It is trivial to get rid of 2 of the 3 defines. Using:

```c
#ifdef __CYGWIN__
#define GMX_CYGWIN
#endif

/* We currently don't support MingW. And ICC also defines it */
#ifdef __MSC_VER
#define GMX_NATIVE_WINDOWS
#endif
```

The question is what do we want to do with GMX_CXX11. Only trajectoryanalysis/analysismodule.h, gromacs/commandline/cmdlineprogramcontext.h, gromacs/analysisdata/modules/histogram.h, gromacs/options/abstractoption.h, and gromacs/onlinehelp/helptopicinterface.h are public headers which also use our unique_ptr. Does it make sense to only use shared_ptr for public API but keep using our unique_ptr internally?

The main advantages:
- Removes GMX_CXX11 dependencies from installed headers.
- Improves memory management, where the memory is deallocated in the same library/executable that allocated it (through the shared_ptr deleter that gets created during construction).

The only disadvantage is that this blurs the ownership of the objects slightly.

There are only two other instances of gmx_unique_ptr left in installed headers, and those should be relatively easy to remove as well with a bit of refactoring.

Related to #1454.

Change-Id: I1f694b30e4f8129f5d0100115c0968a44e927ca0
This has the added benefit that now the whole onlinehelp module is free from public headers, making the interface cleaner. Adjust doxygen documentation accordingly.

Part of #1454
Change-Id: lfb557da86fb0d422650029812d30cc6766730fb9

**Revision e9b414f3 - 07/08/2014 03:52 PM - Teemu Murtola**

Remove gmx_unique_ptr from options public headers

There was a single factory function in the generic option interface that created objects and passed ownership to the caller. The implementation for this method always follows the same pattern, and there only a single caller, so it is easy to make it exception-safe while returning a plain pointer. As a side effect, it removes some boilerplate code from all the implementations for this method.

There are more complicated alternatives that avoid the plain pointer, based on creating a helper class that can only be forward declared in the installed headers, but I don't think those are worth the extra code.

This removes the last use of gmx_unique_ptr in an installed header.

Part of #1454
Change-Id: li948973d32042d388341265d5db6e1bbcf077db8

**Revision 1078df1d - 07/08/2014 04:16 PM - Teemu Murtola**

Move GMX_CXX11 to config.h

Now that uniqueptr.h is not used by any installed header, do not install it. Move GMX_CXX11 that was used only by this file to config.h now that config.h can be freely included in that file.

Part of #1454
Change-Id: l4e4c2f23057962f2dc1f96c309c10d5c5ce0c2c2a74

**Revision 22d74432 - 07/09/2014 07:09 AM - Teemu Murtola**

Clean up gmx_header_config

- Remove some probably unnecessary uses for GMX_CYGWIN.
- Move GMX_CYGWIN to config.h as it is not used in installed headers (maybe it would be better to get rid of even the remaining uses in favor of some other mechanism).
- Remove gmx_header_config_gen.h and all complexity that supports it as there is nothing generated there any longer.
- GMX_NATIVE_WINDOWS could possibly be moved somewhere else, but currently it is only used in two headers, so possibly even better would be to just get rid of those two.

Closes #1454
Change-Id: ld27370a007eda30b7c803a3c14aa687d43555db8

**Revision 68fc3130 - 01/20/2015 06:45 PM - Mark Abraham**

Removed gmx_header_config.h

Now we no longer install a header just to get platform independence for functionality that is not actually part of the the installed API.

Moved DIR_SEPARATOR into its own non-installed header.

Moved snprintf MSVC-workaround define into its own non-installed header.

