目 录

[1. 概述 1](#_Toc574499342)

[2. AST文件AST常量 1](#_Toc173255506)

[2.1. classes 1](#_Toc1835737919)

[2.1.1. struct clang::serialization::LocalRedeclarationsInfo 1](#_Toc1436136773)

[2.1.2. struct clang::serialization::ObjCCategoriesInfo 1](#_Toc1269206864)

[2.1.3. class clang::serialization::DeclarationNameKey 1](#_Toc2132590301)

[2.2. 枚举类型 1](#_Toc704217158)

[2.2.1. enum clang::serialization::PredefinedTypeIDs 1](#_Toc1912285777)

[2.2.2. enum clang::serialization::TypeCode 1](#_Toc2115410994)

[2.2.3. enum clang::serialization::SpecialTypeIDs 1](#_Toc1738396584)

[2.2.4. enum clang::serialization::PredefinedDeclIDs 1](#_Toc1774607204)

[2.2.5. enum clang::serialization::DeclCode 1](#_Toc45673760)

[2.2.6. enum clang::serialization::StmtCode 1](#_Toc101318862)

[2.2.7. enum clang::serialization::DesignatorTypes 1](#_Toc710473176)

[2.2.8. enum clang::serialization::CtorInitializerType 2](#_Toc933771798)

[2.2.9. enum clang::serialization::CleanupObjectKind 2](#_Toc911717826)

[2.3. const变量 2](#_Toc43754355)

[2.3.1. clang::serialization::NUM\_PREDEF\_TYPE\_IDS 2](#_Toc1750370281)

[2.3.2. clang::serialization::NumSpecialTypeIDs 2](#_Toc1565899510)

[2.3.3. clang::serialization::NUM\_PREDEF\_DECL\_IDS 2](#_Toc1796456677)

[2.3.4. clang::serialization::DECL\_UPDATES 2](#_Toc1720906307)

[2.3.5. clang::serialization::LOCAL\_REDECLARATIONS 2](#_Toc758193950)

[3. 构建系统实用程序 2](#_Toc95436390)

[3.1. Typedefs 2](#_Toc1610037324)

[3.1.1. CXVirtualFileOverlay 2](#_Toc1079906607)

[3.1.2. CXModuleMapDescriptor 2](#_Toc947384454)

[3.2. 函数 3](#_Toc392872035)

[3.2.1. clang\_getBuildSessionTimestamp 3](#_Toc1881685086)

[3.2.2. clang\_VirtualFileOverlay\_create 3](#_Toc896076920)

[3.2.3. clang\_VirtualFileOverlay\_addFileMapping 3](#_Toc28914818)

[3.2.4. clang\_VirtualFileOverlay\_setCaseSensitivity 3](#_Toc1521357955)

[3.2.5. clang\_VirtualFileOverlay\_writeToBuffer 3](#_Toc1470576262)

[3.2.6. clang\_free 3](#_Toc202170324)

[3.2.7. clang\_VirtualFileOverlay\_dispose 3](#_Toc1209612226)

[3.2.8. clang\_ModuleMapDescriptor\_create 3](#_Toc759229388)

[3.2.9. clang\_ModuleMapDescriptor\_setFrameworkModuleName 3](#_Toc1471377188)

[3.2.10. clang\_ModuleMapDescriptor\_setUmbrellaHeader 4](#_Toc1194718879)

[3.2.11. clang\_ModuleMapDescriptor\_writeToBuffer 4](#_Toc1463446546)

[3.2.12. clang\_ModuleMapDescriptor\_dispose 4](#_Toc1236179317)

[4. 诊断报告 4](#_Toc1162646225)

[4.1. Typedefs 4](#_Toc1054359483)

[4.1.1. CXDiagnostic 4](#_Toc863302873)

[4.1.2. CXDiagnosticSet 4](#_Toc1208319985)

[4.2. 枚举 4](#_Toc1155678345)

[4.2.1. CXDiagnosticSeverity 4](#_Toc1573776050)

[4.2.2. CXLoadDiag\_Error 4](#_Toc2142091784)

[4.2.3. CXDiagnosticDisplayOptions 4](#_Toc2067396172)

[4.3. 函数 4](#_Toc1617530405)

[4.3.1. clang\_getNumDiagnosticsInSet 4](#_Toc1744978417)

[4.3.2. clang\_getDiagnosticInSet 5](#_Toc1485812034)

[4.3.3. clang\_loadDiagnostics 5](#_Toc1266503435)

[4.3.4. clang\_disposeDiagnosticSet 5](#_Toc1318401076)

[4.3.5. clang\_getChildDiagnostics 5](#_Toc96522336)

[4.3.6. clang\_disposeDiagnostic 5](#_Toc1361939825)

[4.3.7. clang\_formatDiagnostic 5](#_Toc780954752)

[4.3.8. clang\_defaultDiagnosticDisplayOptions 5](#_Toc1176428943)

[4.3.9. clang\_getDiagnosticSeverity 5](#_Toc161840631)

[4.3.10. clang\_getDiagnosticLocation 5](#_Toc1173826787)

[4.3.11. clang\_getDiagnosticSpelling 5](#_Toc910630382)

[4.3.12. clang\_getDiagnosticOption 6](#_Toc1057917551)

[4.3.13. clang\_getDiagnosticCategory 6](#_Toc1202741606)

[4.3.14. clang\_getDiagnosticCategoryName 6](#_Toc284504689)

[4.3.15. clang\_getDiagnosticCategoryText 6](#_Toc381010166)

[4.3.16. clang\_getDiagnosticNumRanges 6](#_Toc1404911930)

[4.3.17. clang\_getDiagnosticRange 6](#_Toc1494116915)

[4.3.18. clang\_getDiagnosticNumFixIts 6](#_Toc1140239554)

[4.3.19. clang\_getDiagnosticFixIt 6](#_Toc728805471)

[5. 文件操作例程 6](#_Toc541352146)

[5.1. Classes 6](#_Toc456202452)

[5.1.1. CXFileUniqueID 6](#_Toc1964984788)

[5.2. Typedefs 7](#_Toc1703998371)

[5.2.1. CXFile 7](#_Toc1510561935)

[5.3. 函数 7](#_Toc680804014)

[5.3.1. clang\_getFileName 7](#_Toc764834708)

[5.3.2. clang\_getFileTime 7](#_Toc518756633)

[5.3.3. clang\_getFileUniqueID 7](#_Toc107096416)

[5.3.4. clang\_File\_isEqual 7](#_Toc759442844)

[5.3.5. clang\_File\_tryGetRealPathName 7](#_Toc438669157)

[6. 物理source location 7](#_Toc1724626821)

[6.1. Classes 7](#_Toc356937613)

[6.1.1. CXSourceLocation 7](#_Toc1924481191)

[6.1.2. CXSourceRange 7](#_Toc843646608)

[6.1.3. CXSourceRangeList 7](#_Toc1675338689)

[6.2. 函数 8](#_Toc2021003528)

[6.2.1. clang\_getNullLocation 8](#_Toc58102785)

[6.2.2. clang\_equalLocations 8](#_Toc308809793)

[6.2.3. clang\_Location\_isInSystemHeader 8](#_Toc1049948823)

[6.2.4. clang\_Location\_isFromMainFile 8](#_Toc219943417)

[6.2.5. clang\_getNullRange 8](#_Toc1482636581)

[6.2.6. clang\_getRange 8](#_Toc1960579205)

[6.2.7. clang\_equalRanges 8](#_Toc1277860968)

[6.2.8. clang\_Range\_isNull 8](#_Toc537894539)

[6.2.9. clang\_getExpansionLocation 8](#_Toc97600246)

[6.2.10. clang\_getPresumedLocation 9](#_Toc1658871134)

[6.2.11. clang\_getInstantiationLocation 9](#_Toc1942806469)

[6.2.12. clang\_getSpellingLocation 9](#_Toc1591717161)

[6.2.13. clang\_getFileLocation 9](#_Toc651627040)

[6.2.14. clang\_getRangeStart 9](#_Toc524128292)

[6.2.15. clang\_getRangeEnd 9](#_Toc2133069307)

[6.2.16. clang\_disposeSourceRangeList 9](#_Toc1107829493)

[7. comment introspection(内省) 9](#_Toc341629433)

[7.1. Classes 9](#_Toc1689584030)

[7.1.1. CXComment 9](#_Toc470907780)

[7.2. Typedefs 9](#_Toc1022433447)

[7.2.1. CXAPISet 9](#_Toc306935091)

[7.3. 枚举类型 10](#_Toc989664413)

[7.3.1. CXCommentKind 10](#_Toc1129529863)

[7.3.2. CXCommentInlineCommandRenderKind 10](#_Toc1066377935)

[7.3.3. CXCommentParamPassDirection 10](#_Toc1428333570)

[7.4. 函数 10](#_Toc706673036)

[7.4.1. clang\_Cursor\_getParsedComment 10](#_Toc1423315549)

[7.4.2. clang\_Comment\_getKind 10](#_Toc1205331114)

[7.4.3. clang\_Comment\_getNumChildren 10](#_Toc1550319645)

[7.4.4. clang\_Comment\_getChild 10](#_Toc951170590)

[7.4.5. clang\_Comment\_isWhitespace 10](#_Toc1078850994)

[7.4.6. clang\_InlineContentComment\_hasTrailingNewline 10](#_Toc1608422430)

[7.4.7. clang\_TextComment\_getText 11](#_Toc1259980384)

[7.4.8. clang\_InlineCommandComment\_getCommandName 11](#_Toc2128799817)

[7.4.9. clang\_InlineCommandComment\_getRenderKind 11](#_Toc1828365847)

[7.4.10. clang\_InlineCommandComment\_getNumArgs 11](#_Toc595133317)

[7.4.11. clang\_InlineCommandComment\_getArgText 11](#_Toc1941895375)

[7.4.12. clang\_HTMLTagComment\_getTagName 11](#_Toc958743168)

[7.4.13. clang\_HTMLStartTagComment\_isSelfClosing 11](#_Toc1133027856)

[7.4.14. clang\_HTMLStartTag\_getNumAttrs 11](#_Toc2039495621)

[7.4.15. clang\_HTMLStartTag\_getAttrName 11](#_Toc470130654)

[7.4.16. clang\_HTMLStartTag\_getAttrValue 11](#_Toc928350677)

[7.4.17. clang\_BlockCommandComment\_getCommandName 11](#_Toc1483729135)

[7.4.18. clang\_BlockCommandComment\_getNumArgs 11](#_Toc1121757695)

[7.4.19. clang\_BlockCommandComment\_getArgText 11](#_Toc1452478970)

[7.4.20. clang\_BlockCommandComment\_getParagraph 12](#_Toc1469314794)

[7.4.21. clang\_ParamCommandComment\_getParamName 12](#_Toc82103540)

[7.4.22. clang\_ParamCommandComment\_isParamIndexValid 12](#_Toc1794108403)

[7.4.23. clang\_ParamCommandComment\_getParamIndex 12](#_Toc1011415177)

[7.4.24. clang\_ParamCommandComment\_isDirectionExplicit 12](#_Toc553011320)

[7.4.25. clang\_ParamCommandComment\_getDirection 12](#_Toc669058202)

[7.4.26. clang\_TParamCommandComment\_getParamName 12](#_Toc1318350268)

[7.4.27. clang\_TParamCommandComment\_isParamPositionValid 12](#_Toc1542675734)

[7.4.28. clang\_TParamCommandComment\_getDepth 12](#_Toc1798588065)

[7.4.29. clang\_TParamCommandComment\_getIndex 12](#_Toc237244555)

[7.4.30. clang\_VerbatimBlockLineComment\_getText 12](#_Toc823525656)

[7.4.31. clang\_VerbatimLineComment\_getText 12](#_Toc357777453)

[7.4.32. clang\_HTMLTagComment\_getAsString 13](#_Toc1660560104)

[7.4.33. clang\_FullComment\_getAsHTML 13](#_Toc2028856770)

[7.4.34. clang\_FullComment\_getAsXML 13](#_Toc1908097098)

[7.4.35. clang\_createAPISet 13](#_Toc464247047)

[7.4.36. clang\_disposeAPISet 13](#_Toc960224116)

[7.4.37. clang\_getSymbolGraphForUSR 13](#_Toc1369035881)

[7.4.38. clang\_getSymbolGraphForCursor 13](#_Toc1724227431)

[8. libclang：Clang的C接口 13](#_Toc941540286)

[8.1. 包含的模块 13](#_Toc1049918080)

[8.2. classes 14](#_Toc171877100)

[8.2.1. CXUnsavedFile 14](#_Toc735952013)

[8.2.2. CXVersion 14](#_Toc2008661248)

[8.2.3. CXCursor 14](#_Toc1304904956)

[8.2.4. CXIndex 14](#_Toc627963986)

[8.2.5. CXTargetInfo 14](#_Toc331308255)

[8.2.6. CXTranslationUnit 14](#_Toc85771985)

[8.2.7. CXClientData 15](#_Toc2111693121)

[8.2.8. CXVersion 15](#_Toc1453065950)

[8.3. 枚举 15](#_Toc1538250955)

[8.3.1. CXAvailabilityKind 15](#_Toc1433524268)

[8.3.2. CXCursor\_ExceptionSpecificationKind 15](#_Toc1535169490)

[8.3.3. CXGlobalOptFlags 15](#_Toc1184875710)

[8.3.4. CXCursorKind 15](#_Toc297455797)

[8.4. 函数 15](#_Toc2088180810)

[8.4.1. clang\_createIndex 15](#_Toc1853933912)

[8.4.2. clang\_disposeIndex 15](#_Toc1615806065)

[8.4.3. clang\_CXIndex\_setGlobalOptions 15](#_Toc1483372896)

[8.4.4. clang\_CXIndex\_getGlobalOptions 15](#_Toc1505038329)

[8.4.5. clang\_CXIndex\_setInvocationEmissionPathOption 16](#_Toc1853050620)

[8.4.6. clang\_isFileMultipleIncludeGuarded 16](#_Toc159414905)

[8.4.7. clang\_getFile 16](#_Toc1862815783)

[8.4.8. clang\_getFileContents 16](#_Toc1366127077)

[8.4.9. clang\_getLocation 16](#_Toc40788027)

[8.4.10. clang\_getLocationForOffset 16](#_Toc1623429233)

[8.4.11. clang\_getSkippedRanges 16](#_Toc1830374124)

[8.4.12. clang\_getAllSkippedRanges 16](#_Toc1001012144)

[8.4.13. clang\_getNumDiagnostics 16](#_Toc844981466)

[8.4.14. clang\_getDiagnostic 17](#_Toc1407117907)

[8.4.15. clang\_getDiagnosticSetFromTU 17](#_Toc1942552430)

[9. 附录 17](#_Toc1894899547)

# 概述

该文档的位置是tools/clang/docs/doxygen/html/index.html

# AST文件AST常量

该组中的常数描述了AST文件中抽象语法树的各个组件。

## classes

### struct clang::serialization::LocalRedeclarationsInfo

描述一个声明的重新声明。

### struct clang::serialization::ObjCCategoriesInfo

描述一个object-c类的分类。

### class clang::serialization::DeclarationNameKey

通过[DeclarationName](file://home\\songqing\\docllvm\\tmp\\docs\\tools\\clang\\docs\\doxygen\\html\\classclang_1_1DeclarationName.html)在查找一个条目时，使用的key。

## 枚举类型

### enum clang::serialization::PredefinedTypeIDs

预定义的类型ID.

