			N.W.
00:00			14:01
讲述: 冯永吉 大小: 12.8 你好,我是戴铭。今天,我和你分 Clang 使用的是模块化设计,可以 范检查、IDE 中的语法高亮、语法	分享的主题是,如何缩 从将自身功能以库的方	方式来供上层应用来调用	
Clang 库对接上层应用有三个接口个接口库的介绍,我已经在第8篇 其中,LibClang 为了兼容更多 C LibTooling 具备 Clang 的全量能	篇文章中和你详细分别 lang 版本,相比 Cla 力。Clang 插件编写	享过。 ang 少了很多功能;Cla i代码的方式,和 LibTo	ang 插件和 poling 几乎一样,
不同的是 Clang 插件还能够控制组外,编写好的 LibTooling 还能够外,编写好的 LibTooling 还能够所以说,Clang 插件在功能上是最Clang 插件。	非常方便地转成 Cla	ng 插件。	
Clang 插件代码编写后进行编译的 CMake 工具,来解决跨平台编译 我们可以先通过 CMakeList.txt 又 生成目标平台所需的编译文件。这	规范问题。 文件,来定制 CMake	e 编译流程,再根据 Cl	MakeList.txt 文件
平台就是 Visual Studio 工程,m Xcode 来编译 Clang。 接下来,我就和你说说怎么拉 Cla	acOS 里还可以生成 ang 的代码,以及编	Xcode 工程。所以,何	你可以使用熟悉的
在 macOS 平台上编译 Clar 接下来的内容,我会以 macOS 平 <u>方说明</u> 。 首先,从 GitHub 上拉下 Clang f	平台编译 Clang 为例	。如果你想在其他平台	编译,可以参看 <mark>直</mark>
1 git clone https://github.o	com/llvm/llvm-proje	ct.git	■ 复制代码
然后,执行以下命令,来创建构建 1 cd llvm-project 2 mkdir build (in-tree build 3 cd build			틜 复制代码
4 目录结构如下图: ■ Ilvm-project		b uild	
		clang clang-tools-extra compiler-rt debuginfo-tests libclc libcxx	
		libcxxabi libunwind lld lldb llgo	
		openmp parallel-libs polly pstl README.md	
其中,clang 目录就是类 C 语言统源码进行平台无关优化的优化器代目录里是调试器的代码;lld 里是特	说码,另一部分是生 _风 链接器代码。	战平台相关汇编代码的 <u>给</u>	E成器代码;Ildb
来编译。生成 Makefile 文件,你 1 cmake -DLLVM_ENABLE_PROJE(2 make 3		Makefiles"/llvm	■ 复制代码
生成 Xcode 工程,你可以使用这	个命令:		몔 复制代码
1 cmake -G Xcode -DLLVM_ENAI 2 执行完后,会在 build 目录下生成			
Ilvm-project ▶	build clang clang-tools-extra compiler-rt debuginfo-tests libclc libcxx libcxxabi libunwind	benchmarks cmake cmake_install.cmake CMakeCache.txt CMakeFiles CMakeScripts CPackConfig.cmake CPackSourceConfig.cn Debug DummyConfigureOutpu	
	IId IIdb IIgo IIvm openmp parallel-libs polly pstl README.md	DummyConfigureOutpo	
执行 cmake 命令时,你可能会遇 1 The C compiler identif	ication is unknown		围 复制代码
2 The CXX compiler ident: 3 CMake Error at CMakeLists 4 No CMAKE_C_COMPILER cou 5 6 CMake Error at CMakeLists 7 No CMAKE_CXX_COMPILER co	ification is unknow .txt:39 (project): ld be foundtxt:39 (project):		
这表示 cmake 没有找到代码编译 一是,如果你没有安装 Xcode			命令安装:
1 xcode-selectinstall 2 二是,如果你已经安装了 Xcod	le Commo:	ools 的话 专址	■复制代码 即可,命令如下:
二是,如果你已经安装了 Xcod 1 sudo xcode-selectreset 2	Juniandline To	, 旦接 reset	即可,命令如下:
生成 Xcode 工程后,打开生成的 Schemes。编译完后生成的库文的 有了可以编写编译插件的 Xcode	牛,就在 Ilvm-proje	ct/build/Debug/lib/	目录下。
准备编写 Clang 插件 编写之前,先在 llvm-project/cla YourPlugin.cpp 文件和 CMakeL 文件来指导编译,cpp 是源文件。	ists.txt 文件。其中,		
接下来,我们可以使用如下代码编 1 add_llvm_library(YourPlug: 2			鲁 复制代码
这段代码是指,要将 Clang 插件的写调试。 写调试。 想要更多地了解 CMake 的语法和			
件后,再次用 cmake 命令生成 X 到这里,我们已经准备好了 Clang 发 Clang 插件了。	code 工程,里面就	能够集成 YourPlugin.d	pp 文件。
编写 Clang 插件代码,入口就是 么? FrontAction 是什么? FrontActions 是编写 Clang 插件			
基类。其实,FrontActions 并没 ⁻ 了一个入口和工作环境。 通过这个接口,你可以编写你要在	干什么实际的事情,	只是为接下来基于 AST	·操作的函数提供 强过
,		,	
ASTFrontendAction 在 AST 上自Consumer,以获取 AST 上的 AS		,	数
