NAME

llvm-cxxmap - Mangled name remapping tool

SYNOPSIS

llvm-cxxmap [*options*] *symbol-file-1 symbol-file-2*

DESCRIPTION

The **llvm-cxxmap** tool performs fuzzy matching of C++ mangled names, based on a file describing name components that should be considered equivalent.

The symbol files should contain a list of C++ mangled names (one per line). Blank lines and lines starting with **#** are ignored. The output is a list of pairs of equivalent symbols, one per line, of the form

<symbol-1> <symbol-2>

where **<symbol-1>** is a symbol from *symbol-file-1* and **<symbol-2>** is a symbol from *symbol-file-2*. Mappings for which the two symbols are identical are omitted.

OPTIONS

-remapping-file=file, -r=file

Specify a file containing a list of equivalence rules that should be used to determine whether two symbols are equivalent. Required. See *REMAPPING FILE*.

-output=file, -o=file

Specify a file to write the list of matched names to. If unspecified, the list will be written to stdout.

-Wambiguous

Produce a warning if there are multiple equivalent (but distinct) symbols in *symbol-file-2*.

-Wincomplete

Produce a warning if *symbol-file-1* contains a symbol for which there is no equivalent symbol in *symbol-file-2*.

REMAPPING FILE

The remapping file is a text file containing lines of the form

fragmentkind fragment1 fragment2

where **fragmentkind** is one of **name**, **type**, or **encoding**, indicating whether the following

mangled name fragments are <*name*>s, <*type*>s, or <*encoding*>s, respectively. Blank lines and lines starting with **#** are ignored.

Unmangled C names can be expressed as an **encoding** that is a (length-prefixed) <*source-name*>:

```
# C function "void foo_bar()" is remapped to C++ function "void foo::bar()".
encoding 7foo_bar _Z3foo3barv
```

For convenience, built-in <substitution>s such as **St** and **Ss** are accepted as <name>s (even though they technically are not <name>s).

For example, to specify that **absl::string_view** and **std::string_view** should be treated as equivalent, the following remapping file could be used:

```
# absl::string_view is considered equivalent to std::string_view
type N4absl11string_viewE St17basic_string_viewIcSt11char_traitsIcEE

# std:: might be std::__1:: in libc++ or std::__cxx11:: in libstdc++
name St St3__1
name St St7__cxx11
```

NOTE:

Symbol remapping is currently only supported for C++ mangled names following the Itanium C++ ABI mangling scheme. This covers all C++ targets supported by Clang other than Windows targets.

AUTHOR

Maintained by the LLVM Team (https://llvm.org/).

COPYRIGHT