#6 MiniSQL项目框架

框架链接：ZJU GitLab链接，请使用内网访问 <https://git.zju.edu.cn/zjucsdb/minisql>

#0 框架维护日志（更新中）

2022-04-15

- 在部分 `ASSERT(false)` 语句后添加返回值，不使用MSVC编译器的同学可以忽略。本修改提交在 `msvc` 分支上，Commit SHA为 `801d0cd916915ff93d5825e81388f65a6cec4e8a` 。

#1 代码框架介绍

本实验基于CMU-15445 BusTub框架，并做了一些修改和扩展。注意：为了避免代码抄袭，请不要将自己的代码发布到任何公共平台中。

#1.1 编译&开发环境

- `apple clang` : 11.0+ (MacOS)，使用 `gcc --version` 和 `g++ --version` 查看
- `gcc` & `g++` : 8.0+ (Linux)，使用 `gcc --version` 和 `g++ --version` 查看
- `cmake` : 3.20+ (Both)，使用 `cmake --version` 查看
- `gdb` : 7.0+ (Optional)，使用 `gdb --version` 查看
- `flex` & `bison` (暂时不需要安装，但如果需要对SQL编译器的语法进行修改，需要安装)

#1.2 构建

#1.2.1 Windows

目前该代码暂不支持在Windows平台上的编译。但在Win10及以上的系统中，可以通过安装WSL（Windows的Linux子系统）来进行开发和构建。WSL请选择Ubuntu子系统（推荐Ubuntu20及以上）。如果你使用Clion作为IDE，可以在Clion中配置WSL从而进行调试，具体请参考链接 Clion with WSL <https://blog.jetbrains.com/clion/2018/01/clion-and-linux-toolchain-on-windows-are-now-friends/> 。

#1.2.2 MacOS & Linux & WSL

基本构建命令

```bash
mkdir build
cd build
cmake ..
make -j
```

若不涉及到 `CMakeLists` 相关文件的变动且没有新增或删除 `.cpp` 代码（通俗来说，就是只是对现有代码做了修改）则无需重新执行 `cmake..` 命令，直接执行 `make -j` 编译即可。默认以 `debug` 模式进行编译，如果你需要使用 `release` 模式进行编译：

```bash
cmake -DCMAKE_BUILD_TYPE=Release ..
```

#1.3 测试

在构建后，默认会在 `build/test` 目录下生成 `minisql_test` 的可执行文件，通过 `./minisql_test` 即可运行所有测试。如果需要运行单个测试，例如，想要运行 `lru_replacer_test.cpp` 对应的测试文件，可以通过 `make lru_replacer_test` 命令进行构建。

#1.4 工程目录

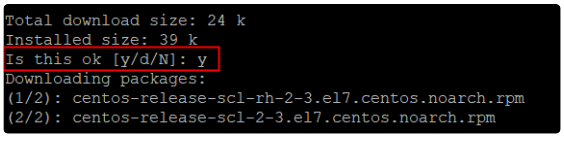
- `src` ：与MiniSQL工程相关的头文件和源代码。`src/include` 中为MiniSQL各个子模块的头文件，`src/buffer` 、`src/record` 、`src/index` 、`src/catalog` 等目录为MiniSQL各个子模块的源代码。
- `test` ：与测试用例相关的源代码和头文件。
- `thirdparty` ：第三方库，包括日志模块 `glog` 和测试模块 `gtest` 。

#2 使用WSL-Ubuntu进行开发

Note：Win10系统下，参考Win10系统安装WSL教程 <https://www.cnblogs.com/jetttang/p/8186315.html> 安装WSL，选择Ubuntu子系统即可，推荐选用Ubuntu 20.04以上的版本。

#2.1 配置编译环境

首次安装时，请使用 `sudo apt-get update` 更新软件源。然后使用命令 `sudo apt install gcc g++ cmake gdb` 安装编译和调试环境。
Note: 安装时一般都会提示是否需要安装，输入 `y` 回车即可：