### enum clang::serialization::TypeCode

记录每种类型的代码。

### enum clang::serialization::SpecialTypeIDs

通过语义分析构建的特殊类型的类型ID。

### enum clang::serialization::PredefinedDeclIDs

预定义的声明ID。

### enum clang::serialization::DeclCode

为每种类型的声明记录代码。

### enum clang::serialization::StmtCode

记录每种语句或表达式的代码。

### enum clang::serialization::DesignatorTypes

在一个DesignatedInitExpr中出现的designators的类型。

### enum clang::serialization::CtorInitializerType

在一个CtorInitializer中出现的不同数据类型。

### enum clang::ser**i**alization::CleanupObjectKind

ExprwithCleanups拥有的各种清理对象。

## const变量

### clang::serialization::NUM\_PREDEF\_TYPE\_IDS

const unsigned clang::serialization::NUM\_PREDEF\_TYPE\_IDS = 300

为predef\_type\_\*常数保留的预定义类型ID的数量。

### clang::serialization::NumSpecialTypeIDs

const unsigned clang::serialization::NumSpecialTypeIDs = 8

特定类型ID的数量。

### clang::serialization::NUM\_PREDEF\_DECL\_IDS

const unsigned int clang::serialization::NUM\_PREDEF\_DECL\_IDS = 18

预定义的声明IDs的数量

### clang::serialization::DECL\_UPDATES

const unsigned int clang::serialization::DECL\_UPDATES = 49

声明的更新记录，该声明是被deserialized后修改的。

### clang::serialization::LOCAL\_REDECLARATIONS

const unsigned int clang::serialization::LOCAL\_REDECLARATIONS = 50

记录一个声明的本地重新声明的列表的代码。

# 构建系统实用程序

## Typedefs

### CXVirtualFileOverlay

typedef struct CXVirtualFileOverlayImpl \* CXVirtualFileOverlay

对象封装有关将虚拟文件/目录叠加在真实文件系统上的信息。

### CXModuleMapDescriptor

typedef struct CXModuleMapDescriptorImpl \* CXModuleMapDescriptor

对象封装一个module.map文件的信息。

## 函数

### clang\_getBuildSessionTimestamp

CINDEX\_LINKAGE unsigned long long clang\_getBuildSessionTimestamp (void)

将时间戳返回，以与Clang 's -fbuild-Session-timestamp =选项一起使用。

### clang\_VirtualFileOverlay\_create

CINDEX\_LINKAGE CXVirtualFileOverlay clang\_VirtualFileOverlay\_create (unsigned options)

创建一个CXVirtualFileOverlay对象。

### clang\_VirtualFileOverlay\_addFileMapping

CINDEX\_LINKAGE enum CXErrorCode clang\_VirtualFileOverlay\_addFileMapping (CXVirtualFileOverlay, const char \*virtualPath, const char \*realPath)

将绝对虚拟文件路径映射到绝对真实的路径。

### clang\_VirtualFileOverlay\_setCaseSensitivity

CINDEX\_LINKAGE enum CXErrorCode clang\_VirtualFileOverlay\_setCaseSensitivity (CXVirtualFileOverlay, int caseSensitive)

为CXVIrtualFileOverlay对象设置case Sensitivity。

### clang\_VirtualFileOverlay\_writeToBuffer

CINDEX\_LINKAGE enum CXErrorCode clang\_VirtualFileOverlay\_writeToBuffer (CXVirtualFileOverlay, unsigned options, char \*\*out\_buffer\_ptr, unsigned \*out\_buffer\_size)

将cxvirtualfileoverlay对象写入char缓冲区。

### clang\_free

CINDEX\_LINKAGE void clang\_free (void \*buffer)

释放由libclang分配的内存，例如，CXVIRTUALFILEOVERLAY（）或Clang\_ModulemapDescriptor\_writeTobuffer（）返回的buffer。

### clang\_VirtualFileOverlay\_dispose

CINDEX\_LINKAGE void clang\_VirtualFileOverlay\_dispose (CXVirtualFileOverlay)

处理一个cxvirtualfileoverlay对象。

### clang\_ModuleMapDescriptor\_create

CINDEX\_LINKAGE CXModuleMapDescriptor clang\_ModuleMapDescriptor\_create (unsigned options)

创建CXModuleMapDescriptor对象。

### clang\_ModuleMapDescriptor\_setFrameworkModuleName

CINDEX\_LINKAGE enum CXErrorCode clang\_ModuleMapDescriptor\_setFrameworkModuleName (CXModuleMapDescriptor, const char \*name)

设置module.map描述的框架模块的名称。

### clang\_ModuleMapDescriptor\_setUmbrellaHeader

CINDEX\_LINKAGE enum CXErrorCode clang\_ModuleMapDescriptor\_setUmbrellaHeader (CXModuleMapDescriptor, const char \*name)

设置module.map描述的 umbrella header 名称。

### clang\_ModuleMapDescriptor\_writeToBuffer

CINDEX\_LINKAGE enum CXErrorCode clang\_ModuleMapDescriptor\_writeToBuffer (CXModuleMapDescriptor, unsigned options, char \*\*out\_buffer\_ptr, unsigned \*out\_buffer\_size)

将CXModulemapDescriptor对象写入char缓冲区。

### clang\_ModuleMapDescriptor\_dispose

CINDEX\_LINKAGE void clang\_ModuleMapDescriptor\_dispose (CXModuleMapDescriptor)

处理CXModulemapDescriptor对象

# 诊断报告

## Typedefs

### CXDiagnostic

typedef void \* CXDiagnostic

一个诊断，包含诊断的严重性，位置，文本，源范围和修复 - 提示。

### CXDiagnosticSet

typedef void \* CXDiagnosticSet

一组诊断。

## 枚举

### CXDiagnosticSeverity

描述特定诊断的严重等级。

### CXLoadDiag\_Error

描述调用clang\_loadDiagnostics的错误的类型。

### CXDiagnosticDisplayOptions

控制诊断显示的选项。

## 函数

### clang\_getNumDiagnosticsInSet

CINDEX\_LINKAGE unsigned clang\_getNumDiagnosticsInSet (CXDiagnosticSet Diags)

确定一个CXDiagnosticSet中的诊断的数量。

### clang\_getDiagnosticInSet

CINDEX\_LINKAGE CXDiagnostic clang\_getDiagnosticInSet (CXDiagnosticSet Diags, unsigned Index)

获取给定CXDiagnosticSet相关联的诊断。

### clang\_loadDiagnostics

CINDEX\_LINKAGE CXDiagnosticSet clang\_loadDiagnostics (const char \*file, enum CXLoadDiag\_Error \*error, CXString \*errorString)

从Clang诊断bitcode 文件中deserialize 一组诊断。

### clang\_disposeDiagnosticSet

CINDEX\_LINKAGE void clang\_disposeDiagnosticSet (CXDiagnosticSet Diags)

释放一个CXDiagnosticSet 和它包含的所有的诊断。

### clang\_getChildDiagnostics

CINDEX\_LINKAGE CXDiagnosticSet clang\_getChildDiagnostics (CXDiagnostic D)

获取一个CXDiagnostic的子诊断。

### clang\_disposeDiagnostic

CINDEX\_LINKAGE void clang\_disposeDiagnostic (CXDiagnostic Diagnostic)

Destroy a diagnostic. More...

销毁一个诊断。

### clang\_formatDiagnostic

CINDEX\_LINKAGE CXString clang\_formatDiagnostic (CXDiagnostic Diagnostic, unsigned Options)

以适合显示的特定格式来对给定的诊断格式化。

### clang\_defaultDiagnosticDisplayOptions

CINDEX\_LINKAGE unsigned clang\_defaultDiagnosticDisplayOptions (void)

检索与Clang编译器的默认行为最相似的显示选项集。

### clang\_getDiagnosticSeverity

CINDEX\_LINKAGE enum CXDiagnosticSeverity clang\_getDiagnosticSeverity (CXDiagnostic)

Determine the severity of the given diagnostic. More...

确定给定诊断的严重等级。

### clang\_getDiagnosticLocation

CINDEX\_LINKAGE CXSourceLocation clang\_getDiagnosticLocation (CXDiagnostic)

Retrieve the source location of the given diagnostic. More...

检索给定诊断的源位置。

### clang\_getDiagnosticSpelling

CINDEX\_LINKAGE CXString clang\_getDiagnosticSpelling (CXDiagnostic)

Retrieve the text of the given diagnostic. More...

检索给定诊断的txt。

### clang\_getDiagnosticOption

CINDEX\_LINKAGE CXString clang\_getDiagnosticOption (CXDiagnostic Diag, CXString \*Disable)

检索启用此诊断的命令行选项的名称。

### clang\_getDiagnosticCategory

CINDEX\_LINKAGE unsigned clang\_getDiagnosticCategory (CXDiagnostic)

检索当前诊断的类别编号。

### clang\_getDiagnosticCategoryName

CINDEX\_DEPRECATED CINDEX\_LINKAGE CXString clang\_getDiagnosticCategoryName (unsigned Category)

检索特定诊断类别的名称。

### clang\_getDiagnosticCategoryText

CINDEX\_LINKAGE CXString clang\_getDiagnosticCategoryText (CXDiagnostic)

为给定的诊断获取诊断类别txt。

### clang\_getDiagnosticNumRanges

CINDEX\_LINKAGE unsigned clang\_getDiagnosticNumRanges (CXDiagnostic)

Determine the number of source ranges associated with the given diagnostic. More...

确定给定诊断的source 范围的数量

### clang\_getDiagnosticRange

CINDEX\_LINKAGE CXSourceRange clang\_getDiagnosticRange (CXDiagnostic Diagnostic, unsigned Range)

获取给定诊断的source 范围的数量

### clang\_getDiagnosticNumFixIts

CINDEX\_LINKAGE unsigned clang\_getDiagnosticNumFixIts (CXDiagnostic Diagnostic)

Determine the number of fix-it hints associated with the given diagnostic. More...

确定给定诊断的fix-it hints的数量。

### clang\_getDiagnosticFixIt

CINDEX\_LINKAGE CXString clang\_getDiagnosticFixIt (CXDiagnostic Diagnostic, unsigned FixIt, CXSourceRange \*ReplacementRange)

获取给定的fix-it的替换信息。

# 文件操作例程

## Classes

### CXFileUniqueID

struct CXFileUniqueID

唯一地标识了一个CXFile，该CXFile在一个索引会话中指的是相同的基础文件。

## Typedefs

### CXFile

typedef void \* CXFile

是翻译单元一部分的特定源文件。

## 函数

### clang\_getFileName

CINDEX\_LINKAGE CXString clang\_getFileName (CXFile SFile)

检索给定文件的完整文件和路径名。

### clang\_getFileTime

CINDEX\_LINKAGE time\_t clang\_getFileTime (CXFile SFile)

检索给定文件的最后一个修改时间。

### clang\_getFileUniqueID

CINDEX\_LINKAGE int clang\_getFileUniqueID (CXFile file, CXFileUniqueID \*outID)

检索给定文件的唯一ID。

### clang\_File\_isEqual

CINDEX\_LINKAGE int clang\_File\_isEqual (CXFile file1, CXFile file2)

如果File1和File2指向同一文件或者它们都是无效的，则返回非零。

### clang\_File\_tryGetRealPathName

CINDEX\_LINKAGE CXString clang\_File\_tryGetRealPathName (CXFile file)

返回文件的真实路径名。

# 物理source location

clang在其抽象语法树中以物理source location进行详细介绍，并在源代码中使用的大多数tokens提供了文件，行和列信息。这些数据类型和函数用于表示程序中的特定点或程序中的一系列点的source location信息，并从这些数据类型中提取特定的位置信息。

## Classes

### CXSourceLocation

struct CXSourceLocation

标识翻译单元中的特定源位置。

### CXSourceRange

struct CXSourceRange

标识源代码中的半开字符范围。

### CXSourceRangeList

struct CXSourceRangeList

标识一系列范围。

## 函数

### clang\_getNullLocation

CINDEX\_LINKAGE CXSourceLocation clang\_getNullLocation (void)

检索零（无效）源位置。

### clang\_equalLocations

CINDEX\_LINKAGE unsigned clang\_equalLocations (CXSourceLocation loc1, CXSourceLocation loc2)

确定指向同一翻译单元的两个源位置，是否必须指向源代码中的完全相同点。

### clang\_Location\_isInSystemHeader

CINDEX\_LINKAGE int clang\_Location\_isInSystemHeader (CXSourceLocation location)

如果给定的源位置位于系统头文件中，则返回非零。

### clang\_Location\_isFromMainFile

CINDEX\_LINKAGE int clang\_Location\_isFromMainFile (CXSourceLocation location)

如果给定的source location位于相应翻译单元的主文件中，则返回非零。

### clang\_getNullRange

CINDEX\_LINKAGE CXSourceRange clang\_getNullRange (void)

检索一个空（无效）源范围。

### clang\_getRange

CINDEX\_LINKAGE CXSourceRange clang\_getRange (CXSourceLocation begin, CXSourceLocation end)

Retrieve a source range given the beginning and ending source locations. More...

在给定的source location的开始和结束，检索源范围。

### clang\_equalRanges

CINDEX\_LINKAGE unsigned clang\_equalRanges (CXSourceRange range1, CXSourceRange range2)

确定两个范围是否等效。

### clang\_Range\_isNull

CINDEX\_LINKAGE int clang\_Range\_isNull (CXSourceRange range)

如果范围为null，则返回非零。

### clang\_getExpansionLocation

CINDEX\_LINKAGE void clang\_getExpansionLocation (CXSourceLocation location, CXFile \*file, unsigned \*line, unsigned \*column, unsigned \*offset)

检索以给定源位置表示的文件，行，列和偏移。

### clang\_getPresumedLocation

CINDEX\_LINKAGE void clang\_getPresumedLocation (CXSourceLocation location, CXString \*filename, unsigned \*line, unsigned \*column)

通过#line 指定来检索给定源位置表示的文件，行，列和偏移。

### clang\_getInstantiationLocation

CINDEX\_LINKAGE void clang\_getInstantiationLocation (CXSourceLocation location, CXFile \*file, unsigned \*line, unsigned \*column, unsigned \*offset)

传统API检索以给定源位置表示的文件，行，列和偏移。

### clang\_getSpellingLocation

CINDEX\_LINKAGE void clang\_getSpellingLocation (CXSourceLocation location, CXFile \*file, unsigned \*line, unsigned \*column, unsigned \*offset)

检索以给定源位置表示的文件，行，列和偏移。

### clang\_getFileLocation

CINDEX\_LINKAGE void clang\_getFileLocation (CXSourceLocation location, CXFile \*file, unsigned \*line, unsigned \*column, unsigned \*offset)

检索以给定源位置表示的文件，行，列和偏移。

### clang\_getRangeStart

CINDEX\_LINKAGE CXSourceLocation clang\_getRangeStart (CXSourceRange range)

检索代表源范围内第一个字符的源位置。

### clang\_getRangeEnd

CINDEX\_LINKAGE CXSourceLocation clang\_getRangeEnd (CXSourceRange range)

检索代表源范围内最后一个字符的源位置。

### clang\_disposeSourceRangeList

CINDEX\_LINKAGE void clang\_disposeSourceRangeList (CXSourceRangeList \*ranges)

销毁给定的cxsourcerangelist。

# comment introspection(内省)

该组中的例程在文档注释中提供了对信息的访问。这些设施与核心不同，可能会遵守自己的 schedule of stability and deprecation。

## Classes

### CXComment

struct CXComment

一个解析过的comment。

## Typedefs

### CXAPISet

typedef struct CXAPISetImpl \* CXAPISet

CXAPISET是一种不透明的类型，它代表包含给定翻译单元的所有API信息的数据结构。

## 枚举类型

### CXCommentKind

描述注释AST节点的类型（CXCOMMENT）。

### CXCommentInlineCommandRenderKind

The most appropriate rendering mode for an inline command, chosen on command semantics in Doxygen. More...

内联命令的最合适的渲染模式，是在doxygen中的命令语义上选择的。

### CXCommentParamPassDirection

描述\ param或\ arg命令的参数传递方向。

## 函数

### clang\_Cursor\_getParsedComment

CINDEX\_LINKAGE CXComment clang\_Cursor\_getParsedComment (CXCursor C)

Given a cursor that represents a documentable entity (e.g., declaration), return the associated parsed comment as a CXComment\_FullComment AST node. More...

给定一个代表可文档实体的光标（例如声明），将关联的解析注释作为一个 CXComment\_FullComment AST 节点返回。

### clang\_Comment\_getKind

CINDEX\_LINKAGE enum CXCommentKind clang\_Comment\_getKind (CXComment Comment)

### clang\_Comment\_getNumChildren

CINDEX\_LINKAGE unsigned clang\_Comment\_getNumChildren (CXComment Comment)

### clang\_Comment\_getChild

CINDEX\_LINKAGE CXComment clang\_Comment\_getChild (CXComment Comment, unsigned ChildIdx)

### clang\_Comment\_isWhitespace

CINDEX\_LINKAGE unsigned clang\_Comment\_isWhitespace (CXComment Comment)

A CXComment\_Paragraph node is considered whitespace if it contains only CXComment\_Text nodes that are empty or whitespace. More...

