Subject: error: duplicate symbol: std::__throw_bad_alloc()
Posted by Xemuth on Sat, 22 Aug 2020 16:19:23 GMT
View Forum Message <> Reply to Message

When using the Bazaar/plugin/assimp package, I got an error at linking (only when using CLANG) (MSVS work perfect) :

```
(): Linking has failed
(): lld-link: error: duplicate symbol: std::__throw_bad_alloc()
(): >>> defined at Core.a(heap.o)
(): >>> defined at libc++.a(new.cpp.obj)
():
(): lld-link: error: duplicate symbol: operator new(unsigned long long)
(): >>> defined at Core.a(heap.o)
(): >>> defined at libc++.a(new.cpp.obj)
 ...
```

The only way I have found to get around this issu is to put the flag "USEMALLOC" but I don't think it's a good way.
I don't know if this"problem" is related to Upp or assimp.

Someone can help me by explaining what's happening ?

Thanks

Subject: Re: error: duplicate symbol: std::__throw_bad_alloc()
Posted by koldo on Mon, 24 Aug 2020 08:49:42 GMT
View Forum Message <> Reply to Message

Sorry Xemuth

I've tried my best, unsuccessfully, trying to remove this duplicate symbol error.

Subject: Re: error: duplicate symbol: std::__throw_bad_alloc()
Posted by mirek on Mon, 24 Aug 2020 09:00:58 GMT
View Forum Message <> Reply to Message

I believe it can still be related to:

https://github.com/mstorsjo/llvm-mingw/issues/91

Subject: Re: error: duplicate symbol: std::__throw_bad_alloc()

Posted by koldo on Mon, 24 Aug 2020 10:03:13 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 24 August 2020 11:00I believe it can still be related to:

https://github.com/mstorsjo/llvm-mingw/issues/91
Yes sure.
However I am clueless. Even commenting all new/delete overloading in plugin/assimp, and rebuilding all, the error remains...

Subject: Re: error: duplicate symbol: std::__throw_bad_alloc()
Posted by Xemuth on Mon, 24 Aug 2020 14:44:14 GMT
View Forum Message <> Reply to Message

What's the impact of flag "USEMALLOC" is using malloc instead of new/delete in U++ slower ?!

Subject: Re: error: duplicate symbol: std::__throw_bad_alloc()
Posted by mirek on Mon, 24 Aug 2020 14:58:20 GMT
View Forum Message <> Reply to Message

Xemuth wrote on Mon, 24 August 2020 16:44What's the impact of flag "USEMALLOC" is using malloc instead of new/delete in U++ slower ?!

I hope so

Mirek

Subject: Re: error: duplicate symbol: std::__throw_bad_alloc()
Posted by Xemuth on Mon, 24 Aug 2020 16:17:51 GMT
View Forum Message <> Reply to Message

I mean do using malloc/free instead of new/delete is slower ? If no then this error is not a priority ^^

Subject: Re: error: duplicate symbol: std::__throw_bad_alloc()
Posted by mirek on Mon, 24 Aug 2020 20:14:59 GMT
View Forum Message <> Reply to Message

Xemuth wrote on Mon, 24 August 2020 18:17I mean do using malloc/free instead of new/delete is slower ? If no then this error is not a priority ^^

I mean, yes, it should. I have spent a lot of time optimizing our own allocator...

Subject: Re: error: duplicate symbol: std::__throw_bad_alloc()
Posted by koldo on Tue, 25 Aug 2020 05:29:17 GMT
View Forum Message <> Reply to Message

The ASSIMP case is a good example of U++ efficiency (thanks to Mirek).
The same ASSIMP example compiled with MSC without USEMALLOC is much faster than with USEMALLOC.
Now it would be great to get CLANG working without USEMALLOC.

Subject: Re: error: duplicate symbol: std::__throw_bad_alloc()
Posted by mirek on Tue, 25 Aug 2020 07:40:20 GMT
View Forum Message <> Reply to Message

koldo wrote on Tue, 25 August 2020 07:29The ASSIMP case is a good example of U++ efficiency (thanks to Mirek).
The same ASSIMP example compiled with MSC without USEMALLOC is much faster than with USEMALLOC.
Now it would be great to get CLANG working without USEMALLOC.

Just for the info: I have looked into it, removed all funny new/delete related code in ASSIMP, but it still does not work.

Subject: Re: error: duplicate symbol: std::__throw_bad_alloc()
Posted by koldo on Tue, 25 Aug 2020 07:42:12 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 25 August 2020 09:40koldo wrote on Tue, 25 August 2020 07:29The ASSIMP case is a good example of U++ efficiency (thanks to Mirek).
The same ASSIMP example compiled with MSC without USEMALLOC is much faster than with USEMALLOC.
Now it would be great to get CLANG working without USEMALLOC.

Just for the info: I have looked into it, removed all funny new/delete related code in ASSIMP, but it still does not work.
Yes, I did that and obtained the same result