构建llvm-mctoll

llvm-mctoll是微软的可以静态将二进制文件转译为LLVM-IR（Intermediate representation）的工具。构建它不需要安装任何第三方依赖。以下是在2023.7.6日尝试编译的过程：

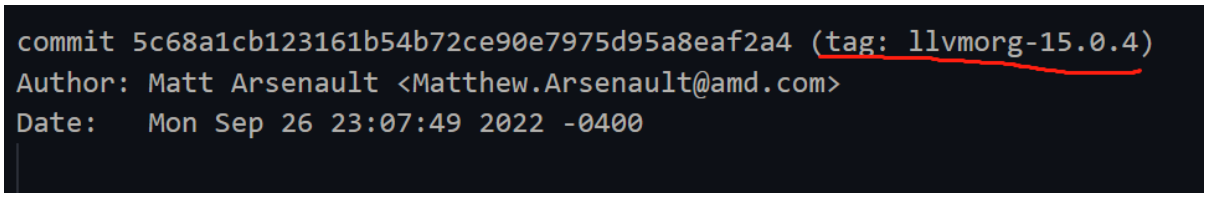
1. 找到它的仓库（https://github.com/microsoft/llvm-mctoll），找到一个你想布置工程的位置，并且运行下列指令：

```
git clone https://github.com/llvm/llvm-project.git
cd llvm-project && git clone -b master https://github.com/microsoft/llvm-mctoll.git llvm/tools/llvm-mctoll
```

2. 检查llvm的版本和llvm-mctoll是否匹配，运行下列指令(此时在llvm-project文件夹下)：

```
vim ./llvm/tools/llvm-mctoll/llvm-project-git-commit-to-use.txt #cat之类的查看命令也可以
```

   接下来找到下图中画线的话



   这边显示llvm的15.0.4版本与当前的llvm-mctoll匹配，故需要将llvm分支切换到15.0.4版本，使用如下命令：

```
git checkout remotes/origin/release/15.x
#或者采用切换tag的方式，小版本也是对的: git checkout llvmorg-15.0.4
```

   如果你画线部分不是15.0.4，使用：

```
git branch -a #查看可用分支，在其中找寻对应的匹配的版本
#或者使用git tag 查看，切换到对应tag
```

3. 新建一个build文件夹，执行下面的命令（来自官方README）构建llvm的项目目录树：

```
mkdir build
cmake -S llvm -B ../build -G "Ninja" \
  -DLLVM_TARGETS_TO_BUILD="X86;ARM;RISCV;AArch64"  \
  -DLLVM_ENABLE_PROJECTS="clang;lld" \
  -DLLVM_ENABLE_ASSERTIONS=true      \
  -DCLANG_DEFAULT_PIE_ON_LINUX=OFF   \
  -DCMAKE_BUILD_TYPE=Debug
```

```
#或者用下面这条

cmake -S llvm -B ../build -G "Ninja" \
  -DLLVM_TARGETS_TO_BUILD="X86;ARM;RISCV;AArch64" \
  -DLLVM_ENABLE_PROJECTS="clang;lld" \
  -DLLVM_ENABLE_ASSERTIONS=true \
  -DCLANG_DEFAULT_PIE_ON_LINUX=OFF \
  -DCMAKE_BUILD_TYPE=Debug \
  -DLLVM_ENABLE_EH=1 \
  -DLLVM_ENABLE_RTTI=1 \
  -DLLVM_USE_LINKER=mold
```

4. 通过构建的llvm项目目录树构建llvm-mctoll：

```
cd build
ninja
```

   如果都没有报错，这时，在./build/bin文件夹下就会出现一个llvm-mctoll的二进制文件，至此就算是成功构建了。

5. 可以最后顺便再跑一个单元测试测试功能是否正常：

```
cd ./build
ninja check-mctoll
```