如果 CXComment\_Paragraph 节点仅包含CXComment\_Text节点， CXComment\_Text节点是空白或空格，则将其视为whitespace。

### clang\_InlineContentComment\_hasTrailingNewline

CINDEX\_LINKAGE unsigned clang\_InlineContentComment\_hasTrailingNewline (CXComment Comment)

### clang\_TextComment\_getText

CINDEX\_LINKAGE CXString clang\_TextComment\_getText (CXComment Comment)

### clang\_InlineCommandComment\_getCommandName

CINDEX\_LINKAGE CXString clang\_InlineCommandComment\_getCommandName (CXComment Comment)

### clang\_InlineCommandComment\_getRenderKind

CINDEX\_LINKAGE enum CXCommentInlineCommandRenderKind clang\_InlineCommandComment\_getRenderKind (CXComment Comment)

### clang\_InlineCommandComment\_getNumArgs

CINDEX\_LINKAGE unsigned clang\_InlineCommandComment\_getNumArgs (CXComment Comment)

### clang\_InlineCommandComment\_getArgText

CINDEX\_LINKAGE CXString clang\_InlineCommandComment\_getArgText (CXComment Comment, unsigned ArgIdx)

### clang\_HTMLTagComment\_getTagName

CINDEX\_LINKAGE CXString clang\_HTMLTagComment\_getTagName (CXComment Comment)

### clang\_HTMLStartTagComment\_isSelfClosing

CINDEX\_LINKAGE unsigned clang\_HTMLStartTagComment\_isSelfClosing (CXComment Comment)

### clang\_HTMLStartTag\_getNumAttrs

CINDEX\_LINKAGE unsigned clang\_HTMLStartTag\_getNumAttrs (CXComment Comment)

### clang\_HTMLStartTag\_getAttrName

CINDEX\_LINKAGE CXString clang\_HTMLStartTag\_getAttrName (CXComment Comment, unsigned AttrIdx)

### clang\_HTMLStartTag\_getAttrValue

CINDEX\_LINKAGE CXString clang\_HTMLStartTag\_getAttrValue (CXComment Comment, unsigned AttrIdx)

### clang\_BlockCommandComment\_getCommandName

CINDEX\_LINKAGE CXString clang\_BlockCommandComment\_getCommandName (CXComment Comment)

### clang\_BlockCommandComment\_getNumArgs

CINDEX\_LINKAGE unsigned clang\_BlockCommandComment\_getNumArgs (CXComment Comment)

### clang\_BlockCommandComment\_getArgText

CINDEX\_LINKAGE CXString clang\_BlockCommandComment\_getArgText (CXComment Comment, unsigned ArgIdx)

### clang\_BlockCommandComment\_getParagraph

CINDEX\_LINKAGE CXComment clang\_BlockCommandComment\_getParagraph (CXComment Comment)

### clang\_ParamCommandComment\_getParamName

CINDEX\_LINKAGE CXString clang\_ParamCommandComment\_getParamName (CXComment Comment)

### clang\_ParamCommandComment\_isParamIndexValid

CINDEX\_LINKAGE unsigned clang\_ParamCommandComment\_isParamIndexValid (CXComment Comment)

### clang\_ParamCommandComment\_getParamIndex

CINDEX\_LINKAGE unsigned clang\_ParamCommandComment\_getParamIndex (CXComment Comment)

### clang\_ParamCommandComment\_isDirectionExplicit

CINDEX\_LINKAGE unsigned clang\_ParamCommandComment\_isDirectionExplicit (CXComment Comment)

### clang\_ParamCommandComment\_getDirection

CINDEX\_LINKAGE enum CXCommentParamPassDirection clang\_ParamCommandComment\_getDirection (CXComment Comment)

### clang\_TParamCommandComment\_getParamName

CINDEX\_LINKAGE CXString clang\_TParamCommandComment\_getParamName (CXComment Comment)

### clang\_TParamCommandComment\_isParamPositionValid

CINDEX\_LINKAGE unsigned clang\_TParamCommandComment\_isParamPositionValid (CXComment Comment)

### clang\_TParamCommandComment\_getDepth

CINDEX\_LINKAGE unsigned clang\_TParamCommandComment\_getDepth (CXComment Comment)

### clang\_TParamCommandComment\_getIndex

CINDEX\_LINKAGE unsigned clang\_TParamCommandComment\_getIndex (CXComment Comment, unsigned Depth)

### clang\_VerbatimBlockLineComment\_getText

CINDEX\_LINKAGE CXString clang\_VerbatimBlockLineComment\_getText (CXComment Comment)

### clang\_VerbatimLineComment\_getText

CINDEX\_LINKAGE CXString clang\_VerbatimLineComment\_getText (CXComment Comment)

### clang\_HTMLTagComment\_getAsString

CINDEX\_LINKAGE CXString clang\_HTMLTagComment\_getAsString (CXComment Comment)

将HTML标签AST节点转换为字符串。

### clang\_FullComment\_getAs**H**TML

CINDEX\_LINKAGE CXString clang\_FullComment\_getAsHTML (CXComment Comment)

将给定的完整分析后的comment转换为HTML片段。

### clang\_FullComment\_getAsXML

CINDEX\_LINKAGE CXString clang\_FullComment\_getAsXML (CXComment Comment)

Convert a given full parsed comment to an XML document. More...

将给定的完整解析后的comment转换为XML文档。

### clang\_createAPISet

CINDEX\_LINKAGE enum CXErrorCode clang\_createAPISet (CXTranslationUnit tu, CXAPISet \*out\_api)

遍历翻译单元以创建CXAPISET。

### clang\_disposeAPISet

CINDEX\_LINKAGE void clang\_disposeAPISet (CXAPISet api)

销毁一个APISet。

### clang\_getSymbolGraphForUSR

CINDEX\_LINKAGE CXString clang\_getSymbolGraphForUSR (const char \*usr, CXAPISet api)

为给定的USR生成单个符号图。

### clang\_getSymbolGraphForCursor

CINDEX\_LINKAGE CXString clang\_getSymbolGraphForCursor (CXCursor cursor)

在给定的光标处为声明生成单个符号图。

# libclang：Clang的C接口

Clang的C接口提供了一个相对较小的API，该API将解析源代码的设施暴露于抽象的语法树（AST），加载已经解析的AST，遍历AST，将AST中的元素与物理source location相关联，以及其他支持基于Clang的开发工具。

## 包含的模块

* [CompilationDatabase](file://home\\songqing\\docllvm\\tmp\\docs\\tools\\clang\\docs\\doxygen\\html\\group__COMPILATIONDB.html) functions
* String manipulation routines
* Translation unit manipulation。该组中的例程通过解析文件内容或通过读取翻译单元的序列化表示，来创建和销毁文件中的翻译单元。
* Cursor manipulations
* Mapping between cursors and source code。光标代表抽象语法树（AST）中的位置。这些例程有助于在光标和源代码中描述的实体出现的物理位置之间映射。映射均以两个方向提供，因此可以从源代码到AST的映射和反向映射。
* Type information for CXCursors
* Information for attributes
* Traversing the AST with cursors。 这些例程提供了使用光标遍历抽象语法树的能力。
* Cross-referencing in the AST。 这些例程提供了通过提供光标指向的条目(实体)的名称来确定翻译单元内部和跨越翻译单元的指向的能力。
* Name Mangling API Functions
* Module introspection。该组中的函数提供有关模块的信息的访问。
* C++ AST introspection。组中的例程提供了特定于C ++语言功能的ASTs中的信息的访问。
* Token extraction and manipulation。该组中的例程提供了对翻译单元中tokens的访问，以及将这些tokens的语义映射到相应的光标。
* Debugging facilities。这些例程仅用于测试和调试，不应依靠。
* Code completion。代码完备性涉及获取（不完整的）源文件，以及对用户主动编辑该文件的知识，以及用户可能希望在源代码中的特定点上使用的语法-合法和语义-合法的构造。这些数据结构和例程为代码完备性提供了支持。
* Miscellaneous utility functions
* Remapping functions
* Higher level API functions

这些模块后面会单独介绍。

## classes

### CXUnsavedFile

struct CXUnsavedFile

提供一个文件的内容，该文件还没有被保存到disk中。

### CXVersion

struct CXVersion

以major.minor.subminor形式来描述版本号。

### CXCursor

struct CXCursor

A cursor representing some element in the abstract syntax tree for a translation unit. More...

用于翻译单元的抽象语法树中代表某些元素的光标。

### CXIndex

typedef void \* CXIndex

一个由一组翻译单元组成的“索引”，通常将其链接到可执行文件或库中。

### CXTargetInfo

typedef struct CXTargetInfoImpl \* CXTargetInfo

不透明类型，代表给定翻译单元的目标信息。

### CXTranslationUnit

typedef struct CXTranslationUnitImpl \* CXTranslationUnit

单个翻译单元，该单元位于索引中。

### CXClientData

typedef void \* CXClientData

不透明的指针代表客户数据，将传递给各种回调和访问者。

### CXVersion

typedef struct CXVersion CXVersion

Describes a version number of the form major.minor.subminor. More...

以minor.subminor 格式描述一个版本编号。

## 枚举

### CXAvailabilityKind

enum CXAvailabilityKind { CXAvailability\_Available, CXAvailability\_Deprecated, CXAvailability\_NotAvailable, CXAvailability\_NotAccessible }

描述了特定实体的可用性，这表明该实体的使用是否会导致警告或错误，因为它被降低或不可用。

### CXCursor\_ExceptionSpecificationKind

描述光标的异常规范。

### CXGlobalOptFlags

### CXCursorKind

描述了光标所指的实体。

## 函数

### clang\_createIndex

CINDEX\_LINKAGE CXIndex clang\_createIndex (int excludeDeclarationsFromPCH, int displayDiagnostics)

提供了创建翻译单元的共享上下文。

### clang\_disposeIndex

CINDEX\_LINKAGE void clang\_disposeIndex (CXIndex index)

销毁给定的索引。

### clang\_CXIndex\_setGlobalOptions

CINDEX\_LINKAGE void clang\_CXIndex\_setGlobalOptions (CXIndex, unsigned options)

Sets general options associated with a CXIndex. More...

设置与CXINDEX关联的通用选项。

### clang\_CXIndex\_getGlobalOptions

CINDEX\_LINKAGE unsigned clang\_CXIndex\_getGlobalOptions (CXIndex)

获取与CXINDEX关联的通用选项。

### clang\_CXIndex\_setInvocationEmissionPathOption

CINDEX\_LINKAGE void clang\_CXIndex\_setInvocationEmissionPathOption (CXIndex, const char \*Path)

在一个CXIndex中设置调用发射路径选项。

### clang\_isFileMultipleIncludeGuarded

CINDEX\_LINKAGE unsigned clang\_isFileMultipleIncludeGuarded (CXTranslationUnit tu, CXFile file)

Determine whether the given header is guarded against multiple inclusions, either with the conventional #ifndef/#define/#endif macro guards or with #pragma once. More...

确定给定的header是否可以防止多个inclusions，要么使用常规的＃ifndef/＃define/＃endif宏来保护，要么使用#pragm once。

### clang\_getFile

CINDEX\_LINKAGE CXFile clang\_getFile (CXTranslationUnit tu, const char \*file\_name)

Retrieve a file handle within the given translation unit. More...

在给定翻译单元中检索文件句柄。

### clang\_getFileContents

const CINDEX\_LINKAGE char \* clang\_getFileContents (CXTranslationUnit tu, CXFile file, size\_t \*size)

检索与给定文件关联的缓冲区。

### clang\_getLocation

CINDEX\_LINKAGE CXSourceLocation clang\_getLocation (CXTranslationUnit tu, CXFile file, unsigned line, unsigned column)

检索与特定翻译单元中给定 file/line/column 相关的源位置。

### clang\_getLocationForOffset

CINDEX\_LINKAGE CXSourceLocation clang\_getLocationForOffset (CXTranslationUnit tu, CXFile file, unsigned offset)

检索与特定翻译单元中给定字符偏移相关的源位置。

### clang\_getSkippedRanges

CINDEX\_LINKAGE CXSourceRangeList \* clang\_getSkippedRanges (CXTranslationUnit tu, CXFile file)

Retrieve all ranges that were skipped by the preprocessor. More...

检索前处理器(preprocessor)跳过的所有范围。

### clang\_getAllSkippedRanges

CINDEX\_LINKAGE CXSourceRangeList \* clang\_getAllSkippedRanges (CXTranslationUnit tu)

检索所有范围，从预处理程序跳过的所有文件。

### clang\_getNumDiagnostics

CINDEX\_LINKAGE unsigned clang\_getNumDiagnostics (CXTranslationUnit Unit)

确定给定翻译单元产生的诊断数量。

### clang\_getDiagnostic

CINDEX\_LINKAGE CXDiagnostic clang\_getDiagnostic (CXTranslationUnit Unit, unsigned Index)

检索与给定翻译单元相关的诊断。

### clang\_getDiagnosticSetFromTU

CINDEX\_LINKAGE CXDiagnosticSet clang\_getDiagnosticSetFromTU (CXTranslationUnit Unit)

检索与翻译单元相关的完整诊断集。

# libclang模块:CompilationDatabase 函数

CompilationDatabase 函数的协作图。

## Typedefs

### CXCompilationDatabase

typedef void \* CXCompilationDatabase

编译数据库持有用于编译项目中文件的所有信息。

### CXCompileCommands

typedef void \* CXCompileCommands

包含编译数据库中搜索的结果。

### CXCompileCommand

typedef void \* CXCompileCommand

表示命令行调用以编译特定文件。

## 枚举

### CXCompilationDatabase\_Error

enum CXCompilationDatabase\_Error { CXCompilationDatabase\_NoError = 0, CXCompilationDatabase\_CanNotLoadDatabase = 1 }

编译数据库的错误代码。

## 函数

### clang\_CompilationDatabase\_fromDirectory

CINDEX\_LINKAGE CXCompilationDatabase clang\_CompilationDatabase\_fromDirectory (const char \*BuildDir, CXCompilationDatabase\_Error \*ErrorCode)

从目录BuildDir中的数据库中创建一个编译数据库。

### clang\_CompilationDatabase\_dispose

CINDEX\_LINKAGE void clang\_CompilationDatabase\_dispose (CXCompilationDatabase)

释放给定的编译数据库。

### clang\_CompilationDatabase\_getCompileCommands

CINDEX\_LINKAGE CXCompileCommands clang\_CompilationDatabase\_getCompileCommands (CXCompilationDatabase, const char \*CompleteFileName)

查找用于文件的编译命令。

### clang\_CompilationDatabase\_getAllCompileCommands

CINDEX\_LINKAGE CXCompileCommands clang\_CompilationDatabase\_getAllCompileCommands (CXCompilationDatabase)

在给定的编译数据库中获取所有编译命令。

### clang\_CompileCommands\_dispose

CINDEX\_LINKAGE void clang\_CompileCommands\_dispose (CXCompileCommands)

释放给定的编译。

### clang\_CompileCommands\_getSize

CINDEX\_LINKAGE unsigned clang\_CompileCommands\_getSize (CXCompileCommands)

Get the number of CompileCommand we have for a file. More...

获取文件的CompileCommand 的数量。

### clang\_CompileCommands\_getCommand

CINDEX\_LINKAGE CXCompileCommand clang\_CompileCommands\_getCommand (CXCompileCommands, unsigned I)

获取一个文件的第I个CompileCommand 。

### clang\_CompileCommand\_getDirectory

CINDEX\_LINKAGE CXString clang\_CompileCommand\_getDirectory (CXCompileCommand)

Get the working directory where the CompileCommand was executed from. More...

获取CompileCommand 执行的工作目录。

### clang\_CompileCommand\_getFilename

CINDEX\_LINKAGE CXString clang\_CompileCommand\_getFilename (CXCompileCommand)

获取与CompileCommand相关联的文件名。

### clang\_CompileCommand\_getNumArgs

CINDEX\_LINKAGE unsigned clang\_CompileCommand\_getNumArgs (CXCompileCommand)

获取编译器调用中的参数数量。

### clang\_CompileCommand\_getArg

CINDEX\_LINKAGE CXString clang\_CompileCommand\_getArg (CXCompileCommand, unsigned I)

在编译器调用中获取第 I 个参数值。

### clang\_CompileCommand\_getNumMappedSources

CINDEX\_LINKAGE unsigned clang\_CompileCommand\_getNumMappedSources (CXCompileCommand)

获取编译器调用的源映射数量。

### clang\_CompileCommand\_getMappedSourcePath

CINDEX\_LINKAGE CXString clang\_CompileCommand\_getMappedSourcePath (CXCompileCommand, unsigned I)

获取编译器调用的第I个映射源路径。

### clang\_CompileCommand\_getMappedSourceContent

CINDEX\_LINKAGE CXString clang\_CompileCommand\_getMappedSourceContent (CXCompileCommand, unsigned I)

Get the I'th mapped source content for the compiler invocation. More...

获取编译器调用的第I个映射源内容。

# libclang模块:string操作例程

string操作例程的协作图。

## Classes

### CXString

struct CXString

字符串。

### CXStringSet

struct CXStringSet

## 函数

### clang\_getCString

const CINDEX\_LINKAGE char \* clang\_getCString (CXString string)

获取给定字符串的字符数据。

### clang\_disposeString

CINDEX\_LINKAGE void clang\_disposeString (CXString string)

释放给定的字符串。

### clang\_disposeStringSet

CINDEX\_LINKAGE void clang\_disposeStringSet (CXStringSet \*set)

释放给定的字符串集合。

# libclang模块:翻译单元操作

翻译单元的协作图。

该组中的例程通过解析文件内容或通过读取翻译单元的序列化表示，来创建和销毁文件中的翻译单元。

## Classes

### CXTUResourceUsageEntry

struct CXTUResourceUsageEntry

### CXTUResourceUsage

struct CXTUResourceUsage

一个 CXTranslationUnit的内存使用，分类别。

## Typedefs

### CXTUResourceUsageEntry

typedef struct CXTUResourceUsageEntry CXTUResourceUsageEntry

### CXTUResourceUsage

typedef struct CXTUResourceUsage CXTUResourceUsage

一个 CXTranslationUnit的内存使用，分类别。

## 枚举

### CXTranslationUnit\_Flags

enum CXTranslationUnit\_Flags {

CXTranslationUnit\_None = 0x0, CXTranslationUnit\_DetailedPreprocessingRecord = 0x01, CXTranslationUnit\_Incomplete = 0x02, CXTranslationUnit\_PrecompiledPreamble = 0x04,

CXTranslationUnit\_CacheCompletionResults = 0x08, CXTranslationUnit\_ForSerialization = 0x10, CXTranslationUnit\_CXXChainedPCH = 0x20, CXTranslationUnit\_SkipFunctionBodies = 0x40,

CXTranslationUnit\_IncludeBriefCommentsInCodeCompletion = 0x80, CXTranslationUnit\_CreatePreambleOnFirstParse = 0x100, CXTranslationUnit\_KeepGoing = 0x200, CXTranslationUnit\_SingleFileParse = 0x400,

CXTranslationUnit\_LimitSkipFunctionBodiesToPreamble = 0x800, CXTranslationUnit\_IncludeAttributedTypes = 0x1000, CXTranslationUnit\_VisitImplicitAttributes = 0x2000, CXTranslationUnit\_IgnoreNonErrorsFromIncludedFiles = 0x4000,

CXTranslationUnit\_RetainExcludedConditionalBlocks = 0x8000

}

用于控制翻译单元创建的Flags。

### CXSaveTranslationUnit\_Flags

enum CXSaveTranslationUnit\_Flags { CXSaveTranslationUnit\_None = 0x0 }

用于控制翻译单元怎样保存的Flags。

### CXSaveError

enum CXSaveError { CXSaveError\_None = 0, CXSaveError\_Unknown = 1, CXSaveError\_TranslationErrors = 2, CXSaveError\_InvalidTU = 3 }

用于描述在调用 clang\_saveTranslationUnit()中发生的某种错误。

### CXReparse\_Flags

enum CXReparse\_Flags { CXReparse\_None = 0x0 }

Flags that control the reparsing of translation units. More...