安装完成后，通过 `--version` 查看是否安装完成，如下图所示：

```
ying@DESKTOP-COSICRG:~$ gcc --version
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

ying@DESKTOP-COSICRG:~$ g++ --version
g++ (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

ying@DESKTOP-COSICRG:~$ cmake --version
cmake version 3.16.3

CMake suite maintained and supported by Kitware (kitware.com/cmake).
ying@DESKTOP-COSICRG:~$ gdb --version
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

进入一个目录，从远程仓库中克隆代码到该目录下（建议先Fork到自己小组的私有仓库然后再进行克隆），在这里我选择将代码克隆到 `/mnt/f` 目录下（当然这个目录可以根据你的需要自由选择）， `/mnt/f` 目录实际上就是我们电脑本地磁盘的 `F` 盘：

Bash 复制代码

```bash
cd /mnt/f
git clone https://git.zju.edu.cn/zjucsdb/minisql.git
```

Note：如果在克隆过程中提示 `Permission Denied` ，请在命令前面加上 `sudo` 以执行：

然后进入目录，进行构建或测试：

Bash 复制代码

```bash
# 进入目录
cd /mnt/f/minisql
# 建立并进入build目录
mkdir build
cd build
# 生成Makefile
sudo cmake ..
# 多线程编译生成可执行文件，-j可以指定具体的线程数，如-j4就是使用4线程编译
make -j
```

`cmake` 构建成功后如下图所示：

`make` 构建成功后如下图所示：

编译生成的可执行文件位于 `bin/` 和 `test/` （测试相关文件）下：

最终整个MiniSQL的主程序在这里：

一个运行测试用例的例子：

#2.2 使用Clion连接WSL进行开发

Note： Clion下载网址 <https://www.jetbrains.com/zh-cn/clion/download/#section=windows> ，开始时有30天的免费试用期。在使用ZJU的邮箱进行认证后可以一直免费使用。

打开Clion后，导入MiniSQL项目：

File-->Settings-->Build-->Toolchains 添加WSL相关设置：

File-->Settings-->Build-->CMake中添加CMake相关设置：

保存后会自动运行CMake命令，或是通过下图左边刷新按钮运行。CMake构建成功后如下图所示：

接下来就可以使用Clion进行开发和使用。

#2.3 使用VSCode连接WSL进行开发

Note：VSCode需要事先在扩展 `Ctrl+Shift+X` 中安装以下插件：

- Remote-WSL（在本地安装）
- C/C++ （连接上WSL后再安装，安装在WSL）
- CMake Tools（连接上WSL后再安装，安装在WSL）

然后使用 `Ctrl+Shift+P` 打开选项卡输入 `WSL` ，选择 `Remote-WSL:New Window` 即可打开WSL。可以看到，左下角已连接的Linux子系统WSL:Ubuntu-20.04。

然后打开源代码所在目录：

在WSL中安装C/C++和CMake插件：

安装成功后可以看到下面的工具，对CMake进行Configure：

选择WSL中的编译器：

构建完成后：

点击 `Build` 即可生成可执行文件。

#3 使用MacOS进行开发

MacOS中一般自带Apple Clang，可以使用 `g++ --version` 和 `gcc --version` 查看， `cmake` 和 `gdb` 可以通过 `brew` 命令安装（或是从官网上下载然后添加到环境变量中）。在MacOS中可以使用Clion和VS Code直接在本地进行开发调试，方法与#2中提到的类似。

#4 远程连接服务器进行开发

同学们可以根据自己服务器选择合适的Linux镜像，推荐选用Ubuntu 20.04+或是CentOS 7.2+的镜像。若选用Ubuntu的镜像，请参考#2中的教程进行编译环境的配置，然后参考#4.2连接远程服务器进行开发调试。**在本节的示例中，服务器镜像选用的是CentOS 7.2。**

Note: 由于外网服务器正常情况下无法访问位于学校内网的 `ZJU GitLab`，一个可行的解决办法是，先从 `ZJU GitLab` 上克隆源代码到本地，然后将代码推送到自己小组**私有**的远程 `GitHub` 仓库中，这样外网服务器就可以通过 `GitHub` 存储库访问到代码。另外一种可行的方法是，通过 `scp` 命令将源代码直接上传到远程服务器中，然后在远程服务器中新建 `Git` 仓库。

#4.1 配置编译环境

本节以**CentOS 7.2镜像**进行示例。对于其它镜像，配置编译环境的方法类似，可以自行网上搜索在该类型的系统镜像中如何安装 `GCC` 、 `G++` 、 `CMake` 和 `GDB` 。

服务器镜像中如果自带 `GCC` 、 `G++` 、 `CMake` 和 `GDB` ，但版本较低的（如下图中 `GCC` 的版本是4.8.5），则需要对相应的软件进行升级（具体升级教程可上网查找）。

CentOS 7.2镜像中自带了 `GCC` （但未自带 `G++` ），在这里简单叙述一下CentOS 7.2升级 `GCC` 、 `G++` 的方法，升级方法参考引用链接 <https://www.cnblogs.com/jixiaohua/p/11732225.html> ，命令如下：

Bash 复制代码

