How to update the version of clang supported by clangml: example going from clang-3.8.0 to clang-3.9.0

francois.berenger@inria.fr

Thu Nov 17 2016

1 Prerequisite

1.1 OS

Recommended is a recent Ubuntu Linux version or a Mac OSX with brew[0] installed.

1.2 OCaml

You should have OCaml v4.03.0 installed by OPAM[1] as a non system switch.

```
opam switch 4.03.0
eval ‘opam config env‘
```

1.3 Clangml for clang-3.8.0

```
cd ~/src
git clone https://github.com/Antique-team/clangml.git
cd clangml
git checkout clang_3.8 # branch for clang-3.8.0
opam pin -n add clangml $PWD
opam remove clangml
opam depext -i clangml # install all dependencies
```

1.4 Compile and test it

Simple test:

```
make echo 'int main() { return 0; }' > test.c
./processor.native test.c
```

You should see the clang AST being printed out.
Much more thorough test:

```
# get a published version of MemCAD
cd ~/src
wget https://github.com/Antique-team/memcad/archive/v1.0.0.tar.gz
tar xzf v1.0.0.tar.gz
cd memcad-1.0.0
opam pin -n add memcad $PWD
make
# launch the regression test suite
make prtp
```

All memcad tests should pass without any error.

2 Switch to clang-3.9.0

2.1 Install clang-3.9.0

```
# remove previous clang version
sudo apt-get remove clang-3.8 libclang-3.8-dev llvm-3.8-dev
# get the new version
mkdir ~/usr
cd ~usr
wget \ http://llvm.org/releases/3.9.0/\
clang+llvm-3.9.0-x86_64-linux-gnu-ubuntu-16.04.tar.xz
tar \
xJf clang+llvm-3.9.0-x86_64-linux-gnu-ubuntu-16.04.tar.xz
mv clang+llvm-3.9.0* clang39
```

Do some setup so that the newly installed commands clang++, llvm-config and clang-3.9 can all be found in your PATH.

3 General guidelines regarding the upgrade

Each compilation error will force you to update the file clang/clang/ast.ml (AST nodes).

For each AST node that you need to add in this file, you will need to update the file clang/clang/pp.ml accordingly (pretty printing of AST nodes).

If some [name]Type AST node was added, the file clang/clang/types.ml needs to be updated accordingly.

If some [name]Decl AST node was added, the file plugin/c++/OcamlVisitor/Decl.cpp needs to be modified accordingly.

If some [name]Expr AST node was added, the file plugin/c++/OcamlVisitor/Expr.cpp needs to be updated.

If some [name]Stmt or [name]Directive AST node was added, the file plugin/c++/OcamlVisitor/Stmt.cpp needs to be updated.

If some [name]Type AST node was added, the file plugin/c++/OcamlVisitor/Type.cpp needs to be updated.

If a [name]Type AST node is added/updated, the file plugin/c++/OCamlVisitor/TypeLoc.cpp also needs to be updated.

If some enums were modified in clang, some modifications might be needed in plugin/c++/clang_enums.cpp and plugin/c++/clang_enums.h.

To modify one of the previously mentioned file: look at the doxygen documentation of the new/modified AST node first. Then, modify any impacted file by taking example from code that was already in the file previously. It is recommended to modify clang/clang/ast.ml only step by step: do one modification at a time in there, then modify all the other impacted files until they compile before introducing one more change in ast.ml.

4 Update clangml until it fully compiles

Here is the list of files that are impacted by this clang update (they were discovered by actually doing the upgrade):

1. clang/clang/api.ml
2. clang/clang/ast.ml
3. clang/clang/pp.ml
4. myocamlbuild.ml
5. plugin/c++/OCamlVisitor/Decl.cpp
6. plugin/c++/OCamlVisitor/Expr.cpp
7. plugin/c++/OCamlVisitor/Stmt.cpp
8. plugin/c++/clang_enums.cpp plugin/c++/clang_ranges.h

You will have to change something in each of them.