用于控制翻译单元的reparsing 的Flags。

### CXTUResourceUsageKind

enum CXTUResourceUsageKind {

CXTUResourceUsage\_AST = 1, CXTUResourceUsage\_Identifiers = 2, CXTUResourceUsage\_Selectors = 3, CXTUResourceUsage\_GlobalCompletionResults = 4,

CXTUResourceUsage\_SourceManagerContentCache = 5, CXTUResourceUsage\_AST\_SideTables = 6, CXTUResourceUsage\_SourceManager\_Membuffer\_Malloc = 7, CXTUResourceUsage\_SourceManager\_Membuffer\_MMap = 8,

CXTUResourceUsage\_ExternalASTSource\_Membuffer\_Malloc = 9, CXTUResourceUsage\_ExternalASTSource\_Membuffer\_MMap = 10, CXTUResourceUsage\_Preprocessor = 11, CXTUResourceUsage\_PreprocessingRecord = 12,

CXTUResourceUsage\_SourceManager\_DataStructures = 13, CXTUResourceUsage\_Preprocessor\_HeaderSearch = 14, CXTUResourceUsage\_MEMORY\_IN\_BYTES\_BEGIN = CXTUResourceUsage\_AST, CXTUResourceUsage\_MEMORY\_IN\_BYTES\_END,

CXTUResourceUsage\_First = CXTUResourceUsage\_AST, CXTUResourceUsage\_Last = CXTUResourceUsage\_Preprocessor\_HeaderSearch

}

对一个翻译单元使用的内存进行进行分类。

## 函数

### clang\_getTranslationUnitSpelling

CINDEX\_LINKAGE CXString clang\_getTranslationUnitSpelling (CXTranslationUnit CTUnit)

Get the original translation unit source file name. More...

获取原始的翻译单元源文件名称。

### clang\_createTranslationUnitFromSourceFile

CINDEX\_LINKAGE CXTranslationUnit clang\_createTranslationUnitFromSourceFile (CXIndex CIdx, const char \*source\_filename, int num\_clang\_command\_line\_args, const char \*const \*clang\_command\_line\_args, unsigned num\_unsaved\_files, struct CXUnsavedFile \*unsaved\_files)

为给定的源文件返回CXTranslationUnit，并且返回要传递给编译器的已经提供的命令行参数。

### clang\_createTranslationUnit

CINDEX\_LINKAGE CXTranslationUnit clang\_createTranslationUnit (CXIndex CIdx, const char \*ast\_filename)

跟clang\_createTranslationUnit2相同，但是，它返回的是CXTranslationUnit，而不是一个error code。

### clang\_createTranslationUnit2

CINDEX\_LINKAGE enum CXErrorCode clang\_createTranslationUnit2 (CXIndex CIdx, const char \*ast\_filename, CXTranslationUnit \*out\_TU)

从一个AST文件(-emit-ast)创建一个翻译单元。

### clang\_defaultEditingTranslationUnitOptions

CINDEX\_LINKAGE unsigned clang\_defaultEditingTranslationUnitOptions (void)

返回一组flags，这些flags适用于解析一个正在编辑的翻译单元。

### clang\_parseTranslationUnit

CINDEX\_LINKAGE CXTranslationUnit clang\_parseTranslationUnit (CXIndex CIdx, const char \*source\_filename, const char \*const \*command\_line\_args, int num\_command\_line\_args, struct CXUnsavedFile \*unsaved\_files, unsigned num\_unsaved\_files, unsigned options)

跟clang\_parseTranslationUnit2相同，但是，它返回的是CXTranslationUnit，而不是一个error code。

### clang\_parseTranslationUnit2

CINDEX\_LINKAGE enum CXErrorCode clang\_parseTranslationUnit2 (CXIndex CIdx, const char \*source\_filename, const char \*const \*command\_line\_args, int num\_command\_line\_args, struct CXUnsavedFile \*unsaved\_files, unsigned num\_unsaved\_files, unsigned options, CXTranslationUnit \*out\_TU)

解析给定的源文件和该源文件对应的翻译单元。

### clang\_parseTranslationUnit2FullArgv

CINDEX\_LINKAGE enum CXErrorCode clang\_parseTranslationUnit2FullArgv (CXIndex CIdx, const char \*source\_filename, const char \*const \*command\_line\_args, int num\_command\_line\_args, struct CXUnsavedFile \*unsaved\_files, unsigned num\_unsaved\_files, unsigned options, CXTranslationUnit \*out\_TU)

跟 clang\_parseTranslationUnit2相同;但是，因为command\_line\_args包含argv[0]，因此，它需要整个命令行。

### clang\_defaultSaveOptions

CINDEX\_LINKAGE unsigned clang\_defaultSaveOptions (CXTranslationUnit TU)

返回一组flags，这些flags适用于保存一个翻译单元。

### clang\_saveTranslationUnit

CINDEX\_LINKAGE int clang\_saveTranslationUnit (CXTranslationUnit TU, const char \*FileName, unsigned options)

保存一个翻译单元到disk上的该翻译单元的序列化表示。

### clang\_suspendTranslationUnit

CINDEX\_LINKAGE unsigned clang\_suspendTranslationUnit (CXTranslationUnit)

悬挂一个翻译单元，来释放该翻译单元关联的内存。

### clang\_disposeTranslationUnit

CINDEX\_LINKAGE void clang\_disposeTranslationUnit (CXTranslationUnit)

销毁一个特定的CXTranslationUnit 对象。

### clang\_defaultReparseOptions

CINDEX\_LINKAGE unsigned clang\_defaultReparseOptions (CXTranslationUnit TU)

返回一组flags，这些flags适用于一个翻译单元的reparsing。

### clang\_reparseTranslationUnit

CINDEX\_LINKAGE int clang\_reparseTranslationUnit (CXTranslationUnit TU, unsigned num\_unsaved\_files, struct CXUnsavedFile \*unsaved\_files, unsigned options)

reparse由当前翻译单元产生源文件。

### clang\_getTUResourceUsageName

const CINDEX\_LINKAGE char \* clang\_getTUResourceUsageName (enum CXTUResourceUsageKind kind)

返回human-readable null-terminated C 字符串，该字符串标识内存类别的名称。

### clang\_getCXTUResourceUsage

CINDEX\_LINKAGE CXTUResourceUsage clang\_getCXTUResourceUsage (CXTranslationUnit TU)

返回一个翻译单元的内存使用。

### clang\_disposeCXTUResourceUsage

CINDEX\_LINKAGE void clang\_disposeCXTUResourceUsage (CXTUResourceUsage usage)

### clang\_getTranslationUnitTargetInfo

CINDEX\_LINKAGE CXTargetInfo clang\_getTranslationUnitTargetInfo (CXTranslationUnit CTUnit)

获取该翻译单元的目标信息。

### clang\_TargetInfo\_dispose

CINDEX\_LINKAGE void clang\_TargetInfo\_dispose (CXTargetInfo Info)

销毁CXTargetInfo 对象。

### clang\_TargetInfo\_getTriple

CINDEX\_LINKAGE CXString clang\_TargetInfo\_getTriple (CXTargetInfo Info)

以字符串方式，获取“normalized target triple”。

### clang\_TargetInfo\_getPointerWidth

CINDEX\_LINKAGE int clang\_TargetInfo\_getPointerWidth (CXTargetInfo Info)

以bits形式，获取目标的指针宽度。

# libclang模块:光标操作

光标操作的协作图。

## Classes

### CXPlatformAvailability

struct CXPlatformAvailability

描述了特定平台上的给定实体的有效性;例如，一个特定的class可能只在Mac OS10.7或更新上可用。

## Typedefs

### CXPlatformAvailability

typedef struct CXPlatformAvailability CXPlatformAvailability

描述了特定平台上的给定实体的有效性;例如，一个特定的class可能只在Mac OS10.7或更新上可用。

### CXCursorSet

typedef struct CXCursorSetImpl \* CXCursorSet

一个代表一组CXCursors的fast container。

## 枚举

### CXLinkageKind

enum CXLinkageKind {

CXLinkage\_Invalid, CXLinkage\_NoLinkage, CXLinkage\_Internal, CXLinkage\_UniqueExternal,

CXLinkage\_External

}

描述由光标指向的实体的链接。

### CXVisibilityKind

enum CXVisibilityKind { CXVisibility\_Invalid, CXVisibility\_Hidden, CXVisibility\_Protected, CXVisibility\_Default }

### CXLanguageKind

enum CXLanguageKind { CXLanguage\_Invalid = 0, CXLanguage\_C, CXLanguage\_ObjC, CXLanguage\_CPlusPlus }

描述由光标指向的实体的“language”。

### CXTLSKind

enum CXTLSKind { CXTLS\_None = 0, CXTLS\_Dynamic, CXTLS\_Static }

描述由光标指向的声明的 "thread-local storage (TLS) kind"。

## 函数

### clang\_getNullCursor

CINDEX\_LINKAGE CXCursor clang\_getNullCursor (void)

检索NULL 光标，它代表“no entity”。

### clang\_getTranslationUnitCursor

CINDEX\_LINKAGE CXCursor clang\_getTranslationUnitCursor (CXTranslationUnit)

检索光标，它代表给定的翻译单元。

### clang\_equalCursors

CINDEX\_LINKAGE unsigned clang\_equalCursors (CXCursor, CXCursor)

确定两个光标是否是等价的。

### clang\_Cursor\_isNull

CINDEX\_LINKAGE int clang\_Cursor\_isNull (CXCursor cursor)

如果光标是null，则返回非零。

### clang\_hashCursor

CINDEX\_LINKAGE unsigned clang\_hashCursor (CXCursor)

为给定的光标计算一个hash值。

### clang\_getCursorKind

CINDEX\_LINKAGE enum CXCursorKind clang\_getCursorKind (CXCursor)

检索给定光标的类型。

### clang\_isDeclaration

CINDEX\_LINKAGE unsigned clang\_isDeclaration (enum CXCursorKind)

确定给定光标的类型是否标识一个声明。

### clang\_isInvalidDeclaration

CINDEX\_LINKAGE unsigned clang\_isInvalidDeclaration (CXCursor)

确定给定的声明是否合法。

### clang\_isReference

CINDEX\_LINKAGE unsigned clang\_isReference (enum CXCursorKind)

确定给定光标类型是否标识一个简单的引用。

### clang\_isExpression

CINDEX\_LINKAGE unsigned clang\_isExpression (enum CXCursorKind)

确定给定光标类型是否标识一个表达式。

### clang\_isStatement

CINDEX\_LINKAGE unsigned clang\_isStatement (enum CXCursorKind)

确定给定光标类型是否标识一个语句。

### clang\_isAttribute

CINDEX\_LINKAGE unsigned clang\_isAttribute (enum CXCursorKind)

确定一个光标类型是否标识一个属性。

### clang\_Cursor\_hasAttrs

CINDEX\_LINKAGE unsigned clang\_Cursor\_hasAttrs (CXCursor C)

确定光标类型是否有任一属性。

### clang\_isInvalid

CINDEX\_LINKAGE unsigned clang\_isInvalid (enum CXCursorKind)

确定给定光标类型是否标识一个非法的光标。

### clang\_isTranslationUnit

CINDEX\_LINKAGE unsigned clang\_isTranslationUnit (enum CXCursorKind)

确定给定光标类型是否标识一个翻译单元。

### clang\_isPreprocessing

CINDEX\_LINKAGE unsigned clang\_isPreprocessing (enum CXCursorKind)

### clang\_isUnexposed

CINDEX\_LINKAGE unsigned clang\_isUnexposed (enum CXCursorKind)

### clang\_getCursorLinkage

CINDEX\_LINKAGE enum CXLinkageKind clang\_getCursorLinkage (CXCursor cursor)

确定给定光标指定的实体的链接。

### clang\_getCursorVisibility

CINDEX\_LINKAGE enum CXVisibilityKind clang\_getCursorVisibility (CXCursor cursor)

描述光标指定的实体的可见性。

### clang\_getCursorAvailability

CINDEX\_LINKAGE enum CXAvailabilityKind clang\_getCursorAvailability (CXCursor cursor)

确定当前光标指向的实体的有效性，要考虑到当前的目标平台。

### clang\_getCursorPlatformAvailability

CINDEX\_LINKAGE int clang\_getCursorPlatformAvailability (CXCursor cursor, int \*always\_deprecated, CXString \*deprecated\_message, int \*always\_unavailable, CXString \*unavailable\_message, CXPlatformAvailability \*availability, int availability\_size)

确定当前光标指向的实体对于已知的所有可用平台的有效性。

### clang\_disposeCXPlatformAvailability

CINDEX\_LINKAGE void clang\_disposeCXPlatformAvailability (CXPlatformAvailability \*availability)

释放一个CXPlatformAvailability 对象关联的内存。

### clang\_Cursor\_getVarDeclInitializer

CINDEX\_LINKAGE CXCursor clang\_Cursor\_getVarDeclInitializer (CXCursor cursor)

如果光标指向一个变量的声明并且它有初始化器，则返回指向初始化器的光标;否则，返回null 光标。

### clang\_Cursor\_hasVarDeclGlobalStorage

CINDEX\_LINKAGE int clang\_Cursor\_hasVarDeclGlobalStorage (CXCursor cursor)

如果光标指向了一个变量声明且该有全局存储，则返回1。

### clang\_Cursor\_hasVarDeclExternalStorage

CINDEX\_LINKAGE int clang\_Cursor\_hasVarDeclExternalStorage (CXCursor cursor)

如果光标指向一个变量声明且声明有外部存储，则返回1.

### clang\_getCursorLanguage

CINDEX\_LINKAGE enum CXLanguageKind clang\_getCursorLanguage (CXCursor cursor)

确定给定光标指向的实体的“language”。

### clang\_getCursorTLSKind

CINDEX\_LINKAGE enum CXTLSKind clang\_getCursorTLSKind (CXCursor cursor)

确定光标指向的声明的"thread-local storage (TLS) kind"。

### clang\_Cursor\_getTranslationUnit

CINDEX\_LINKAGE CXTranslationUnit clang\_Cursor\_getTranslationUnit (CXCursor)

返回光标起源的翻译单元。

### clang\_createCXCursorSet

CINDEX\_LINKAGE CXCursorSet clang\_createCXCursorSet (void)

创建一个空的CXCursorSet。

### clang\_disposeCXCursorSet

CINDEX\_LINKAGE void clang\_disposeCXCursorSet (CXCursorSet cset)

销毁一个CXCursorSet且释放相关联的内存。

### clang\_CXCursorSet\_contains

CINDEX\_LINKAGE unsigned clang\_CXCursorSet\_contains (CXCursorSet cset, CXCursor cursor)

查询一个CXCursorSet，来查看它是否包含有特定的CXCursor。

### clang\_CXCursorSet\_insert

CINDEX\_LINKAGE unsigned clang\_CXCursorSet\_insert (CXCursorSet cset, CXCursor cursor)

在一个CXCursorSet插入一个CXCursor 。

### clang\_getCursorSemanticParent

CINDEX\_LINKAGE CXCursor clang\_getCursorSemanticParent (CXCursor cursor)

确定给定光标的语义父亲。

### clang\_getCursorLexicalParent

CINDEX\_LINKAGE CXCursor clang\_getCursorLexicalParent (CXCursor cursor)

确定给定光标的词法父亲。

### clang\_getOverriddenCursors

CINDEX\_LINKAGE void clang\_getOverriddenCursors (CXCursor cursor, CXCursor \*\*overridden, unsigned \*num\_overridden)

确定给定的method是否override 一组methods。

### clang\_disposeOverriddenCursors

CINDEX\_LINKAGE void clang\_disposeOverriddenCursors (CXCursor \*overridden)

释放由clang\_getOverriddenCursors()返回的一组 overridden 光标。

### clang\_getIncludedFile

CINDEX\_LINKAGE CXFile clang\_getIncludedFile (CXCursor cursor)

检索由给定包含指令光标包含的文件。

# libclang模块:光标与源代码之间的映射

光标代表抽象语法树（AST）中的位置。这些例程有助于在光标和源代码中描述的实体出现的物理位置之间映射。映射均以两个方向提供，因此可以从源代码到AST和反向映射。

## 函数

### clang\_getCursor

CINDEX\_LINKAGE CXCursor clang\_getCursor (CXTranslationUnit, CXSourceLocation)

从源代码位置映射到光标，该光标描述了源代码位置上的实体。

### clang\_getCursorLocation

CINDEX\_LINKAGE CXSourceLocation clang\_getCursorLocation (CXCursor)

检索源构造器(不知道是否是指构造函数)的物理位置，该源构造器由给定的光标指定。

### clang\_getCursorExtent

CINDEX\_LINKAGE CXSourceRange clang\_getCursorExtent (CXCursor)

Retrieve the physical extent of the source construct referenced by the given cursor. More...

