CoCoALib - Support #1700

boost_1_80_0

14 Oct 2022 10:16 - Anna Maria Bigatti

Status:	Closed	**Start date:**	14 Oct 2022
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Documentation	**Estimated time:**	3.99 hours
Target version:	CoCoALib-0.99800	**Spent time:**	3.95 hours

Description

Problems and solutions

Related issues:

Related to CoCoA-5 - Design #1697: Specifying BOOST in configure	**Closed**	**18 Aug 2022**
Related to CoCoA-5 - Design #1696: Which BOOST libs are actually needed?	**Closed**	**18 Aug 2022**
Related to CoCoA-5 - Bug #1695: boost-find-lib.sh	**Closed**	**16 Aug 2022**
Related to CoCoALib - Design #1528: MacOS compilation with clang	**Closed**	**30 Oct 2020**
Related to CoCoA-5 - Support #632: MacOSX: many warnings with compiler CLANG/...	**Closed**	**03 Oct 2014**
Related to CoCoALib - Bug #1600: Detect updated versions of external libs	**In Progress**	**14 Jun 2021**

History

#1 - 14 Oct 2022 10:18 - Anna Maria Bigatti

- % Done changed from 0 to 80

- Estimated time set to 2.00 h

Problems with

```
make[2]: *** No rule to make target `../../configuration/ExternalLibs/include/boost/system/detail/to_std_categ
ory.hpp', needed by `AST.o'.  Stop.
```

?

just run make veryclean to remake dependencies: there is no to_std_category.hpp in boost_1_80_0

#2 - 14 Oct 2022 10:18 - Anna Maria Bigatti

- Related to Design #1697: Specifying BOOST in configure added

#3 - 14 Oct 2022 10:18 - Anna Maria Bigatti

- Related to Design #1696: Which BOOST libs are actually needed? added

#4 - 14 Oct 2022 10:18 - Anna Maria Bigatti

- Related to Bug #1695: boost-find-lib.sh added

#5 - 14 Oct 2022 11:45 - John Abbott

- Status changed from New to Resolved

I wonder if there is a simple away to warn users of potential problems?

A similar problem could arise with any of the external libs, and the solution is no doubt to rebuild dependencies -- and this should normally be done by re-running configure (which may include some other "sanity checks" for external libs).

Here is an idea:
for each ext lib configure computes a check-sum for some header file in that ext lib, and saves the result in autoconf.mk. Then make can verify that the check-sums are correct -- if not, a warning with a delay, can be produced before proceeding with compilation.
[the delay gives the user a chance to hit Ctrl-C]

#6 - 14 Oct 2022 12:32 - Anna Maria Bigatti

I still have some warnings, but it compiles and works fine

```
bash-3.2$ g++  -std=c++14  -Wall  -pedantic    -O2    -I../../include  -I ../../configuration/ExternalLibs/inc
lude -g  -DCoCoA_WITH_READLINE AST.o Lexer.o Main.o Interpreter.o LineProviders.o Parser.o CoCoALibSupplement.
o BuiltInFunctions.o BuiltInFunctions-CoCoALib.o BuiltInFunctionsVarArgs-CoCoALib.o BuiltInOneLiners-CoCoALib.
o BuiltInFunctions-Frobby.o BuiltInFunctions-GFan.o BuiltInFunctions-GSL.o BuiltInFunctions-MathSAT.o BuiltInF
unctions-Normaliz.o globals.o OnlineHelp.o VersionInfo.o Banner.o CompilationDate.o -o CoCoAInterpreter ../../
lib/libcocoa.a -L../../configuration/ExternalLibs/lib  -lfrobby-symlink  -lgfan-symlink  -lcddgmp-symlink    -
lmathsat-symlink  -lnormaliz-symlink    -lgmpxx-symlink  -lgmp-symlink  /Users/bigatti/0.99/boost_1_80_0/lib/l
ibboost_filesystem.a  /Users/bigatti/0.99/readline-7.0/lib/libreadline.a  -ltermcap  -lpthread
ld: warning: PIE disabled. Absolute addressing (perhaps -mdynamic-no-pic) not allowed in code signed PIE, but 
used in ___gmpn_divexact_1 from ../../configuration/ExternalLibs/lib/libgmp-symlink.a(dive_1.o). To fix this w
arning, don't compile with -mdynamic-no-pic or link with -Wl,-no_pie
ld: warning: direct access in function 'boost::system::error_code::what() const' from file '/Users/bigatti/0.9
9/boost_1_80_0/lib/libboost_filesystem.a(exception.o)' to global weak symbol 'boost::system::error_code::locat
ion() const::loc' from file 'AST.o' means the weak symbol cannot be overridden at runtime. This was likely cau
sed by different translation units being compiled with different visibility settings.
ld: warning: direct access in function 'boost::system::error_code::what() const' from file '/Users/bigatti/0.9
9/boost_1_80_0/lib/libboost_filesystem.a(exception.o)' to global weak symbol 'boost::system::error_code::locat
ion() const::loc' from file 'AST.o' means the weak symbol cannot be overridden at runtime. This was likely cau
sed by different translation units being compiled with different visibility settings.
ld: warning: direct access in function 'boost::system::error_code::what() const' from file '/Users/bigatti/0.9
9/boost_1_80_0/lib/libboost_filesystem.a(exception.o)' to global weak symbol 'boost::system::error_code::locat
ion() const::loc' from file 'AST.o' means the weak symbol cannot be overridden at runtime. This was likely cau
sed by different translation units being compiled with different visibility settings.
```

#7 - 14 Oct 2022 16:23 - Anna Maria Bigatti

- Related to Design #1528: MacOS compilation with clang added

#8 - 14 Oct 2022 16:24 - Anna Maria Bigatti

- Related to Support #632: MacOSX: many warnings with compiler CLANG/LLVM when compiling BOOST code. added

#9 - 12 Nov 2022 13:53 - John Abbott

On my MacBook I get over 100000 (one hundred thousand) lines of warnings when compiling.
I think there is some mild incompatibility between BOOST and the compiler (g++ on MacOS)

My proposal in note 5 is not unreasonable, but is it really KISS?

#10 - 12 Nov 2022 13:54 - John Abbott

- Related to Bug #1600: Detect updated versions of external libs added

#11 - 23 Nov 2022 17:36 - John Abbott

- Status changed from Resolved to Closed

- Assignee set to John Abbott

- % Done changed from 80 to 100

- Estimated time changed from 2.00 h to 3.99 h

JAA is closing this issue because it is a **near duplicate of issue** #1600**.**

I'll copy note-5 over to issue #1600, just to keep everything together.