ASTFrontendAction 在 AST 上自 Consumer,以获取 AST 上的 AS	n : public clang::A r 方法 <clang::astconsumer ce &Compiler, llvm: 元 r<clang::astconsume sConsumer);</clang::astconsume </clang::astconsumer 	STFrontendAction { > CreateASTConsumer(:StringRef InFile) { r>(■复制代码
ASTFrontendAction 在 AST 上自Consumer, 以获取 AST 上的 ASCHORGEN CONSUMER AST LODE ASSET CONSUME ASTER AST CONSUME AST CONSUME AST COMPILE INSTANCE AST CONSUMER AST COMPILE INSTANCE AST CONSUMER AST COMPILE INSTANCE AST COMPIL	n: public clang::A r 方法 clang::ASTConsumer ce &Compiler, llvm: 元 r <clang::astconsume detranslationunit()="" sconsumer);="" th="" translationunit()="" 函<="" 数会在接收每个节点="" 数是在访问到全局变=""><th>eateASTConsumer 区 STFrontendAction { > CreateASTConsumer(:StringRef InFile) { r>(AST 的抽象基类,可以 两个函数,以接收访问 量、函数定义这样最上 访问时的回调。 a数,使用 ASTContex</th><th>■复制代码 ■复制代码 AST 时的回调。 层声明时进行回</th></clang::astconsume>	eateASTConsumer 区 STFrontendAction { > CreateASTConsumer(:StringRef InFile) { r>(AST 的抽象基类,可以 两个函数,以接收访问 量、函数定义这样最上 访问时的回调。 a数,使用 ASTContex	■复制代码 ■复制代码 AST 时的回调。 层声明时进行回
ASTFrontendAction 在 AST 上自Consumer,以获取 AST 上的 ASCACTION ASTAIN ASTAIN ASTAIN ASTAIN ASTAIN ASTAIN ASTAIN ASTCONSUMER 可以提供很多入口 ASTCONSUMER TO THE ASTON A	n: public clang::A r 方法 clang::ASTConsumer ce &Compiler, llvm: 元 r <clang::astconsumer clang::slationunit(c<="" decl="" mer:="" public="" sconsumer);="" td="" translationunit()="" 单元。具体代码="" 如是在访问到全局变数会在接收每个节点="" 数是在该问到全局变数会在接收每个节点=""><td>eateASTConsumer 函 STFrontendAction { > CreateASTConsumer(:StringRef InFile) { r>(AST 的抽象基类,可以 两个函数,以接收请量 证为问时的回调。 a数,使用 ASTContext additional actions and actions actions actions action actions actions</td><td>重载 AST 时的时间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间</td></clang::astconsumer>	eateASTConsumer 函 STFrontendAction { > CreateASTConsumer(:StringRef InFile) { r>(AST 的抽象基类,可以 两个函数,以接收请量 证为问时的回调。 a数,使用 ASTContext additional actions and actions actions actions action actions	重载 AST 时的时间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间
ASTFrontendAction 在 AST 上自Consumer,以获取 AST 上的 ASC 代码示例如下所示: 1 class FindNamedClassAction public: 3 // 实现 CreateASTConsume virtual std::unique_ptroclang::CompilerInstance // 返回 ASTConsumer 单: return std::unique_ptroclang	TranslationUnit() 函decl 单元。 具体代码 decl 单元。 具体代码 decl 单元 context.getTranslation to nym	eateASTConsumer 函 STFrontendAction { > CreateASTConsumer(:StringRef InFile) { r>(AST 的抽象基类,可以 两个函数,以接收请量 证为问时的回调。 a数,使用 ASTContext additional actions and actions actions actions action actions	重载 AST 时的时间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间间
ASTFrontendAction 在 AST 上自Consumer,以获取 AST 上的 ASCCONSUMER,以获取 AST 上的 ASCCONSUMER AST LENDAGE ASTCONSUMER ASTCONSU	TranslationUnit() 数会在接收每个节点。 TranslationUnit() 数会在接收每个节点。 TranslationUnit() 数合性。 如果在接收每个节点。 TranslationUnit() 数合性。 如果在接收每个节点。 TranslationUnit() 包含性。 如果在接收每个节点。	eateASTConsumer 函STFrontendAction { > CreateASTConsumer(:StringRef InFile) { r>(AST 的抽象基类,可问 动数,定义这样最上 访问时的回调。 数数,使用 ASTContext 3如下: :ASTConsumer { ASTContext &Context) 元。会访问所有 AST 里的 ionUnitDecl()); onUnitDecl 函数通过重 and	重载 AST 时的进行 IIII IIII IIII IIII IIII IIII IIII I