检索源构造器(不知道是否是指构造函数)的物理范围，该源构造器由给定的光标指定。

# libclang模块:CXCursors的类型信息

## Classes

### CXType

struct CXType

抽象语法树的元素的类型。

## 枚举

### CXTypeKind

类型的种类。

### CXCallingConv

enum CXCallingConv {

CXCallingConv\_Default = 0, CXCallingConv\_C = 1, CXCallingConv\_X86StdCall = 2, CXCallingConv\_X86FastCall = 3,

CXCallingConv\_X86ThisCall = 4, CXCallingConv\_X86Pascal = 5, CXCallingConv\_AAPCS = 6, CXCallingConv\_AAPCS\_VFP = 7,

CXCallingConv\_X86RegCall = 8, CXCallingConv\_IntelOclBicc = 9, CXCallingConv\_Win64 = 10, CXCallingConv\_X86\_64Win64 = CXCallingConv\_Win64,

CXCallingConv\_X86\_64SysV = 11, CXCallingConv\_X86VectorCall = 12, CXCallingConv\_Swift = 13, CXCallingConv\_PreserveMost = 14,

CXCallingConv\_PreserveAll = 15, CXCallingConv\_AArch64VectorCall = 16, CXCallingConv\_SwiftAsync = 17, CXCallingConv\_AArch64SVEPCS = 18,

CXCallingConv\_Invalid = 100, CXCallingConv\_Unexposed = 200

}

描述函数类型的调用规范。

### CXTemplateArgumentKind

enum CXTemplateArgumentKind {

CXTemplateArgumentKind\_Null, CXTemplateArgumentKind\_Type, CXTemplateArgumentKind\_Declaration, CXTemplateArgumentKind\_NullPtr,

CXTemplateArgumentKind\_Integral, CXTemplateArgumentKind\_Template, CXTemplateArgumentKind\_TemplateExpansion, CXTemplateArgumentKind\_Expression,

CXTemplateArgumentKind\_Pack, CXTemplateArgumentKind\_Invalid

}

描述模板参数的类型。

### CXTypeNullabilityKind

enum CXTypeNullabilityKind {

CXTypeNullability\_NonNull = 0, CXTypeNullability\_Nullable = 1, CXTypeNullability\_Unspecified = 2, CXTypeNullability\_Invalid = 3,

CXTypeNullability\_NullableResult = 4

}

### CXTypeLayoutError

enum CXTypeLayoutError {

CXTypeLayoutError\_Invalid = -1, CXTypeLayoutError\_Incomplete = -2, CXTypeLayoutError\_Dependent = -3, CXTypeLayoutError\_NotConstantSize = -4,

CXTypeLayoutError\_InvalidFieldName = -5, CXTypeLayoutError\_Undeduced = -6

}

clang\_Type\_getSizeOf, clang\_Type\_getAlignOf, clang\_Type\_getOffsetOf 和clang\_Cursor\_getOffsetOf 的可能出错码列表。

### CXRefQualifierKind

enum CXRefQualifierKind { CXRefQualifier\_None = 0, CXRefQualifier\_LValue, CXRefQualifier\_RValue }

### CXXAccessSpecifier

enum CX\_CXXAccessSpecifier { CX\_CXXInvalidAccessSpecifier, CX\_CXXPublic, CX\_CXXProtected, CX\_CXXPrivate }

代表C++对基类的访问控制等级， 该基类由带有CX\_CXXBaseSpecifier的光标指定。

### CX\_StorageClass

enum CX\_StorageClass {

CX\_SC\_Invalid, CX\_SC\_None, CX\_SC\_Extern, CX\_SC\_Static,

CX\_SC\_PrivateExtern, CX\_SC\_OpenCLWorkGroupLocal, CX\_SC\_Auto, CX\_SC\_Register

}

代表跟源代码中声明的一样的存储classes。

## 函数

### clang\_getCursorType

CINDEX\_LINKAGE CXType clang\_getCursorType (CXCursor C)

检索CXCursor (如果有的话)的类型。

### clang\_getTypeSpelling

CINDEX\_LINKAGE CXString clang\_getTypeSpelling (CXType CT)

使用它的翻译单元的语言规则来打印基础类型。

### clang\_getTypedefDeclUnderlyingType

CINDEX\_LINKAGE CXType clang\_getTypedefDeclUnderlyingType (CXCursor C)

检索Typedef声明的基本类型。

### clang\_getEnumDeclIntegerType

CINDEX\_LINKAGE CXType clang\_getEnumDeclIntegerType (CXCursor C)

检索枚举声明的整数类型。

### clang\_getEnumConstantDeclValue

CINDEX\_LINKAGE long long clang\_getEnumConstantDeclValue (CXCursor C)

检索声明为signed long long的枚举常数的整数值。

### clang\_getEnumConstantDeclUnsignedValue

CINDEX\_LINKAGE unsigned long long clang\_getEnumConstantDeclUnsignedValue (CXCursor C)

检索声明为unsigned long long的枚举常数的整数值。

### clang\_getFieldDeclBitWidth

CINDEX\_LINKAGE int clang\_getFieldDeclBitWidth (CXCursor C)

Retrieve the bit width of a bit field declaration as an integer. More...

检索一个位域声明为整数的位宽。

### clang\_Cursor\_getNumArguments

CINDEX\_LINKAGE int clang\_Cursor\_getNumArguments (CXCursor C)

Retrieve the number of non-variadic arguments associated with a given cursor. More...

检索给定光标关联的 non-variadic(非可变)参数的数量。

### clang\_Cursor\_getArgument

CINDEX\_LINKAGE CXCursor clang\_Cursor\_getArgument (CXCursor C, unsigned i)

检索一个函数或方法的参数光标。

### clang\_Cursor\_getNumTemplateArguments

CINDEX\_LINKAGE int clang\_Cursor\_getNumTemplateArguments (CXCursor C)

检索一个函数、struct或class decl 的模板参数的个数，它们代表了一个模板的特化。

### clang\_Cursor\_getTemplateArgumentKind

CINDEX\_LINKAGE enum CXTemplateArgumentKind clang\_Cursor\_getTemplateArgumentKind (CXCursor C, unsigned I)

检索CXCursor C的第I个模板参数的类型。

### clang\_Cursor\_getTemplateArgumentType

CINDEX\_LINKAGE CXType clang\_Cursor\_getTemplateArgumentType (CXCursor C, unsigned I)

检索一个CXType ，该CXType 代表一个函数decl 的 TemplateArgument的类型，代表了一个模板的特化。

### clang\_Cursor\_getTemplateArgumentValue

CINDEX\_LINKAGE long long clang\_Cursor\_getTemplateArgumentValue (CXCursor C, unsigned I)

获取一个完整的TemplateArgument 的值，该值是signed long long类型。TemplateArgument在一个函数的decl内，代表一个模板的特化。

### clang\_Cursor\_getTemplateArgumentUnsignedValue

CINDEX\_LINKAGE unsigned long long clang\_Cursor\_getTemplateArgumentUnsignedValue (CXCursor C, unsigned I)

获取一个完整的TemplateArgument 的值，该值是unsigned long long类型。TemplateArgument在一个函数的decl内，代表一个模板的特化。

### clang\_equalTypes

CINDEX\_LINKAGE unsigned clang\_equalTypes (CXType A, CXType B)

确定两个CXTypes 是否是代表同一个类型。

### clang\_getCanonicalType

CINDEX\_LINKAGE CXType clang\_getCanonicalType (CXType T)

为CXType返回规范的类型。

### clang\_isConstQualifiedType

CINDEX\_LINKAGE unsigned clang\_isConstQualifiedType (CXType T)

确定CXType是有“const”限定符，不遍历可能在不同level撒谎那个添加“const”的typedefs 。

### clang\_Cursor\_isMacroFunctionLike

CINDEX\_LINKAGE unsigned clang\_Cursor\_isMacroFunctionLike (CXCursor C)

Determine whether a CXCursor that is a macro, is function like. More...

确定CXCursor 是否是一个像函数的宏。

clang\_Cursor\_isMacroBuiltin

CINDEX\_LINKAGE unsigned clang\_Cursor\_isMacroBuiltin (CXCursor C)

确定CXCursor 是否是一个内置的宏。

### clang\_Cursor\_isFunctionInlined

CINDEX\_LINKAGE unsigned clang\_Cursor\_isFunctionInlined (CXCursor C)

确定CXCursor 是否是一个函数声明，一个inline的声明。

### clang\_isVolatileQualifiedType

CINDEX\_LINKAGE unsigned clang\_isVolatileQualifiedType (CXType T)

确定CXType 是否有“volatile”的限定符，不遍历可能在不同level添加“volatile”的typedefs 。

### clang\_isRestrictQualifiedType

CINDEX\_LINKAGE unsigned clang\_isRestrictQualifiedType (CXType T)

Determine whether a CXType has the "restrict" qualifier set, without looking through typedefs that may have added "restrict" at a different level. More...

确定CXType 是否有“restrict”的限定符，不遍历可能在不同level添加“restrict”的typedefs 。

### clang\_getAddressSpace

CINDEX\_LINKAGE unsigned clang\_getAddressSpace (CXType T)

返回给定type的地址空间。

### clang\_getTypedefName

CINDEX\_LINKAGE CXString clang\_getTypedefName (CXType CT)

返回给定type的typedef 名称。

### clang\_getPointeeType

CINDEX\_LINKAGE CXType clang\_getPointeeType (CXType T)

对于指针类型，返回它指向的单元的类型。

### clang\_getUnqualifiedType

CINDEX\_LINKAGE CXType clang\_getUnqualifiedType (CXType CT)

检索给定type的unqualified variant，尽可能的移除掉它。

### clang\_getNonReferenceType

CINDEX\_LINKAGE CXType clang\_getNonReferenceType (CXType CT)

对于引用类型(例如，”const int&”)，返回该引用指向的类型(例如，”const int”)。

### clang\_getTypeDeclaration

CINDEX\_LINKAGE CXCursor clang\_getTypeDeclaration (CXType T)

返回给定type的声明对应的光标。

### clang\_getDeclObjCTypeEncoding

CINDEX\_LINKAGE CXString clang\_getDeclObjCTypeEncoding (CXCursor C)

为特定的声明返回Objective-C 类型的编码。

### clang\_Type\_getObjCEncoding

CINDEX\_LINKAGE CXString clang\_Type\_getObjCEncoding (CXType type)

为特定的CXType返回Objective-C 类型的编码。

### clang\_getTypeKindSpelling

CINDEX\_LINKAGE CXString clang\_getTypeKindSpelling (enum CXTypeKind K)

检索给定CXTypeKind的拼写。

### clang\_getFunctionTypeCallingConv

CINDEX\_LINKAGE enum CXCallingConv clang\_getFunctionTypeCallingConv (CXType T)

检索函数类型关联的调用惯例。

### clang\_getResultType

CINDEX\_LINKAGE CXType clang\_getResultType (CXType T)

检索函数类型关联的return 类型。

### clang\_getExceptionSpecificationType

CINDEX\_LINKAGE int clang\_getExceptionSpecificationType (CXType T)

检索一个函数类型关联的特定异常。

### clang\_getNumArgTypes

CINDEX\_LINKAGE int clang\_getNumArgTypes (CXType T)

检索一个函数类型关联的非法参数的数量。

### clang\_getArgType

CINDEX\_LINKAGE CXType clang\_getArgType (CXType T, unsigned i)

检索函数类型的参数的类型。

### clang\_Type\_getObjCObjectBaseType

CINDEX\_LINKAGE CXType clang\_Type\_getObjCObjectBaseType (CXType T)

检索ObjCObjectType的基础类型。

### clang\_Type\_getNumObjCProtocolRefs

CINDEX\_LINKAGE unsigned clang\_Type\_getNumObjCProtocolRefs (CXType T)

检索一个 ObjC object/id关联的协议引用的数量。

### clang\_Type\_getObjCProtocolDecl

CINDEX\_LINKAGE CXCursor clang\_Type\_getObjCProtocolDecl (CXType T, unsigned i)

检索一个 ObjC object/id的协议引用的decl。

### clang\_Type\_getNumObjCTypeArgs

CINDEX\_LINKAGE unsigned clang\_Type\_getNumObjCTypeArgs (CXType T)

检索一个ObjC对象关联的类型参数的数量。

### clang\_Type\_getObjCTypeArg

CINDEX\_LINKAGE CXType clang\_Type\_getObjCTypeArg (CXType T, unsigned i)

检索一个ObjC对象关联的类型参数。

### clang\_isFunctionTypeVariadic

CINDEX\_LINKAGE unsigned clang\_isFunctionTypeVariadic (CXType T)

如果CXType 是一个合法的函数类型，则返回1;否则，返回0.

### clang\_getCursorResultType

CINDEX\_LINKAGE CXType clang\_getCursorResultType (CXCursor C)

检索给定光标关联的return type。

### clang\_getCursorExceptionSpecificationType

CINDEX\_LINKAGE int clang\_getCursorExceptionSpecificationType (CXCursor C)

检索给定光标关联的特定异常类型。

### clang\_isPODType

CINDEX\_LINKAGE unsigned clang\_isPODType (CXType T)

如果CXType 是一个POD(plain old data)类型，返回1; 否则， 返回0.

### clang\_getElementType

CINDEX\_LINKAGE CXType clang\_getElementType (CXType T)

返回一个array、complex或vector类型的元素类型。

### clang\_getNumElements

CINDEX\_LINKAGE long long clang\_getNumElements (CXType T)

返回一个array或vector类型的元素个数。

### clang\_getArrayElementType

CINDEX\_LINKAGE CXType clang\_getArrayElementType (CXType T)

返回一个array类型的元素类型。

### clang\_getArraySize

CINDEX\_LINKAGE long long clang\_getArraySize (CXType T)

返回一个array常量的array的大小。

### clang\_Type\_getNamedType

CINDEX\_LINKAGE CXType clang\_Type\_getNamedType (CXType T)

检索由qualified-id命名的类型。

### clang\_Type\_isTransparentTagTypedef

CINDEX\_LINKAGE unsigned clang\_Type\_isTransparentTagTypedef (CXType T)

确定一个typedef是否是一个 'transparent' tag。

### clang\_Type\_getNullability

CINDEX\_LINKAGE enum CXTypeNullabilityKind clang\_Type\_getNullability (CXType T)

检索指针类型的无效性。

### clang\_Type\_getAlignOf

CINDEX\_LINKAGE long long clang\_Type\_getAlignOf (CXType T)

以bytes方式，返回一个类型的对齐，就像C++[expr.alignof] 标准一样。

### clang\_Type\_getClassType

CINDEX\_LINKAGE CXType clang\_Type\_getClassType (CXType T)

返回一个成员指针类型的class 类型。

### clang\_Type\_getSizeOf

CINDEX\_LINKAGE long long clang\_Type\_getSizeOf (CXType T)

返回类型的占用空间大小，以bytes方式，就像C++[expr.sizeof] 标准一样。

### clang\_Type\_getOffsetOf

CINDEX\_LINKAGE long long clang\_Type\_getOffsetOf (CXType T, const char \*S)

返回在类型T的一个record中的名为S的field的偏移，以bits方式;就像 C++11[18.2p4]中由offsetof 返回的一样。

### clang\_Type\_getModifiedType

CINDEX\_LINKAGE CXType clang\_Type\_getModifiedType (CXType T)

返回该属性类型修改的类型。

### clang\_Type\_getValueType

CINDEX\_LINKAGE CXType clang\_Type\_getValueType (CXType CT)

获取该atomic 类型包含的类型。

### clang\_Cursor\_getOffsetOfField

CINDEX\_LINKAGE long long clang\_Cursor\_getOffsetOfField (CXCursor C)

返回由光标代表的域的offset。

### clang\_Cursor\_isAnonymous

CINDEX\_LINKAGE unsigned clang\_Cursor\_isAnonymous (CXCursor C)

确定给定光标是否代表一个匿名的tag或命名空间。

### clang\_Cursor\_isAnonymousRecordDecl

CINDEX\_LINKAGE unsigned clang\_Cursor\_isAnonymousRecordDecl (CXCursor C)

确定给定的光标是否代表一个匿名的record 声明。

### clang\_Cursor\_isInlineNamespace

CINDEX\_LINKAGE unsigned clang\_Cursor\_isInlineNamespace (CXCursor C)

确定给定光标是否代表一个inline 名字空间的声明。

### clang\_Type\_getNumTemplateArguments

CINDEX\_LINKAGE int clang\_Type\_getNumTemplateArguments (CXType T)

返回给定模板特化的模板参数的数量; 或者是-1,如果类型T不是一个模板的特化。

### clang\_Type\_getTemplateArgumentAsType

CINDEX\_LINKAGE CXType clang\_Type\_getTemplateArgumentAsType (CXType T, unsigned i)

返回一个给定index的模板class特化的类型模板参数。

### clang\_Type\_getCXXRefQualifier

CINDEX\_LINKAGE enum CXRefQualifierKind clang\_Type\_getCXXRefQualifier (CXType T)

索引一个函数或方法的ref-qualifier类型(ref qualifier 修饰的是引用的 this)。

### clang\_Cursor\_isBitField

CINDEX\_LINKAGE unsigned clang\_Cursor\_isBitField (CXCursor C)

如果光标指定的Record成员是一个bitfield，则返回非零。

### clang\_isVirtualBase

CINDEX\_LINKAGE unsigned clang\_isVirtualBase (CXCursor)

如果带有类别 CX\_CXXBaseSpecifier的光标指定的基类是virtual，则返回1.