```
# 安装centos-release-scl
sudo yum install centos-release-scl
# 安装devtoolset
sudo yum install devtoolset-9-gcc*
sudo yum install devtoolset-9-g++*
# 替换旧的gcc和g++
mv /usr/bin/gcc /usr/bin/gcc-4.8.5
ln -s /opt/rh/devtoolset-9/root/bin/gcc /usr/bin/gcc
mv /usr/bin/g++ /usr/bin/g++-4.8.5  # Note: 如果CentOS中没有自带g++,
                                    # 即g++ --version提示命令不存在,
                                    # 则不需要执行该步命令，只需要执行下面的ln即可。
ln -s /opt/rh/devtoolset-9/root/bin/g++ /usr/bin/g++
```

Note: 安装时一般都会提示是否需要安装，输入 `y` 回车即可：

升级完成后，查看 `GCC` 和 `G++` 是否正确安装：

由于CentOS 7.2中通过 `yum` 源安装的 `CMake` 版本较老（2.X版本），因此需要从官网下载，下载链接：CMake Download <https://cmake.org/download/> ，根据不同的CPU架构，选择不同的链接下载，上面的是X86架构，下面的是ARM架构：

然后通过 `wget` 命令进行下载：

Bash 复制代码

```
# X86架构
wget https://github.com/Kitware/CMake/releases/download/v3.23.0/cmake-3.23.0-linux-x86_64.tar.gz
# ARM架构
wget https://github.com/Kitware/CMake/releases/download/v3.23.0/cmake-3.23.0-linux-aarch64.tar.gz
```

下载完成后：

```bash
# 解压压缩包
tar -xzvf cmake-3.23.0-linux-aarch64.tar.gz #ARM架构下使用该命令
tar -xzvf cmake-3.23.0-rc2-linux-x86_64.tar.gz #X86架构下使用该命令
# 重命名
mv cmake-3.23.0-linux-aarch64 cmake #ARM架构下使用该命令
mv cmake-3.23.0-rc2-linux-x86_64 cmake #X86架构下使用该命令
# 移动 & 链接
mv cmake /usr/local/
ln -s /usr/local/cmake/bin/cmake /usr/bin/cmake
```

使用 `cmake --version` 即可查看 `CMake` 是否安装成功：

调试工具 `GDB` 使用 `sudo yum install gdb` 命令直接安装即可，然后使用 `gdb --version` 查看是否安装成功：

#4.2 使用VSCode连接进行开发

Note：VSCode需要事先在扩展 `Ctrl+Shift+X` 中安装以下插件：

- Remote-SSH（在本地安装）
- C/C++ （连接服务器后再安装，安装在服务器）
- CMake Tools（连接服务器后再安装，安装在服务器）

安装Remote-SSH扩展后，点击“+”号新建连接：

在弹框中输入 `ssh <USERNAME>@<IP ADDRESS>`

选择任意一个配置文件保存，通常是选上面那个，保存到用户下的配置文件：

在SSH TARGETS中选择刚刚添加的服务器，连接：

输入密码后回车：

打开你存放MiniSQL代码的文件夹：

后续操作方法与#2.3中类似，这里不做赘述。