ASTFrontendAction 在 AST 上自Consumer,以获取 AST 上的 ASC 代码示例如下所示: 1 class FindNamedClassAction	TCOnsumer 单元。 In: public clang::A In: public cla	eateASTConsumer 函 STFrontendAction {	重和 AST 明明 记录 as a sea of the AST Visitor 明明 证别 是制代码 lea of the AST Visitor 明明 lea of the AST Visitor 的正式 是不是 as a sea of the AST Visitor 的正式 是 as a sea of the AST Visitor 的正式 是 as a sea of the AST Visitor 的正式 是 as a sea of the AST Visitor 的正式 是 as a sea of the AST Visitor 的正式 是 as a sea of the AST Visitor 的正式 是 as a sea of the AST Visitor 的正式 是 as a sea of the AST Visitor 的正式 是 as a sea of the AST Visitor 的正式 是 as a sea of the AST Visitor 的正式 是 as a sea of the AST Visitor 的正式 是 as a sea of the AST Visitor of the AST Visitor 的正式 是 as a sea of the AST Visitor of the AST Visitor 的正式 是 as a sea of the AST Visitor of
ASTFrontendAction 在 AST 上自Consumer,以获取 AST 上的 ASCONSUMER,以获取 AST 上的 ASCONSUMER,以实现 CreateASTConsumer virtual std::unique_ptrescaled class::compilerInstanter of clang::CompilerInstanter of clang::CompilerInstanter of clang::CompilerInstanter of clang::CompilerInstanter of clang::CompilerInstanter of clang::CompilerInstanter of class o	Translation Unit (clang:: stor xiaget Translation Wisitor xiaget Translation Yisitor; sicontext.get Translation is it or it is it or; sicontext.get Translation is it or it is it or; sicontext.get Translation is it or it i	eateASTConsumer 函 STFrontendAction { > CreateASTConsumer(:StringRef InFile) { 如	重 AST 明明 all the AST of the AST
ASTFrontendAction 在 AST 上自Consumer,以获取 AST 上的 ASC 代码示例如下所示: 1 class FindNamedClassAction 2 public: 3 // 实现 CreateASTConsume 4 virtual std::unique_ptr- 5 clang::CompilerInstance 6 // 返回 ASTConsumer 单 7 return std::unique_pt 8 new FindNamedClass 9 } 10 }; 11 ASTConsumer 可以提供很多入口 HandleTopLevelDecl() 和 Handle 其中,HandleTopLevelDecl() 函 in,HandleTranslationUnit() 函 in 下面有一个示例,会重载 Handle 其中,HandleTopLevelDecl() 函 in,HandleTranslationUnit() 函 in Formation 1 class FindNamedClassConsument 2 public: 3 virtual void HandleTransel 4 // 通过 RecursiveASTVisitor virtual void HandleTransel 5 visitor.TraverseDecl() 6 } 7 private: 8 // 一个 RecursiveASTVisitor virtual void HandleTransel 5 yisitor.TraverseDecl() 6 } 7 private: 8 // 一个 RecursiveASTVisitor virtual void HandleTransel 7 indNamedClassVisitor virtual void HandleTransel 7 indNamedClassVisitor virtual void HandleTransel 7 indNamedClassVisitor virtual void HandleTransel 8 indNamedClassVisitor virtual void HandleTransel 9 indNamedClassVisitor virtual virtual void HandleTransel 9 indNamedClassVisitor virtual virtual virtual virtual virtual vir	STConsumer 单元。 