### clang\_getCXXAccessSpecifier

CINDEX\_LINKAGE enum CX\_CXXAccessSpecifier clang\_getCXXAccessSpecifier (CXCursor)

为一个索引的对象返回访问控制level。

### clang\_Cursor\_getStorageClass

CINDEX\_LINKAGE enum CX\_StorageClass clang\_Cursor\_getStorageClass (CXCursor)

为一个函数或变量声明返回存储 class。

### clang\_getNumOverloadedDecls

CINDEX\_LINKAGE unsigned clang\_getNumOverloadedDecls (CXCursor cursor)

确定由一个CXCursor\_OverloadedDeclRef 光标引用的overloaded 声明的个数。

### clang\_getOverloadedDecl

CINDEX\_LINKAGE CXCursor clang\_getOverloadedDecl (CXCursor cursor, unsigned index)

检索由一个CXCursor\_OverloadedDeclRef 光标引用的overloaded 声明的一个光标。

# libclang模块:属性信息

## 函数

### clang\_getIBOutletCollectionType

CINDEX\_LINKAGE CXType clang\_getIBOutletCollectionType (CXCursor)

对于代表iboutletcollection 属性的光标，此函数返回集合元素类型。

# libclang模块:使用光标遍历AST

这些例程提供了使用光标来便利AST的能力。

## Typedefs

### CXCursorVisitor

typedef enum CXChildVisitResult(\* CXCursorVisitor) (CXCursor cursor, CXCursor parent, CXClientData client\_data)

由遍历建立的，为每个光标唤醒的visitor。

## 枚举

### CXChildVisitResult

enum CXChildVisitResult { CXChildVisit\_Break, CXChildVisit\_Continue, CXChildVisit\_Recurse }

描述特定光标的子光标的遍历在访问特定子光标后，应如何进行。

## 函数

### clang\_visitChildren

CINDEX\_LINKAGE unsigned clang\_visitChildren (CXCursor parent, CXCursorVisitor visitor, CXClientData client\_data)

访问特定光标的子光标。

# libclang模块:AST的交叉引用

这些例程提供了在翻译单元内或跨越翻译单元确定引用的能力; 提供光标引用的实体的名称，遵循引用光标到它们引用的声明，关联它们的定义到声明。

## Typedefs

### CXPrintingPolicy

typedef void \* CXPrintingPolicy

不透明的指针代表控制Clang\_getCursorPrettyprinted的漂亮打印的策略。

## 枚举

### CXPrintingPolicyProperty

enum CXPrintingPolicyProperty {

CXPrintingPolicy\_Indentation, CXPrintingPolicy\_SuppressSpecifiers, CXPrintingPolicy\_SuppressTagKeyword, CXPrintingPolicy\_IncludeTagDefinition,

CXPrintingPolicy\_SuppressScope, CXPrintingPolicy\_SuppressUnwrittenScope, CXPrintingPolicy\_SuppressInitializers, CXPrintingPolicy\_ConstantArraySizeAsWritten,

CXPrintingPolicy\_AnonymousTagLocations, CXPrintingPolicy\_SuppressStrongLifetime, CXPrintingPolicy\_SuppressLifetimeQualifiers, CXPrintingPolicy\_SuppressTemplateArgsInCXXConstructors,

CXPrintingPolicy\_Bool, CXPrintingPolicy\_Restrict, CXPrintingPolicy\_Alignof, CXPrintingPolicy\_UnderscoreAlignof,

CXPrintingPolicy\_UseVoidForZeroParams, CXPrintingPolicy\_TerseOutput, CXPrintingPolicy\_PolishForDeclaration, CXPrintingPolicy\_Half,

CXPrintingPolicy\_MSWChar, CXPrintingPolicy\_IncludeNewlines, CXPrintingPolicy\_MSVCFormatting, CXPrintingPolicy\_ConstantsAsWritten,

CXPrintingPolicy\_SuppressImplicitBase, CXPrintingPolicy\_FullyQualifiedName, CXPrintingPolicy\_LastProperty = CXPrintingPolicy\_FullyQualifiedName

}

打印策略的性质。

### CXObjCPropertyAttrKind

enum CXObjCPropertyAttrKind {

CXObjCPropertyAttr\_noattr = 0x00, CXObjCPropertyAttr\_readonly = 0x01, CXObjCPropertyAttr\_getter = 0x02, CXObjCPropertyAttr\_assign = 0x04,

CXObjCPropertyAttr\_readwrite = 0x08, CXObjCPropertyAttr\_retain = 0x10, CXObjCPropertyAttr\_copy = 0x20, CXObjCPropertyAttr\_nonatomic = 0x40,

CXObjCPropertyAttr\_setter = 0x80, CXObjCPropertyAttr\_atomic = 0x100, CXObjCPropertyAttr\_weak = 0x200, CXObjCPropertyAttr\_strong = 0x400,

CXObjCPropertyAttr\_unsafe\_unretained = 0x800, CXObjCPropertyAttr\_class = 0x1000

}

一个CXCursor\_ObjCPropertyDecl 性质的属性。

### CXObjCDeclQualifierKind

enum CXObjCDeclQualifierKind {

CXObjCDeclQualifier\_None = 0x0, CXObjCDeclQualifier\_In = 0x1, CXObjCDeclQualifier\_Inout = 0x2, CXObjCDeclQualifier\_Out = 0x4,

CXObjCDeclQualifier\_Bycopy = 0x8, CXObjCDeclQualifier\_Byref = 0x10, CXObjCDeclQualifier\_Oneway = 0x20

}

在Objective-C 的方法声明中，限定符写在return 和参数类型之后。

## 函数

clang\_getCursorUSR

CINDEX\_LINKAGE CXString clang\_getCursorUSR (CXCursor)

检索给定光标引用的实体的Unified Symbol Resolution（USR）。

### clang\_constructUSR\_ObjCClass

CINDEX\_LINKAGE CXString clang\_constructUSR\_ObjCClass (const char \*class\_name)

为指定的Objective-C类构建USR。

### clang\_constructUSR\_ObjCCategory

CINDEX\_LINKAGE CXString clang\_constructUSR\_ObjCCategory (const char \*class\_name, const char \*category\_name)

为指定的Objective-C类别构建USR。

### clang\_constructUSR\_ObjCProtocol

CINDEX\_LINKAGE CXString clang\_constructUSR\_ObjCProtocol (const char \*protocol\_name)

为指定的Objective-C协议构建USR。

### clang\_constructUSR\_ObjCIvar

CINDEX\_LINKAGE CXString clang\_constructUSR\_ObjCIvar (const char \*name, CXString classUSR)

为指定的Objective-C实例变量构建USR及其包含类的USR。

### clang\_constructUSR\_ObjCMethod

CINDEX\_LINKAGE CXString clang\_constructUSR\_ObjCMethod (const char \*name, unsigned isInstanceMethod, CXString classUSR)

为指定的Objective-C方法构建USR及其包含类的USR。

### clang\_constructUSR\_ObjCProperty

CINDEX\_LINKAGE CXString clang\_constructUSR\_ObjCProperty (const char \*property, CXString classUSR)

为指定的Objective-C Property构建USR及其包含类的USR。

### clang\_getCursorSpelling

CINDEX\_LINKAGE CXString clang\_getCursorSpelling (CXCursor)

检索该光标引用的实体名称。

### clang\_Cursor\_getSpellingNameRange

CINDEX\_LINKAGE CXSourceRange clang\_Cursor\_getSpellingNameRange (CXCursor, unsigned pieceIndex, unsigned options)

检索构成光标拼写名称的片段范围。

### clang\_PrintingPolicy\_getProperty

CINDEX\_LINKAGE unsigned clang\_PrintingPolicy\_getProperty (CXPrintingPolicy Policy, enum CXPrintingPolicyProperty Property)

获取给定打印策略的property 值。

### clang\_PrintingPolicy\_setProperty

CINDEX\_LINKAGE void clang\_PrintingPolicy\_setProperty (CXPrintingPolicy Policy, enum CXPrintingPolicyProperty Property, unsigned Value)

检索光标的默认策略。

## clang\_getCursorPrintingPolicy

CINDEX\_LINKAGE CXPrintingPolicy clang\_getCursorPrintingPolicy (CXCursor)

检索光标的默认策略。

### clang\_PrintingPolicy\_dispose

CINDEX\_LINKAGE void clang\_PrintingPolicy\_dispose (CXPrintingPolicy Policy)

销毁打印测路。

### clang\_getCursorPrettyPrinted

CINDEX\_LINKAGE CXString clang\_getCursorPrettyPrinted (CXCursor Cursor, CXPrintingPolicy Policy)

漂亮的打印声明。

### clang\_getCursorDisplayName

CINDEX\_LINKAGE CXString clang\_getCursorDisplayName (CXCursor)

检索该光标引用的实体的显示名称。

### clang\_getCursorReferenced

CINDEX\_LINKAGE CXCursor clang\_getCursorReferenced (CXCursor)

对于一个引用的光标，检索代表其引用实体的光标。

### clang\_getCursorDefinition

CINDEX\_LINKAGE CXCursor clang\_getCursorDefinition (CXCursor)

对于光标是对某个实体的引用或声明，检索描述该实体定义的光标。

### clang\_isCursorDefinition

CINDEX\_LINKAGE unsigned clang\_isCursorDefinition (CXCursor)

确定该光标指向的声明是否也是该实体的定义。

### clang\_getCanonicalCursor

CINDEX\_LINKAGE CXCursor clang\_getCanonicalCursor (CXCursor)

检索与给定光标相对应的规范光标。

### clang\_Cursor\_getObjCSelectorIndex

CINDEX\_LINKAGE int clang\_Cursor\_getObjCSelectorIndex (CXCursor)

如果光标指向Objective-C的方法或消息表达式中的选择器标识符，则返回选择器索引。

### clang\_Cursor\_isDynamicCall

CINDEX\_LINKAGE int clang\_Cursor\_isDynamicCall (CXCursor C)

给定指向C ++方法调用或Objective-C消息的光标，如果方法/消息为“dynamic”，则返回非零。

### clang\_Cursor\_getReceiverType

CINDEX\_LINKAGE CXType clang\_Cursor\_getReceiverType (CXCursor C)

给定指向Objective-C消息或property的引用或C ++方法的调用的光标，返回接收器的CXTYPE。

### clang\_Cursor\_getObjCPropertyAttributes

CINDEX\_LINKAGE unsigned clang\_Cursor\_getObjCPropertyAttributes (CXCursor C, unsigned reserved)

给定一个代表 property 声明的光标，请返回关联的property 的属性。

### clang\_Cursor\_getObjCPropertyGetterName

CINDEX\_LINKAGE CXString clang\_Cursor\_getObjCPropertyGetterName (CXCursor C)

给定一个代表property 声明的光标，请返回实现Getter的方法的名称。

### clang\_Cursor\_getObjCPropertySetterName

CINDEX\_LINKAGE CXString clang\_Cursor\_getObjCPropertySetterName (CXCursor C)

给定一个代表property 的声明的光标，请返回实现setter的方法的名称（如果有）。

### clang\_Cursor\_getObjCDeclQualifiers

CINDEX\_LINKAGE unsigned clang\_Cursor\_getObjCDeclQualifiers (CXCursor C)

给定代表Objective-C方法或参数声明的光标，分别返回return type或参数关联的Objective-C限定符。

### clang\_Cursor\_isObjCOptional

CINDEX\_LINKAGE unsigned clang\_Cursor\_isObjCOptional (CXCursor C)

给定一个代表Objective-C方法或property 声明的光标，如果声明受 "\@optional"的影响，则返回非零。

### clang\_Cursor\_isVariadic

CINDEX\_LINKAGE unsigned clang\_Cursor\_isVariadic (CXCursor C)

如果给定的光标是变异函数(variadic function )或方法，则返回非零。

### clang\_Cursor\_isExternalSymbol

CINDEX\_LINKAGE unsigned clang\_Cursor\_isExternalSymbol (CXCursor C, CXString \*language, CXString \*definedIn, unsigned \*isGenerated)

如果给定的光标指向带有 external\_source\_symbol 属性标记的符号，则返回非零。

### clang\_Cursor\_getCommentRange

CINDEX\_LINKAGE CXSourceRange clang\_Cursor\_getCommentRange (CXCursor C)

给定一个代表声明的光标，请返回关联的comment的源范围。

### clang\_Cursor\_getRawCommentText

CINDEX\_LINKAGE CXString clang\_Cursor\_getRawCommentText (CXCursor C)

给定一个代表声明的光标，请返回关联的comment文本，包括comment标记。

### clang\_Cursor\_getBriefCommentText

CINDEX\_LINKAGE CXString clang\_Cursor\_getBriefCommentText (CXCursor C)

给定一个代表可文档实体（例如声明）的光标，返回关联。

# libclang模块:Name Mangling API 功能

## 函数

### clang\_Cursor\_getMangling

CINDEX\_LINKAGE CXString clang\_Cursor\_getMangling (CXCursor)

检索代表光标的mangled名称的 CXString。、

### clang\_Cursor\_getCXXManglings

CINDEX\_LINKAGE CXStringSet \* clang\_Cursor\_getCXXManglings (CXCursor)

检索代表光标处的C++构造函数或析构函数的mangled 符号的CXStrings 。

### clang\_Cursor\_getObjCManglings

CINDEX\_LINKAGE CXStringSet \* clang\_Cursor\_getObjCManglings (CXCursor)

检索代表光标处的ObjC classs接口或实现的mangled 符号的CXStrings 。

# libclang模块:模块自查

此组函数提供了模块的信息的访问。

## Typedefs

### CXModule

typedef void \* CXModule

## 函数

### clang\_Cursor\_getModule

CINDEX\_LINKAGE CXModule clang\_Cursor\_getModule (CXCursor C)

给定一个CXCursor\_ModuleImportDecl光标，返回相关联的模块。

### clang\_getModuleForFile

CINDEX\_LINKAGE CXModule clang\_getModuleForFile (CXTranslationUnit, CXFile)

给定CXFile 头文件，返回包含它的模块，如果存在的话。

### clang\_Module\_getASTFile

CINDEX\_LINKAGE CXFile clang\_Module\_getASTFile (CXModule Module)

### clang\_Module\_getParent

CINDEX\_LINKAGE CXModule clang\_Module\_getParent (CXModule Module)

### clang\_Module\_getName

CINDEX\_LINKAGE CXString clang\_Module\_getName (CXModule Module)

### clang\_Module\_getFullName

CINDEX\_LINKAGE CXString clang\_Module\_getFullName (CXModule Module)

### clang\_Module\_isSystem

CINDEX\_LINKAGE int clang\_Module\_isSystem (CXModule Module)

### clang\_Module\_getNumTopLevelHeaders

CINDEX\_LINKAGE unsigned clang\_Module\_getNumTopLevelHeaders (CXTranslationUnit, CXModule Module)

### clang\_Module\_getTopLevelHeader

CINDEX\_LINKAGE CXFile clang\_Module\_getTopLevelHeader (CXTranslationUnit, CXModule Module, unsigned Index)

# libclang模块:C++ AST自查

此组的例程提供了C++语言特性的AST的访问信息。

## 枚举

### CXNameRefFlags

enum CXNameRefFlags { CXNameRange\_WantQualifier = 0x1, CXNameRange\_WantTemplateArgs = 0x2, CXNameRange\_WantSinglePiece = 0x4 }

## 函数

### clang\_CXXConstructor\_isConvertingConstructor

CINDEX\_LINKAGE unsigned clang\_CXXConstructor\_isConvertingConstructor (CXCursor C)