Moved GMX_NATIVE_WINDOWS definition to config.h

Removed checks related to the use of gmx_header_config.h
It would be nice if you provided some justification for why you want to do this. I don't necessarily disagree, but just recently in 
https://gerrit.gromacs.org/#/c/3157/ you seemed to propose exactly the opposite, tying the headers much more tightly to one instance of the library. 
Also, what would be the plan to deal with other config.h dependencies in installed headers? Currently, they are in a much better shape than what they used to be, though, with majority of the issues coming from thread_mpi. It should not be very difficult to get rid of GMX_CYGWIN and GMX_NATIVE_WINDOWS in installed headers completely, either, which would be even better.

For the public API, I think it would make a lot of sense to only use shared_ptr to remove that dependency of the public API on the internal configuration options. That may ripple through to a lot of the internal implementation, though, since it is not possible to convert between the pointer types. If it ends up removing a large fraction of current unique_ptr instances, the question then is, is it worth to keep the unique_ptr at all, with all the complexity it brings, or just replace it with shared_ptr everywhere? Or alternatively, should we use raw pointers at some parts to avoid these smart pointers on the API? But I won't object to such a change, just wanted to bring up these points, since you were quite aggressively pushing for use of smart pointers in #887.

While working on #1452 I considered the TODO you added to gmx_header_config_gen.h and noticed that part of the problem is trivial. So the reason is mostly because of the TODO you put there ;-). But I agree with your TODO. I think if we want a API the API should ideally not depend on any ifdef.

I opened this task to ask what we wanted to do with the 3rd. You are right that I don't know what the best approach is, and thus I suggested different solutions in the past and here.

I think tightly coupling the header to the library is an option if we really have to make the API different. And as long as e.g. only GMX_DOUBLE is left than we could do it nicely the same as fftw (having both fftw3 and fftw3f). But I think we agree that it would be helpful if the number of variables with change the API is limited as much as possible.

I don't see a big advantage of getting rid of variables like GMX_CYGWIN or GMX_NATIVE_WINDOWS if there don't come from cmake but from some preprocessor logic as suggested in the question. Especially if they don't prevent compiling libgromacs with one compiler and using it with a different one. What is the reason you would like to remove them altogether?

shared_ptr has a constructor for unique_ptr thus that direction is no problem. But the other direction isn't possible. But even with just being able to convert it in one direction it might help that it doesn't ripple as far. The question is how far would it ripple. But it probably still ripples too far to make this a good idea. I think there is an advantage of keeping unique_ptr which is bigger than getting rid of GMX_CXX11. So my idea was probably not good.

I think having thought about it more I think replacing GMX_CYGWIN and GMX_NATIVE_WINDOWS as shown above and tightly coupling for GMX_DOUBLE and GMX_CXX11 is the best solution. What do you think?

Another solution: https://gerrit.gromacs.org/#/c/3750/ (forgot the ref number so Review Bot didn't post it)

Gerrit received a related patchset '1' for Issue #1454.
Uploader: Teemu Murtola (teemu.murtola@gmail.com)
Change-Id: 1f694b30e4812995d10115c0968a444e927ca0
Gerrit URL: https://gerrit.gromacs.org/3768

Another solution: https://gerrit.gromacs.org/#/c/3750/ (forgot the ref number so Review Bot didn't post it)

Gerrit received a related patchset '1' for Issue #1454.
Uploader: Teemu Murtola (teemu.murtola@gmail.com)
Change-Id: le4c2f230579e62fdcd199c309c10d5c0ec0c2a74
Gerrit URL: https://gerrit.gromacs.org/3779

Gerrit received a related patchset '1' for Issue #1454.
Uploader: Teemu Murtola (teemu.murtola@gmail.com)
Change-Id: ld27370a007eda30b7c803a3c14aa687d43555db8
Gerrit URL: https://gerrit.gromacs.org/3779
Now there's a single define left in gmx_header_config.h, and the generated header is gone. It should be straightforward to get rid of this as well in installed headers and move to config.h, but that's outside the scope of this issue.

As written, the task was complete a while ago. Are there remaining related issues of avoiding installing stuff that isn't very smart to install?

Resolved well enough by Mark's latest change.