n: public clang::A r 方法 clang::ASTConsumer ce &Compiler, llvm: 元 r <clang::astconsumer 1,="" petranslationunit()="" sconsumer);="" slatio<="" slationunit(clang::="" slationwait(clang::="" td="" translationunit()="" translationunit(clang::="" 如会在接收="" 是可以以证证的=""><td>EateASTConsumer 函STFrontendAction { STFrontendAction { CreateASTConsumer(StringRef InFile) {</td><td>電型 AST 时时时 调 la la</td></clang::astconsumer>	EateASTConsumer 函STFrontendAction { STFrontendAction { CreateASTConsumer(StringRef InFile) {	電型 AST 时时时 调 la
ASTFrontendAction 在 AST 上自Consumer,以获取 AST 上的 ASC 代码示例如下所示: 1 class FindNamedClassAction public: 3 // 实现 CreateASTConsume virtual std::unique_ptr-clang::CompilerInstame // 返回 ASTConsumer 单 return std::unique_ptr-seturn std::unique_ptr-new FindNamedClass	STConsumer 单元。 n: public clang::A r/sk <clang::astconsumer &compiler,="" 1,="" ast="" ce="" llvm:="" sconsumer);="" td="" translationunit(clang::slationunit(clang::slationunit(clang::slationunit(clang::slation));="" 元<clang::astconsumer="" 动动动动动动动动动动动动动动动动动动动动动动动动动动动动动动动动动动动<="" 数会在接收每个可以的一个可以是在接收每个可以的一个可以是在接收有效。="" 是enslationunit()="" 法方向="" 点向="" 节点做好活=""><td>EateASTConsumer 函 STFrontendAction { > CreateASTConsumer(:StringRef InFile) { AST 的函数基类收详最上的函数的回调。 indunity 使用 ASTContext & Context) (</td><td>電AST 明明 是 電AST 明明 记 明明 元 明复 是 をAST Visitor 的正 明明 一 元 明明 一 元 一 元 一 元 一 元 一 元 一 元 一 元 一</td></clang::astconsumer>	EateASTConsumer 函 STFrontendAction { > CreateASTConsumer(:StringRef InFile) { AST 的函数基类收详最上的函数的回调。 indunity 使用 ASTContext & Context) (電AST 明明 是 電AST 明明 记 明明 元 明复 是 をAST Visitor 的正 明明 一 元 明明 一 元 一 元 一 元 一 元 一 元 一 元 一 元 一
ASTFrontendAction 在 AST 上自Consumer, 以获取 AST 上的 AST Consumer, 以获取 AST 上的 AST 代码示例如下所示: 1 class FindNamedClassAction public: 3 // 实现 CreateASTConsume virtual std::unique_ptr-clang::CompilerInstand // 返回 ASTConsumer 单: return std::unique_ptr-mew FindNamedClass 9 } 10 }; 1 Class FindNamedClassConsumer with public: 3 virtual void Handle Jeph, Handle Translation Unit() 函证 Recursive AST Visitor 来遍历 virtual void Handle Translation Visitor. Traverse Decl()	n: public clang::A r 方法 <clang::astconsumer &compiler,="" 1,="" 20="" betranslationunit()="" ce="" clang::="" context.gettra<="" context.gettranslation="" decl="" isitor="" llvm:="" mer:="" public="" r<clang::astconsumer="" sconsumer);="" sitor="" td="" 元="" 单元。具体代码="" 数会在接收每个市域="" 是一个可以访问="" 来=""><td>eateASTConsumer 函 STFrontendAction {</td><td>重和 AST 明明 是制代码 是一个 AST 明明 是是一个 是一个 是一个 是一个 AST Visitor 的正正是一个 AST Visitor 的正是一个 AST Visitor 和正是一个 AST Visitor 和正是</td></clang::astconsumer>	eateASTConsumer 函 STFrontendAction {	重和 AST 明明 是制代码 是一个 AST 明明 是是一个 是一个 是一个 是一个 AST Visitor 的正正是一个 AST Visitor 的正是一个 AST Visitor 和正是一个 AST Visitor 和正是