确定C++构造函数是否为转换构造函数。

### clang\_CXXConstructor\_isCopyConstructor

CINDEX\_LINKAGE unsigned clang\_CXXConstructor\_isCopyConstructor (CXCursor C)

确定C++构造函数是否为复制构造函数。

### clang\_CXXConstructor\_isDefaultConstructor

CINDEX\_LINKAGE unsigned clang\_CXXConstructor\_isDefaultConstructor (CXCursor C)

确定C++构造函数是否为默认构造函数。

### clang\_CXXConstructor\_isMoveConstructor

CINDEX\_LINKAGE unsigned clang\_CXXConstructor\_isMoveConstructor (CXCursor C)

确定C++构造函数是否为移动构造函数。

### clang\_CXXField\_isMutable

CINDEX\_LINKAGE unsigned clang\_CXXField\_isMutable (CXCursor C)

确定C++字段是否声明为“可变”。

### clang\_CXXMethod\_isDefaulted

CINDEX\_LINKAGE unsigned clang\_CXXMethod\_isDefaulted (CXCursor C)

确定C++方法是否声明为“= default”。

### clang\_CXXMethod\_isDeleted

CINDEX\_LINKAGE unsigned clang\_CXXMethod\_isDeleted (CXCursor C)

确定C++方法是否声明为“'= delete”。

### clang\_CXXMethod\_isPureVirtual

CINDEX\_LINKAGE unsigned clang\_CXXMethod\_isPureVirtual (CXCursor C)

确定C++成员函数或成员函数模板是纯虚拟的。

### clang\_CXXMethod\_isStatic

CINDEX\_LINKAGE unsigned clang\_CXXMethod\_isStatic (CXCursor C)

确定C++成员函数或成员函数模板是否声明为“'static'”。

### clang\_CXXMethod\_isVirtual

CINDEX\_LINKAGE unsigned clang\_CXXMethod\_isVirtual (CXCursor C)

确定 C++ 成员函数或成员函数模板是否显式声明为“virtual”，或者它是否重写了其中一个基类中的虚拟方法。

### clang\_CXXMethod\_isCopyAssignmentOperator

CINDEX\_LINKAGE unsigned clang\_CXXMethod\_isCopyAssignmentOperator (CXCursor C)

确定 C++ 成员函数是否为复制赋值运算符，如果是这种情况，则返回 1，否则返回 0。

### clang\_CXXMethod\_isMoveAssignmentOperator

CINDEX\_LINKAGE unsigned clang\_CXXMethod\_isMoveAssignmentOperator (CXCursor C)

确定 C++ 成员函数是否为移动赋值运算符，如果是这种情况，则返回 1，否则返回 0。

### clang\_CXXMethod\_isExplicit

CINDEX\_LINKAGE unsigned clang\_CXXMethod\_isExplicit (CXCursor C)

确定是否显式声明了C++构造函数或转换函数，如果是这种情况，则返回 1，否则返回 0。

### clang\_CXXRecord\_isAbstract

CINDEX\_LINKAGE unsigned clang\_CXXRecord\_isAbstract (CXCursor C)

确定C++记录是否是abstract。

### clang\_EnumDecl\_isScoped

CINDEX\_LINKAGE unsigned clang\_EnumDecl\_isScoped (CXCursor C)

确定枚举声明是否引用作用域内的枚举。

### clang\_CXXMethod\_isConst

CINDEX\_LINKAGE unsigned clang\_CXXMethod\_isConst (CXCursor C)

确定C++成员函数或成员函数模板是否声明为“const”。

### clang\_getTemplateCursorKind

CINDEX\_LINKAGE enum CXCursorKind clang\_getTemplateCursorKind (CXCursor C)

给定表示模板的光标，确定通过实例化模板生成的特化的光标类型。

### clang\_getSpecializedCursorTemplate

CINDEX\_LINKAGE CXCursor clang\_getSpecializedCursorTemplate (CXCursor C)

Given a cursor that may represent a specialization or instantiation of a template, retrieve the cursor that represents the template that it specializes or from which it was instantiated. More...

定一个可能表示模板特化或实例化的光标，检索表示特化或从中实例化的模板的光标。

### clang\_getCursorReferenceNameRange

CINDEX\_LINKAGE CXSourceRange clang\_getCursorReferenceNameRange (CXCursor C, unsigned NameFlags, unsigned PieceIndex)

给定引用其他内容的光标，返回覆盖该引用的源范围。

# libclang模块:tokens提取和操作

此组中的例程提供对翻译单元中tokens的访问，以及这些tokens到其相应光标的语义映射。

## Classes

### CXToken

struct CXToken

描述一个简单的预处理token。

## Typedefs

### CXTokenKind

typedef enum CXTokenKind CXTokenKind

Describes a kind of token. More...

描述token的类别

## 枚举

### CXTokenKind

enum CXTokenKind {

CXToken\_Punctuation, CXToken\_Keyword, CXToken\_Identifier, CXToken\_Literal,

CXToken\_Comment

}

描述token的类别

## 函数

### clang\_getToken

CINDEX\_LINKAGE CXToken \* clang\_getToken (CXTranslationUnit TU, CXSourceLocation Location)

获取从给定位置开始的原始词法token。

### clang\_getTokenKind

CINDEX\_LINKAGE CXTokenKind clang\_getTokenKind (CXToken)

确定给定token的类别。

### clang\_getTokenSpelling

CINDEX\_LINKAGE CXString clang\_getTokenSpelling (CXTranslationUnit, CXToken)

确定给定token的拼写。

### clang\_getTokenLocation

CINDEX\_LINKAGE CXSourceLocation clang\_getTokenLocation (CXTranslationUnit, CXToken)

检索给定token的源位置。

### clang\_getTokenExtent

CINDEX\_LINKAGE CXSourceRange clang\_getTokenExtent (CXTranslationUnit, CXToken)

检索涵盖给定token的源范围。

### clang\_tokenize

CINDEX\_LINKAGE void clang\_tokenize (CXTranslationUnit TU, CXSourceRange Range, CXToken \*\*Tokens, unsigned \*NumTokens)

将给定范围描述的源代码标记为原始词法token。

### clang\_annotateTokens

CINDEX\_LINKAGE void clang\_annotateTokens (CXTranslationUnit TU, CXToken \*Tokens, unsigned NumTokens, CXCursor \*Cursors)

通过为每个token提供光标来批注给定的tokens集，这些tokens可以映射到抽象语法树中的特定实体。

### clang\_disposeTokens

CINDEX\_LINKAGE void clang\_disposeTokens (CXTranslationUnit TU, CXToken \*Tokens, unsigned NumTokens)

释放给定的tokens集合。

# libclang模块:Debugging 工具

这些例程仅用于测试和调试，不应依赖这些例程。

## 函数

### clang\_getCursorKindSpelling

CINDEX\_LINKAGE CXString clang\_getCursorKindSpelling (enum CXCursorKind Kind)

### clang\_getDefinitionSpellingAndExtent

CINDEX\_LINKAGE void clang\_getDefinitionSpellingAndExtent (CXCursor, const char \*\*startBuf, const char \*\*endBuf, unsigned \*startLine, unsigned \*startColumn, unsigned \*endLine, unsigned \*endColumn)

### clang\_enableStackTraces

CINDEX\_LINKAGE void clang\_enableStackTraces (void)

### clang\_executeOnThread

CINDEX\_LINKAGE void clang\_executeOnThread (void(\*fn)(void \*), void \*user\_data, unsigned stack\_size)

# libclang模块:code-completion

代码完成涉及获取（不完整的）源文件，以及了解用户在何处主动编辑该文件，并建议用户可能希望在源代码中的该特定点使用的语法和语义上有效的构造。这些数据结构和例程为代码完成提供支持。

## classes

### CXCompletionResult

struct CXCompletionResult

code completion的一个简单结果

### CXCodeCompleteResults

struct CXCodeCompleteResults

包含 code-completion的结果

### Typedefs

typedef void \* CXCompletionString

一个描述code-completion结果的语义字符串。

## 枚举

### CXCompletionChunkKind

enum CXCompletionChunkKind {

CXCompletionChunk\_Optional, CXCompletionChunk\_TypedText, CXCompletionChunk\_Text, CXCompletionChunk\_Placeholder,

CXCompletionChunk\_Informative, CXCompletionChunk\_CurrentParameter, CXCompletionChunk\_LeftParen, CXCompletionChunk\_RightParen,

CXCompletionChunk\_LeftBracket, CXCompletionChunk\_RightBracket, CXCompletionChunk\_LeftBrace, CXCompletionChunk\_RightBrace,

CXCompletionChunk\_LeftAngle, CXCompletionChunk\_RightAngle, CXCompletionChunk\_Comma, CXCompletionChunk\_ResultType,

CXCompletionChunk\_Colon, CXCompletionChunk\_SemiColon, CXCompletionChunk\_Equal, CXCompletionChunk\_HorizontalSpace,

CXCompletionChunk\_VerticalSpace

}

描述一个 code-completion字符串的单个文本片段。

### CXCodeComplete\_Flags

enum CXCodeComplete\_Flags {

CXCodeComplete\_IncludeMacros = 0x01, CXCodeComplete\_IncludeCodePatterns = 0x02, CXCodeComplete\_IncludeBriefComments = 0x04, CXCodeComplete\_SkipPreamble = 0x08,

CXCodeComplete\_IncludeCompletionsWithFixIts = 0x10

}

可以被传给 clang\_codeCompleteAt()来修改它的行为的Flags。

### CXCompletionContext

enum CXCompletionContext {

CXCompletionContext\_Unexposed = 0, CXCompletionContext\_AnyType = 1 << 0, CXCompletionContext\_AnyValue = 1 << 1, CXCompletionContext\_ObjCObjectValue = 1 << 2,

CXCompletionContext\_ObjCSelectorValue = 1 << 3, CXCompletionContext\_CXXClassTypeValue = 1 << 4, CXCompletionContext\_DotMemberAccess = 1 << 5, CXCompletionContext\_ArrowMemberAccess = 1 << 6,

CXCompletionContext\_ObjCPropertyAccess = 1 << 7, CXCompletionContext\_EnumTag = 1 << 8, CXCompletionContext\_UnionTag = 1 << 9, CXCompletionContext\_StructTag = 1 << 10,

CXCompletionContext\_ClassTag = 1 << 11, CXCompletionContext\_Namespace = 1 << 12, CXCompletionContext\_NestedNameSpecifier = 1 << 13, CXCompletionContext\_ObjCInterface = 1 << 14,

CXCompletionContext\_ObjCProtocol = 1 << 15, CXCompletionContext\_ObjCCategory = 1 << 16, CXCompletionContext\_ObjCInstanceMessage = 1 << 17, CXCompletionContext\_ObjCClassMessage = 1 << 18,

CXCompletionContext\_ObjCSelectorName = 1 << 19, CXCompletionContext\_MacroName = 1 << 20, CXCompletionContext\_NaturalLanguage = 1 << 21, CXCompletionContext\_IncludedFile = 1 << 22,

CXCompletionContext\_Unknown = ((1 << 23) - 1)

}

代表完成已经发生的的context的Bits。

## 函数

### clang\_getCompletionChunkKind

CINDEX\_LINKAGE enum CXCompletionChunkKind clang\_getCompletionChunkKind (CXCompletionString completion\_string, unsigned chunk\_number)

确定一个完成的字符串中的特定chunk的类别。

### clang\_getCompletionChunkText

CINDEX\_LINKAGE CXString clang\_getCompletionChunkText (CXCompletionString completion\_string, unsigned chunk\_number)

检索一个完成的字符串中的特定chunk关联的文本。

### clang\_getCompletionChunkCompletionString

CINDEX\_LINKAGE CXCompletionString clang\_getCompletionChunkCompletionString (CXCompletionString completion\_string, unsigned chunk\_number)

检索一个完成的字符串中的特定chunk关联的完成字符串。

### clang\_getNumCompletionChunks

CINDEX\_LINKAGE unsigned clang\_getNumCompletionChunks (CXCompletionString completion\_string)

检索一个code-completion字符串中的chunks 的数量。

### clang\_getCompletionPriority

CINDEX\_LINKAGE unsigned clang\_getCompletionPriority (CXCompletionString completion\_string)

确定这个code-completion的特性。

### clang\_getCompletionAvailability

CINDEX\_LINKAGE enum CXAvailabilityKind clang\_getCompletionAvailability (CXCompletionString completion\_string)

确定这个code-completion字符串的指向的实体的可用性。

### clang\_getCompletionNumAnnotations

CINDEX\_LINKAGE unsigned clang\_getCompletionNumAnnotations (CXCompletionString completion\_string)

确定这个code-completion字符串关联的注释的数量。

### clang\_getCompletionAnnotation

CINDEX\_LINKAGE CXString clang\_getCompletionAnnotation (CXCompletionString completion\_string, unsigned annotation\_number)

检索这个code-completion字符串关联的注释。

### clang\_getCompletionParent

CINDEX\_LINKAGE CXString clang\_getCompletionParent (CXCompletionString completion\_string, enum CXCursorKind \*kind)

检索这个 completion字符串的父 context。

### clang\_getCompletionBriefComment

CINDEX\_LINKAGE CXString clang\_getCompletionBriefComment (CXCompletionString completion\_string)

检索主文档注释，该注释绑定到给定的code-completion字符串对应的声明。

### clang\_getCursorCompletionString

CINDEX\_LINKAGE CXCompletionString clang\_getCursorCompletionString (CXCursor cursor)

检索任意声明或宏定义光标的completion 字符串。

### clang\_getCompletionNumFixIts

CINDEX\_LINKAGE unsigned clang\_getCompletionNumFixIts (CXCodeCompleteResults \*results, unsigned completion\_index)

检索给定完成索引的 fix-its的数量。

### clang\_getCompletionFixIt

CINDEX\_LINKAGE CXString clang\_getCompletionFixIt (CXCodeCompleteResults \*results, unsigned completion\_index, unsigned fixit\_index, CXSourceRange \*replacement\_range)

在为相应的完成插入文本之前必须应用的Fix-its。

### clang\_defaultCodeCompleteOptions

CINDEX\_LINKAGE unsigned clang\_defaultCodeCompleteOptions (void)

返回 code-completion 选项的默认集合，这些选项可以被传递到 toclang\_codeCompleteAt（） 。

### clang\_codeCompleteAt

CINDEX\_LINKAGE CXCodeCompleteResults \* clang\_codeCompleteAt (CXTranslationUnit TU, const char \*complete\_filename, unsigned complete\_line, unsigned complete\_column, struct CXUnsavedFile \*unsaved\_files, unsigned num\_unsaved\_files, unsigned options)

在翻译单元中的给定位置执行代码完成。

### clang\_sortCodeCompletionResults

CINDEX\_LINKAGE void clang\_sortCodeCompletionResults (CXCompletionResult \*Results, unsigned NumResults)

按不区分大小写的字母顺序对代码完成结果进行排序。

### clang\_disposeCodeCompleteResults

CINDEX\_LINKAGE void clang\_disposeCodeCompleteResults (CXCodeCompleteResults \*Results)

释放给定的代码完成结果集。

### clang\_codeCompleteGetNumDiagnostics

CINDEX\_LINKAGE unsigned clang\_codeCompleteGetNumDiagnostics (CXCodeCompleteResults \*Results)

确定在执行代码完成的位置之前生成的诊断程序的数量。

### clang\_codeCompleteGetDiagnostic

CINDEX\_LINKAGE CXDiagnostic clang\_codeCompleteGetDiagnostic (CXCodeCompleteResults \*Results, unsigned Index)

检索与给定代码完成关联的诊断。

### clang\_codeCompleteGetContexts

CINDEX\_LINKAGE unsigned long long clang\_codeCompleteGetContexts (CXCodeCompleteResults \*Results)

为给定代码完成的上下文，确定哪些完成是合适的。

### clang\_codeCompleteGetContainerKind

CINDEX\_LINKAGE enum CXCursorKind clang\_codeCompleteGetContainerKind (CXCodeCompleteResults \*Results, unsigned \*IsIncomplete)

返回当前代码完成上下文的容器的光标类型。

### clang\_codeCompleteGetContainerUSR

CINDEX\_LINKAGE CXString clang\_codeCompleteGetContainerUSR (CXCodeCompleteResults \*Results)

返回当前代码完成上下文的容器的 USR。

### clang\_codeCompleteGetObjCSelector

CINDEX\_LINKAGE CXString clang\_codeCompleteGetObjCSelector (CXCodeCompleteResults \*Results)

返回用于Objective-C 消息发送的当前输入的选择器，格式类似于“initWithFoo：bar：。

# libclang模块:杂项实用函数

## Typedefs

### CXInclusionVisitor

typedef void(\* CXInclusionVisitor) (CXFile included\_file, CXSourceLocation \*inclusion\_stack, unsigned include\_len, CXClientData client\_data)

Visitor invoked for each file in a translation unit (used with clang\_getInclusions()). More...

