LLVM库文件分析与使用方法

目录

[1. LLVM Support Library Manual 1](#_Toc23924945)

[1.1 LLVM Support Library 介绍 1](#_Toc23924946)

[1.2 CommandLine 2.0 Library Manual 1](#_Toc23924947)

[1.2.1 CommandLine基本结构 1](#_Toc23924948)

[1.2.2 CommandLine其他形式 2](#_Toc23924949)

[1.2.3 CommandLine高级特性 3](#_Toc23924950)

# 1. LLVM Support Library Manual

## 1.1 LLVM Support Library 介绍

Llvm support 库是llvm软件架构里面的工具类，包括文件处理，错误信息, 命令行参数，格式转换，哈希码生成等等，有很多工具类可以使用，当你需要某类基础功能时，不妨先看看这个库是否已经有合适的。

## 1.2 CommandLine 2.0 Library Manual

### 1.2.1 CommandLine基本结构

CommandLine是LLVM软件系统里面针对命令行参数处理的库。使用该库可以方便应用程序的参数设计，它使用了声明式的方法构建命令行参数，即当你需要支持某个新的命令行参数时，只需要一句简单的全局声明即可。

使用该库的方法为：

第一步： #include "llvm/Support/CommandLine.h" // 加入main.cpp

第二步：

int main(int argc, char \*\*argv) {

cl::ParseCommandLineOptions(argc, argv); // 加入main函数

...

}

第三步：声明命令行参数

每条命令行参数生成需要3个参数，常见命令行结构如下，例如

cl::opt<string> OutputFilename("o", cl::desc("Specify output filename"), cl::value\_desc("filename"));

声明了一个OutputFilename全局变量，

第一个参数：针对‘-o’参数的， string类型是说明解析的参数在程序里面是以string类型体现的，

第二个参数：用于-help参数打印，用于命令行参数说明

第三个参数：用于-help参数打印，用于参数输入格式说明

命令行打印结果如下：

USAGE: compiler [options]

OPTIONS:

-h - Alias for -help

-help - display available options (-help-hidden for more)

-o <filename> - Specify output filename

第四步：最后你的主程序就可以根据命令行参数，运行不同的执行函数，你可以将参数理解为外部传入的全局变量，根据变量值执行不同函数。

### 1.2.2 CommandLine其他形式

CommandLine将命令行参数分为标量式(scalar)（bool,strings,integers,enums,double）和链式(list)。它们分别使用cl::opt和cl::list接口进行构造。结构如下：

标量式：

cl::opt<bool> Force ("f", cl::desc("Enable binary output on terminals"));

cl::opt<string> OutputFilename("o", cl::desc("Specify output filename"), cl::value\_desc("filename"));

enum OptLevel {

g, O1, O2, O3

};

cl::opt<OptLevel> OptimizationLevel(cl::desc("Choose optimization level:"),

cl::values(

clEnumVal(g , "No optimizations, enable debugging"),

clEnumVal(O1, "Enable trivial optimizations"),

clEnumVal(O2, "Enable default optimizations"),

clEnumVal(O3, "Enable expensive optimizations")));

链式：

enum Opts {

// 'inline' is a C++ keyword, so name it 'inlining'

dce, constprop, inlining, strip

};

cl::list<Opts> OptimizationList(cl::desc("Available Optimizations:"),

cl::values(

clEnumVal(dce , "Dead Code Elimination"),

clEnumVal(constprop , "Constant Propagation"),

clEnumValN(inlining, "inline", "Procedure Integration"),

clEnumVal(strip , "Strip Symbols")));

### 1.2.3 CommandLine高级特性

还有更多的细节问题你可能有兴趣研究，如参数别名，将参数设置为flag集合，参数分组，占位参数，参数初始化等等，你可以参考

<https://www.llvm.org/docs/CommandLine.html#the-cl-opt-class>。