ASTFrontendAction 在 AST 上自Consumer,以获取 AST 上的 AST Consumer,以获取 AST 上的 AST Consumer,以获取 AST 上的 AST Consumer,以实现 CreateASTConsumed virtual std::unique_ptrclamptocomputer std::unique_ptrcl	STCOnsumer 单元。 n: public clang::A n: public clang::A rocang::ASTConsumer cce & Compiler, llvm: 元 rcclang::ASTConsumer sconsumer); 1, 是—个可以以访问() 要为为人。在接触的人。是一个可以中的人。 decl 单元。具体代码 mer: public clang:: slationUnit(clang:: slationUnit(clang:: slationUnit(clang:: slationUnit(clang:: slationYi context.getTranslation tor 的方過溯,就写 Clang in cursiveAST + struct/union/class 多類以控制 RecursiveAST + struct/union/class S数以控制 Recursive CXXRecordDecl 的 or sitor <findnamedclass (cxxrecorddecl="" and="" ast,="" clang="" consider="" consider<="" cursiveast,="" false="" special="" td="" 表示停力=""><td>eateASTConsumer 函 STFrontendAction { CreateASTConsumer(StringRef InFile) { CreateASTConsumer(StringRef InFile) {</td><td>重AST 声单单 看 ST 中国 国际 AST 可用 用 制代</td></findnamedclass>	eateASTConsumer 函 STFrontendAction { CreateASTConsumer(StringRef InFile) { CreateASTConsumer(StringRef InFile) {	重AST 声单单 看 ST 中国 国际 AST 可用 用 制代
ASTFrontendAction 在 AST 上自 Consumer,以获取 AST 上的 AST Consumer,以获取 AST 上的 AST Consumer,以获取 AST 上的 AST Consumer,以获取 AST 上的 AST Consumer,以交现 CreateASTConsume 4 virtual std::unique_ptr-5 clang::CompilerInstant 6 // 返回 ASTConsumer 单:	n: public clang::A n: public clang::A n: public clang::A cclang::ASTConsumer cce &Compiler, llvm: 元 r <clang::astconsumer 1,="" clang::="" decl="" etranslationunit()="" mer:="" public="" sconsumer);="" slationunit(cl<="" slationunit(clang::="" td="" 函数会在接位的每个节点="" 单元。具体代码="" 是—个可以以访问问=""><td>eateASTConsumer 函 STFrontendAction { CreateASTConsumer(:StringRef InFile) {</td><td>重AST 声单单 看 ST 中国 国际 AST 可用 用 制代</td></clang::astconsumer>	eateASTConsumer 函 STFrontendAction { CreateASTConsumer(:StringRef InFile) {	重AST 声单单 看 ST 中国 国际 AST 可用 用 制代
ASTFrontendAction 在 AST 上自Consumer,以获取 AST 上的 AST Consumer,以获取 AST 上的 AST Consumer,以获取 AST 上的 AST Consumer,以获取 CreateASTConsume 4 virtual std::unique_ptr-5 clang::CompilerInstanne6 // 返回 ASTConsumer 单 7 return std::unique_ptr-6 new FindNamedClass 9 } 10 }; ASTCOnsumer 可以提供很多人口HandleTopLevelDecl() A HandleTopLevelDecl() A HandleTopLevelDecl() A HandleTopLevelDecl() A HandleTopLevelDecl() A HandleTopLevelDecl() A HandleTopLevelDecl() A HandleTranslationUnit() 函	STConsumer 单元。 n: public clang::A n: public clang::A n: public clang::A rr 方法 cclang::ASTConsumer cc &Compiler, llvm: 元 cclang::ASTConsumer sconsumer); 1, 是一个可以认访问, 要为为人。 decl 单元。具体代码 mer: public clang: slationUnit(clang:: slationUn	eateASTConsumer 函 STFrontendAction { > CreateASTConsumer(:StringRef InFile) { > AST 的抽象以关键样数。 AST 的抽象以关键样数。 i数如下: :ASTContext & Context &	重AST 声单单
