, with maybe having a repository of all instanciations at least by packages... Then the linker shall be only smart enough not to link twice the same code.

This would not only save code bloat, but also compilation time, so far template would be compile only once, and changes in .tpp won't affect other modules

---

Subject: Re: what technique does U++ use to reduce code bloat from too many template instantiations?
Posted by mirek on Sun, 10 Sep 2006 16:59:29 GMT
View Forum Message <> Reply to Message

Hm, I am not sure there are toolchains able to compile U++ (which itself is in fact limited to gcc 3.1+ or msc 7.1+ or EDG frontends) and not support this kind of optimization.

Mirek