为翻译单元中的每个文件调用的访问者（与 clang\_getInclusions（） 一起使用）。

### CXEvalResult

typedef void \* CXEvalResult

光标的计算结果。

## 枚举

### CXEvalResultKind

enum CXEvalResultKind {

CXEval\_Int = 1, CXEval\_Float = 2, CXEval\_ObjCStrLiteral = 3, CXEval\_StrLiteral = 4,

CXEval\_CFStr = 5, CXEval\_Other = 6, CXEval\_UnExposed = 0

}

## 函数

### clang\_getClangVersion

CINDEX\_LINKAGE CXString clang\_getClangVersion (void)

返回适合向用户显示但不打算解析的版本字符串（不保证格式稳定）。

### clang\_toggleCrashRecovery

CINDEX\_LINKAGE void clang\_toggleCrashRecovery (unsigned isEnabled)

启用/禁用崩溃恢复。

### clang\_getInclusions

CINDEX\_LINKAGE void clang\_getInclusions (CXTranslationUnit tu, CXInclusionVisitor visitor, CXClientData client\_data)

访问翻译单元中的预处理器包含集。

### clang\_Cursor\_Evaluate

CINDEX\_LINKAGE CXEvalResult clang\_Cursor\_Evaluate (CXCursor C)

如果光标是语句声明，则尝试计算语句;如果是变量，则尝试计算其初始值设定项;将其计算为相应的类型。

### clang\_EvalResult\_getKind

CINDEX\_LINKAGE CXEvalResultKind clang\_EvalResult\_getKind (CXEvalResult E)

返回计算结果的类别。

### clang\_EvalResult\_getAsInt

CINDEX\_LINKAGE int clang\_EvalResult\_getAsInt (CXEvalResult E)

如果计算结果的类型为 Int，则以整数形式返回。

### clang\_EvalResult\_getAsLongLong

CINDEX\_LINKAGE long long clang\_EvalResult\_getAsLongLong (CXEvalResult E)

如果计算结果的类型为 Int，则以长整型形式返回计算结果。

### clang\_EvalResult\_isUnsignedInt

CINDEX\_LINKAGE unsigned clang\_EvalResult\_isUnsignedInt (CXEvalResult E)

如果类型为 Int 并且计算结果生成无符号整数，则返回非零值。

### clang\_EvalResult\_getAsUnsigned

CINDEX\_LINKAGE unsigned long long clang\_EvalResult\_getAsUnsigned (CXEvalResult E)

如果计算结果的类型为 Int 且 clang\_EvalResult\_isUnsignedInt 不为零，则以无符号整数的形式返回计算结果。

### clang\_EvalResult\_getAsDouble

CINDEX\_LINKAGE double clang\_EvalResult\_getAsDouble (CXEvalResult E)

如果类型为双精度，则返回双精度计算结果。

### clang\_EvalResult\_getAsStr

const CINDEX\_LINKAGE char \* clang\_EvalResult\_getAsStr (CXEvalResult E)

如果类型不是 Int 或浮点数，则以常量字符串的形式返回计算结果。

### clang\_EvalResult\_dispose

CINDEX\_LINKAGE void clang\_EvalResult\_dispose (CXEvalResult E)

释放创建的计算内存。

# libclang模块:Remapping 函数

## Typedefs

### CXRemapping

typedef void \* CXRemapping

原始源文件及其翻译文件的重新映射。

## 函数

### clang\_getRemappings

CINDEX\_LINKAGE CXRemapping clang\_getRemappings (const char \*path)

检索重新映射。

### clang\_getRemappingsFromFileList

CINDEX\_LINKAGE CXRemapping clang\_getRemappingsFromFileList (const char \*\*filePaths, unsigned numFiles)

检索重新映射。

### clang\_remap\_getNumFiles

CINDEX\_LINKAGE unsigned clang\_remap\_getNumFiles (CXRemapping)

确定重新映射的次数。

### clang\_remap\_getFilenames

CINDEX\_LINKAGE void clang\_remap\_getFilenames (CXRemapping, unsigned index, CXString \*original, CXString \*transformed)

从重新映射中获取原始文件名和关联的文件名。

### clang\_remap\_dispose

CINDEX\_LINKAGE void clang\_remap\_dispose (CXRemapping)

释放重新映射。

# libclang模块:Higher level API 函数

## Classes

### CXCursorAndRangeVisitor

struct CXCursorAndRangeVisitor

### CXIdxLoc

struct CXIdxLoc

传递给索引回调的源位置。

### CXIdxIncludedFileInfo

struct CXIdxIncludedFileInfo

ppIncludeFile 回调使用的数据。

### CXIdxImportedASTFileInfo

struct CXIdxImportedASTFileInfo

IndexerCallbacks::importedASTFile使用的数据。

### CXIdxAttrInfo

struct CXIdxAttrInfo

### CXIdxEntityInfo

struct CXIdxEntityInfo

### CXIdxContainerInfo

struct CXIdxContainerInfo

### CXIdxIBOutletCollectionAttrInfo

struct CXIdxIBOutletCollectionAttrInfo

### CXIdxDeclInfo

struct CXIdxDeclInfo

### CXIdxObjCContainerDeclInfo

struct CXIdxObjCContainerDeclInfo

### CXIdxBaseClassInfo

struct CXIdxBaseClassInfo

### CXIdxObjCProtocolRefInfo

struct CXIdxObjCProtocolRefInfo

### CXIdxObjCProtocolRefListInfo

struct CXIdxObjCProtocolRefListInfo

### CXIdxObjCInterfaceDeclInfo

struct CXIdxObjCInterfaceDeclInfo

### CXIdxObjCCategoryDeclInfo

struct CXIdxObjCCategoryDeclInfo

### CXIdxObjCPropertyDeclInfo

struct CXIdxObjCPropertyDeclInfo

### CXIdxCXXClassDeclInfo

struct CXIdxCXXClassDeclInfo

### CXIdxEntityRefInfo

struct CXIdxEntityRefInfo

IndexerCallbacks::indexEntityReference使用的数据。

### IndexerCallbacks

struct IndexerCallbacks

由clang\_indexSourceFile 和 clang\_indexTranslationUnit 使用的一组回调。

## Typedefs

### CXCursorAndRangeVisitor

typedef struct CXCursorAndRangeVisitor CXCursorAndRangeVisitor

### CXIdxClientFile

typedef void \* CXIdxClientFile

关联到CXFile的client数据对象。

### CXIdxClientEntity

typedef void \* CXIdxClientEntity

关联到一个语义实体的client数据对象。

### CXIdxClientContainer

typedef void \* CXIdxClientContainer

关联到一个实体的语义container的client数据对象。

### CXIdxClientASTFile

typedef void \* CXIdxClientASTFile

关联到一个AST文件 (PCH or module)的client数据对象。

### CXIndexAction

typedef void \* CXIndexAction

被用于一个或多个翻译单元的indexing action/session。

### CXFieldVisitor

typedef enum CXVisitorResult(\* CXFieldVisitor) (CXCursor C, CXClientData client\_data)

为由遍历建立的每个field唤醒的Visitor 。

## 枚举

### CXVisitorResult

enum CXVisitorResult { CXVisit\_Break, CXVisit\_Continue }

### CXResult

enum CXResult { CXResult\_Success = 0, CXResult\_Invalid = 1, CXResult\_VisitBreak = 2 }

### CXIdxEntityKind

enum CXIdxEntityKind {

CXIdxEntity\_Unexposed = 0, CXIdxEntity\_Typedef = 1, CXIdxEntity\_Function = 2, CXIdxEntity\_Variable = 3,

CXIdxEntity\_Field = 4, CXIdxEntity\_EnumConstant = 5, CXIdxEntity\_ObjCClass = 6, CXIdxEntity\_ObjCProtocol = 7,

CXIdxEntity\_ObjCCategory = 8, CXIdxEntity\_ObjCInstanceMethod = 9, CXIdxEntity\_ObjCClassMethod = 10, CXIdxEntity\_ObjCProperty = 11,

CXIdxEntity\_ObjCIvar = 12, CXIdxEntity\_Enum = 13, CXIdxEntity\_Struct = 14, CXIdxEntity\_Union = 15,

CXIdxEntity\_CXXClass = 16, CXIdxEntity\_CXXNamespace = 17, CXIdxEntity\_CXXNamespaceAlias = 18, CXIdxEntity\_CXXStaticVariable = 19,

CXIdxEntity\_CXXStaticMethod = 20, CXIdxEntity\_CXXInstanceMethod = 21, CXIdxEntity\_CXXConstructor = 22, CXIdxEntity\_CXXDestructor = 23,

CXIdxEntity\_CXXConversionFunction = 24, CXIdxEntity\_CXXTypeAlias = 25, CXIdxEntity\_CXXInterface = 26, CXIdxEntity\_CXXConcept = 27

}

### CXIdxEntityLanguage

enum CXIdxEntityLanguage {

CXIdxEntityLang\_None = 0, CXIdxEntityLang\_C = 1, CXIdxEntityLang\_ObjC = 2, CXIdxEntityLang\_CXX = 3,

CXIdxEntityLang\_Swift = 4

}

### CXIdxEntityCXXTemplateKind

enum CXIdxEntityCXXTemplateKind { CXIdxEntity\_NonTemplate = 0, CXIdxEntity\_Template = 1, CXIdxEntity\_TemplatePartialSpecialization = 2, CXIdxEntity\_TemplateSpecialization = 3 }

Extra C++ template information for an entity. More...

### CXIdxAttrKind

enum CXIdxAttrKind { CXIdxAttr\_Unexposed = 0, CXIdxAttr\_IBAction = 1, CXIdxAttr\_IBOutlet = 2, CXIdxAttr\_IBOutletCollection = 3 }

### CXIdxDeclInfoFlags

enum CXIdxDeclInfoFlags { CXIdxDeclFlag\_Skipped = 0x1 }

### CXIdxObjCContainerKind

enum CXIdxObjCContainerKind { CXIdxObjCContainer\_ForwardRef = 0, CXIdxObjCContainer\_Interface = 1, CXIdxObjCContainer\_Implementation = 2 }

### CXIdxEntityRefKind

enum CXIdxEntityRefKind { CXIdxEntityRef\_Direct = 1, CXIdxEntityRef\_Implicit = 2 }

IndexerCallbacks::indexEntityReference使用的数据。

### CXSymbolRole

enum CXSymbolRole {

CXSymbolRole\_None = 0, CXSymbolRole\_Declaration = 1 << 0, CXSymbolRole\_Definition = 1 << 1, CXSymbolRole\_Reference = 1 << 2,

CXSymbolRole\_Read = 1 << 3, CXSymbolRole\_Write = 1 << 4, CXSymbolRole\_Call = 1 << 5, CXSymbolRole\_Dynamic = 1 << 6,

CXSymbolRole\_AddressOf = 1 << 7, CXSymbolRole\_Implicit = 1 << 8

}

符号产生时，该符号对应的角色。

### CXIndexOptFlags

enum CXIndexOptFlags {

CXIndexOpt\_None = 0x0, CXIndexOpt\_SuppressRedundantRefs = 0x1, CXIndexOpt\_IndexFunctionLocalSymbols = 0x2, CXIndexOpt\_IndexImplicitTemplateInstantiations = 0x4,

CXIndexOpt\_SuppressWarnings = 0x8, CXIndexOpt\_SkipParsedBodiesInSession = 0x10

}

## 函数

### clang\_findReferencesInFile

CINDEX\_LINKAGE CXResult clang\_findReferencesInFile (CXCursor cursor, CXFile file, CXCursorAndRangeVisitor visitor)

在特定文件中查找声明的引用。

### clang\_findIncludesInFile

CINDEX\_LINKAGE CXResult clang\_findIncludesInFile (CXTranslationUnit TU, CXFile file, CXCursorAndRangeVisitor visitor)

在特定文件中查找 #import/include 指令。

### clang\_index\_isEntityObjCContainerKind

CINDEX\_LINKAGE int clang\_index\_isEntityObjCContainerKind (CXIdxEntityKind)

### clang\_index\_getObjCContainerDeclInfo

const CINDEX\_LINKAGE CXIdxObjCContainerDeclInfo \* clang\_index\_getObjCContainerDeclInfo (const CXIdxDeclInfo \*)

### clang\_index\_getObjCInterfaceDeclInfo

const CINDEX\_LINKAGE CXIdxObjCInterfaceDeclInfo \* clang\_index\_getObjCInterfaceDeclInfo (const CXIdxDeclInfo \*)

### clang\_index\_getObjCCategoryDeclInfo

const CINDEX\_LINKAGE CXIdxObjCCategoryDeclInfo \* clang\_index\_getObjCCategoryDeclInfo (const CXIdxDeclInfo \*)

### clang\_index\_getObjCProtocolRefListInfo

const CINDEX\_LINKAGE CXIdxObjCProtocolRefListInfo \* clang\_index\_getObjCProtocolRefListInfo (const CXIdxDeclInfo \*)

### clang\_index\_getObjCPropertyDeclInfo

const CINDEX\_LINKAGE CXIdxObjCPropertyDeclInfo \* clang\_index\_getObjCPropertyDeclInfo (const CXIdxDeclInfo \*)

### clang\_index\_getIBOutletCollectionAttrInfo

const CINDEX\_LINKAGE CXIdxIBOutletCollectionAttrInfo \* clang\_index\_getIBOutletCollectionAttrInfo (const CXIdxAttrInfo \*)

### clang\_index\_getCXXClassDeclInfo

const CINDEX\_LINKAGE CXIdxCXXClassDeclInfo \* clang\_index\_getCXXClassDeclInfo (const CXIdxDeclInfo \*)

### clang\_index\_getClientContainer

CINDEX\_LINKAGE CXIdxClientContainer clang\_index\_getClientContainer (const CXIdxContainerInfo \*)

用于检索绑定到容器的自定义 CXIdxClientContainer。

### clang\_index\_setClientContainer

CINDEX\_LINKAGE void clang\_index\_setClientContainer (const CXIdxContainerInfo \*, CXIdxClientContainer)

用于设置绑定到容器的自定义 CXIdxClientContainer。

### clang\_index\_getClientEntity

CINDEX\_LINKAGE CXIdxClientEntity clang\_index\_getClientEntity (const CXIdxEntityInfo \*)

用于检索绑定到实体的自定义 CXIdxClientEntity 。

### clang\_index\_setClientEntity

CINDEX\_LINKAGE void clang\_index\_setClientEntity (const CXIdxEntityInfo \*, CXIdxClientEntity)

用于设置绑定到实体的自定义 CXIdxClientEntity 。

### clang\_IndexAction\_create

CINDEX\_LINKAGE CXIndexAction clang\_IndexAction\_create (CXIndex CIdx)

indexing action/session，应用于一个或多个翻译单元。

### clang\_IndexAction\_dispose

CINDEX\_LINKAGE void clang\_IndexAction\_dispose (CXIndexAction)

销毁给定的索引操作。

### clang\_indexSourceFile

CINDEX\_LINKAGE int clang\_indexSourceFile (CXIndexAction, CXClientData client\_data, IndexerCallbacks \*index\_callbacks, unsigned index\_callbacks\_size, unsigned index\_options, const char \*source\_filename, const char \*const \*command\_line\_args, int num\_command\_line\_args, struct CXUnsavedFile \*unsaved\_files, unsigned num\_unsaved\_files, CXTranslationUnit \*out\_TU, unsigned TU\_options)

通过 IndexerCallback 实现的回调，为给定源文件和与该文件对应的翻译单元编制索引到特定文件。

### clang\_indexSourceFileFullArgv

CINDEX\_LINKAGE int clang\_indexSourceFileFullArgv (CXIndexAction, CXClientData client\_data, IndexerCallbacks \*index\_callbacks, unsigned index\_callbacks\_size, unsigned index\_options, const char \*source\_filename, const char \*const \*command\_line\_args, int num\_command\_line\_args, struct CXUnsavedFile \*unsaved\_files, unsigned num\_unsaved\_files, CXTranslationUnit \*out\_TU, unsigned TU\_options)

与clang\_indexSourceFile相同，但需要完整的命令行，command\_line\_args包括 argv[0]。

### clang\_indexTranslationUnit

CINDEX\_LINKAGE int clang\_indexTranslationUnit (CXIndexAction, CXClientData client\_data, IndexerCallbacks \*index\_callbacks, unsigned index\_callbacks\_size, unsigned index\_options, CXTranslationUnit)

通过 IndexerCallback 实现的回调。为给定的翻译单元编制索引。

### clang\_indexLoc\_getFileLocation

CINDEX\_LINKAGE void clang\_indexLoc\_getFileLocation (CXIdxLoc loc, CXIdxClientFile \*indexFile, CXFile \*file, unsigned \*line, unsigned \*column, unsigned \*offset)

检索由给定 CXIdxLoc 表示的 CXIdxFile、文件、行、列和偏移量。

### clang\_indexLoc\_getCXSourceLocation

CINDEX\_LINKAGE CXSourceLocation clang\_indexLoc\_getCXSourceLocation (CXIdxLoc loc)

检索由给定的 CXIdxLoc 表示的 CXSourceLocation。

### clang\_Type\_visitFields

CINDEX\_LINKAGE unsigned clang\_Type\_visitFields (CXType T, CXFieldVisitor visitor, CXClientData client\_data)

访问特定类型的字段。

# 附录