ASTFrontendAction 在 AST 上自Consumer, 以获取 AST 上的 AST Consumer, 以获取 AST 上的 AST Consumer 以提供证明 Std::unique_pt-sclang::CompilerInstan-6	STConsumer 单元。 n: public clang::A r/ file clang::ASTConsumer cet & Compiler, llvm: crclang::ASTConsumer crclang::ASTConsumer crclang::ASTConsumer sconsumer); 1, 是一个可以访问问。 petranslationUnit() 数数会在接收每个节点 decl 单元。具体代码 mer: public clang:: sitor was a clang: sitor was a clang: sitor was a clang: sitor hops isitor; 点问 AST 式历来回调用用国调的 Translation context.getTranslation tor hops application context.getTranslation context.g	eateASTConsumer 函 STFrontendAction { > CreateASTConsumer(:StringRef InFile) { AST 的抽象以接接这这。 AST 的抽象以定调。 i数如下: :ASTContext &Context) :ASTConsumer { ASTConsumer { AST	重AST 声单单 包含 Trick看 Land 不知 是别代码 land land land land land land land land
ASTFrontendAction 在 AST 上自 AST Consumer,以获取 AST 上的 AST Consumer,以获取 AST 上的 AST 代码示例如下所示: 1 class FindNamedClassAction public: 3 // 实现 CreateASTConsumer 4 virtual std::unique_ptr clang::ComplerInstan 6 // 返回 ASTCOnsumer 单 7 return std::unique_ptr 8 new FindNamedClass 9 } 10 }; 11 ASTCOnsumer 可以提供很多入口HandleTopLevelDecl() 和 HandleTopLevelDecl() 和 HandleTopLevelDecl() 函 HandleTranslationUnit() 函	STConsumer 单元。 n: public clang::A n: r jx clang::ASTConsumer ce &Compiler, llvm: cclang::ASTConsumer ce &Compiler, llvm: cclang::ASTConsumer ce &Compiler, llvm: cclang::ASTConsumer cconsumer); 1, 是一个可以访问() 要 数会在接收每个节点 deTranslationUnit() 函数 decl 单元。具体代码 mer: public clang:: sitor x jaction y jaction	eateASTConsumer 函 STFrontendAction { CreateASTConsumer(StringRef InFile) { CreateASTConsumer(StringRef InFile) {	重AST 声明 元 自 的进 时时 元 自 的进 时时 元 自 的进 阴 不 自 的进 以 自 的进 以 自 的 的 正 的 码 的 强 和 的 不 自 自 的 正 的 码 的 码 的 码 自 自 的 码 的 码 自 自 自 自 自 自 自 自
ASTFrontendAction 在 AST 上的 AST Consumer, 以获取 AST 上的 AST Consumer, 以获取 AST 上的 AST 代码示例如下所示: 1 class FindNamedClassAction 2 public: 3	STConsumer 单元。 n: public clang::A n: public clang::A rotation (Compiler, llvm: roce & Compiler, llvm: roce & Consumer); ll, De Justion (Compiler) ll, D	EateASTConsumer 函 STFrontendAction { CreateASTConsumer(:StringRef InFile) { AST 的抽象以类的转量上级的数面的数面的数面的数面的数面的数面的数面的数面的的数面的数面的的数面的形式。 ASTContext & Context & Context) inUnitDecl 函数通过重量的以下, inUnitDecl 函数通过重量的可能可能可能可能可能可能可能可能可能可能可能可能可能可能可能可能可能可能可能	重A层 为 看 eA层 为 有 eA层 为 自 land 有 land fland fland hand fland
ASTFrontendAction 在 AST 上的 AST List Consumer, 以获取 AST 上的 AST Consumer, 以获取 AST 上的 AST Consumer, 以获取 AST 上的 AST Consumer, 以获取 AST List AST Consumer devirual std::unique_ptr-5 clang::CompilerInstand // 返回 ASTConsumer devirual std::unique_ptr-6 clang::CompilerInstand // 返回 ASTConsumer devirual std::unique_ptr-7 return std::unique_ptr-8 new FindNamedClass:	STConsumer 单元。 n: public clang::ASTConsumer ce & Consumer ce & Consumer); n: public clang::ASTConsumer ce & Consumer); n: pe Translation Unit()	eateASTConsumer 图 STFrontendAction { CreateASTConsumer(StringRef InFile) { CreateASTConsumer(StringRef InFile) {	重A层 为 看 eA层 为 有 eA层 为 自 land 有 land fland fland hand fland