Here is a preview of what you will have to do, file by file.

myocamlbuild.ml: Update the clang version number. Remove -Werror=date-time from the cxxflags with a sed command. We need to add -I‘llvm_config –includedir‘ to the cxxflags. In ldflags, we need to add -lLLVMCore. After those changes, we should be able to start compiling clangml using ocamlbuild (invoked by 'make').

clang/clang/api.ml: Update c_compiler version number.

clang/clang/ast.ml:

You should do them one by one, but here are all the AST nodes to add:

1. ATK_attr_swiftcall
2. ATK_preserve_most
3. ATK_preserve_all
4. BT_Float128

5. BT_OCLImage1dRO
6. BT_OCLImage1dArrayRO
7. BT_OCLImage1dBufferRO
8. BT_OCLImage2dRO
9. BT_OCLImage2dArrayRO
10. BT_OCLImage2dArrayDepthRO
11. BT_OCLImage2dArrayMSAARO
12. BT_OCLImage2dArrayMSAADepthRO
13. BT_OCLImage2dDepthRO
14. BT_OCLImage2dMSAARO
15. BT_OCLImage2dMSAADepthRO BT_OCLImage3dRO
16. BT_OCLImage1dWO
17. BT_OCLImage1dArrayWO
18. BT_OCLImage1dBufferWO
19. BT_OCLImage2dWO
20. BT_OCLImage2dArrayWO
21. BT_OCLImage2dArrayDepthWO
22. BT_OCLImage2dArrayMSAAWO
23. BT_OCLImage2dArrayMSAADepthWO
24. BT_OCLImage2dDepthWO
25. BT_OCLImage2dMSAAWO
26. BT_OCLImage2dMSAADepthWO
27. BT_OCLImage3dWO
28. BT_OCLImage1dRW
29. BT_OCLImage1dArrayRW BT_OCLImage1dBufferRW
30. BT_OCLImage2dRW
31. BT_OCLImage2dArrayRW
32. BT_OCLImage2dArrayDepthRW

33. BT_OCLImage2dArrayMSAARW
34. BT_OCLImage2dArrayMSAADepthRW
35. BT_OCLImage2dDepthRW
36. BT_OCLImage2dMSAARW
37. BT_OCLImage2dMSAADepthRW BT_OCLImage3dRW
38. CXXInheritedCtorInitExpr
39. ObjCAvailabilityCheckExpr
40. OMPDistributeParallelForDirective
41. OMPDistributeParallelForSimdDirective
42. OMPDistributeSimdDirective
43. OMPTargetEnterDataDirective
44. OMPTargetExitDataDirective
45. OMPTargetParallelDirective
46. OMPTargetParallelForDirective
47. OMPTargetParallelForSimdDirective
48. OMPTargetUpdateDirective
49. ConstructorUsingShadowDecl
50. OMPCapturedExprDecl
51. OMPDeclareReductionDecl
52. PragmaCommentDecl
53. PragmaDetectMismatchDecl

clang/clang/pp.ml: Each of the previously listed new AST nodes should be reflected into pp.ml.

plugin/c++/clang_ranges.h: The class clang::DesignatedInitExpr no more has the methods designators_begin and designators_end, we must find the new method in clang-3.9.0 doxygen documentation and use it.

Here is the doxygen doc for this class in clang-4.0.0 http://clang.llvm.org/doxygen/classclang_1_1Designate (I can't find online the doxygen doc for clang-3.9.0; you can download it however).

plugin/c++/clang_enums.cpp: The newly introduced AttributedTypeKind (new ATK_* nodes in ast.ml) must be reflected into that file. The new OpenCL ast nodes (OCL* in ast.ml) must also be reflected in here.

plugin/c++/OCamlVisitor/Decl.cpp: We can ignore the new AST nodes related to OpenMP or C++ since we are interested only in the C AST. However, maybe one day someone will want to support those AST nodes. To ignore them, use UNIMP_DECL (AST_NODE). All new AST nodes in ast.ml ending in *Decl must be handled in here.

plugin/c++/OCamlVisitor/Expr.cpp All new AST nodes in ast.ml ending in *Expr must be handled in here. Use UNIMP_STMT (AST_NODE) to ignore each.

plugin/c++/OCamlVisitor/Stmt.cpp All new AST nodes in ast.ml ending in *Directive must be handled in here.

4.1 Tag the new version of clangml

Once you are done, create a new git branch so that people can see later on what was done to support this new clang version.

Tag and release the software.

Create a new opam package for that version.

4.2 Previous upgrades examples

The branch (no more maintained) that was working with clang-3.4 is kept in git: https://github.com/Antique-team/clangml/tree/clang_3.4

For clang-3.5 it is here: https://github.com/Antique-team/clangml/tree/clang_update_3.5

For clang-3.6: https://github.com/Antique-team/clangml/tree/clang_update_3.6

So, for example, if you want to see the diff between clangml for clang-3.4 and clangml for clang-3.5, you can use git/github to see a graphical diff:

https://github.com/Antique-team/clangml/compare/clang_3.4...clang_update_3.5

5 References

[0] http://brew.sh/

[1] https://opam.ocaml.org/