ASTFrontendAction 在 AST 上自 AST Lei Consumer,以获取 AST 上的 AST Consumer,以获取 AST 上的 AST Consumer,以求取 AST Lei AST Consumer,以才以正如 Std:unique_ptroclang:compilerInstan // 返回 ASTConsumer 可以提供很多入口	STConsumer 单元。 n: public clang::ASTConsumer ce & Consumer ce & Consumer); n: public clang::ASTConsumer ce & Consumer); n: pe Translation Unit()	eateASTConsumer 图 STFrontendAction { CreateASTConsumer(StringRef InFile) { CreateASTConsumer(StringRef InFile) {	重A层为 \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$
ASTFrontendAction 在 AST 上的 AST Consumer, 以获取 CreateASTConsumer 4 virtual std::unique_ptr clang::CompilerInstan // 返回 ASTConsumer 单 return std::unique_ptr nev FindNamedClass	STCOnsumer 单元。 STCOnsumer 单元。 A : public clang::A A : r / 5法 Cclang::ASTConsumer ce & Compiler, llvm: Traclang::ASTConsumer ce & Compiler, llvm: Traclang::ASTConsumer sconsumer); 1, 是一可以访问等 1, 是一可以归归() 2, 是一可以归归() 3数会在接收何人。 4。	eateASTConsumer 函 STFrontendAction { STFrontendAction { CreateASTConsumer(StringRef InFile) { Processed in a substitute	重ACE 为 载 CI (The state of the
ASTFrontendAction 在 AST 上的 AST	STConsumer 单元。 n: public clang::A r 方法	eateASTConsumer 函 STFrontendAction {	重A层 th 看到 th Hitch Hitc
ASTFrontendAction 在 AST 上自 Consumer, 以获取 AST 上的 AST Lib AST L	STCOnsumer 单元。 STCOnsumer 单元。 In: public clang::Asclang::ASTConsumer ce & Compiler, llvm: Traclang::ASTConsumer ce & Compiler, llvm: Traclang::ASTConsumer ce & Consumer); In, 是一个可以访问和全个节点 是一个可以访问和全个节点 是在接收每个节点 是在接收每个节点 是在接收每个节点 是在接收每个节点 是在接收每个节点 是对方式玩来回调的下点。由于这个方面,是是一个方面,然后是一个方面,是是一个方面,然后是一个方面,然后是一个方面,然后是一个方面,是是一个方面,然后是一个方面,是是一个方面,然后是一个方面,是是一个方面,然后是一个方面,是一个方面,是一个一面,是一个一面,一面,是一个一面,是一个一面,一面,一面,一面,一面,一面,一面,一面,一面,一面,一面,一面,一面,一	eateASTConsumer 图 STFrontendAction { > CreateASTConsumer(:StringRef InFile) { Provided Action (> CreateASTConsumer(:StringRef InFile) { Provided Action (AST May	重A层 that the state of the stat
ASTFrontendAction 在 AST 上色 Consumer, 以表现 AST 上的 AST Consumer, 以表现 AST 上的 AST Consumer, 以表现 AST Lend Ast Consumer, 以表现 Consumer AST Lend Ast Consumer, 以表现 Consumer AST Lend Ast Consumer A	STCONSUMER 单元。 n: public clang::A n: public clang::A r: ring::ASTConsumer ccc & Compiler, tlvm: ccclang::ASTConsumer ccc & Compiler, sltvm: ccclang::ASTConsumer ccc & Compiler, sltvm: ccclang::ASTConsumer ccconsumer); 1, 是一个可以访问() 数数一可以协作。 ccconsumer); 1, 是一个可以访问每个方式 ccconsumer); 1, 是一个可以访问每个方式 ccconsumer); 1, 是一个可以访问每个方式 ccconsumer); 2, 是一个可以访问每个方式 ccconsumer); 2, 是一个可以访问有个。 ccconsumer cccconsumer ccconsumer cccconsumer ccconsumer cccconsumer ccconsumer cccconsumer cccco	eateASTConsumer 函 STFrontendAction { CreateASTConsumer(:StringRef InFile) { CreateASTConsumer(:StringRef InFile) { Profession of the set	重A层 that the state of the stat
ASTFrontendAction 在 AST 上色 Consumer, 以获取 AST 上的 AST	STCOnsumer 单元。 n: public clang: A r: 7 方法 clang::ASTConsumer ce &Compiler, llvm: clang::ASTConsumer ce &Compiler, llvm: clang::ASTConsumer ce &Compiler, llvm: clang::ASTConsumer consumer); 1. 是一个可以访问公司 consumer); 2. 是一个可以访问公司 consumer); 2. 是一个可以访问公司 consumer); 3. 是一个可以访问公司 consumer); 4. 是一个可以访问公司 consumer); 4. 是一个可以访问公司 consumer c	eateASTConsumer 函 STFrontendAction { A CreateASTConsumer(重A层 that the state of the stat
ASTFrontendAction 在 AST 上色 Consumer, 以	Translation Unit (clarg: in	eateASTConsumer 函 STFrontendAction { > CreateASTConsumer(:StringRef InFilen) { re(AST 的函数 purple in	重A层 为 th
AST Frontened Action 完 AST 上色 AST Frontened Action 完 AST 上色 AST Econsumer, 以获取 AST 上的 AST Action (### Processor	eateASTConsumer 函 STFrontendAction { STFrontendAction { CreateASTConsumer(StringRef InFile) { CreateASTConsumer(StringRef InFile) { Proceed ASTContext AST May be proceed as the process of th	重A层 大
### Consumer, 以获取 AST 上的 AST	Translation Tran	eath AST Consumer 函 STFrontend Action { STFrontend Action { Created STConsumer { Created STConsumer { Created STConsumer { STFrontend Action { Created STConsumer { Cre	重AS房 物 (The state of the sta
AST-Frontenid Action ACT Lei Consumer, 以获取 AST Lei Consumer, 以获取 AST Lei AST	n: public clang: A r false scalaging in special size of the section of the secti	eate AST Consumer 区 STFrontendaction { STFr	重A层 为 4节 载 C T I I I I I I I I I I I I I I I I I I
### Compared to the compared	Transition of the property of	### ### ### ### ### ### ### ### ### ##	重A层 大
### ### ### ### ### ### ### ### ### ##	Translation () () () () () () () () () () () () ()	STFrontendAction { STFrontendAc	重A层 为 \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$
### Cast	TCONSUMER P. C.	### ### ### ### ### ### ### ### ### ##	重ASE 大り もい では
### Class Findhamed Class Action 1	TCONSUMER P. C.	Bath AST Consumer is a space in the famous	重A层 数 和 (词。
### ### ### ### ### ### ### ### ### ##	Traconsumer 单元。 Traconsumer 中元。 Traco	eath AST Consumer is is strontendaction { STF-ontendaction { ST	重AS声单 \$\frac{1}{2}\$\$
### Compared Class Fundamental Class Fundamenta	m: public clang::A m: public clang::A m: public clang::ASTConsumer ce & Compiler, lum: perceasionsumer); loop and in public clang::ASTConsumer ce & Compiler, clang::ASTConsumer consumer); loop and clang::ASTConsumer consumer); consumer consumer); consumer consu	TETTANACE OF A STANDERS OF A SASA STANDERS OF A SA	型
### AST FOOT ACT ACT ACT ACT ACT ACT ACT ACT ACT AC	n: public clang::A n: public clang::A n: public clang::A reclassumer 单元。 n: public clang::A reclassumer 单元。 n: public clang::A reclassumer in public clang: rec	STF-ontendaction { STF-o	The state of the
### Clang 指件 ### Clang 指件 ### Clang 加	### Party::Add-MyPulushandandandandandandandandandandandandanda	STF- ontendaction {	重A层 数 A Tan
### Clang 描字 #### Clang 描字 ##### Clang 描字 ###### Clang 描字 ####### ########### ############ ####	Transition of the property of	TET-NOTE AND ADDRESS AND ADDR	Tan Andrew Andr