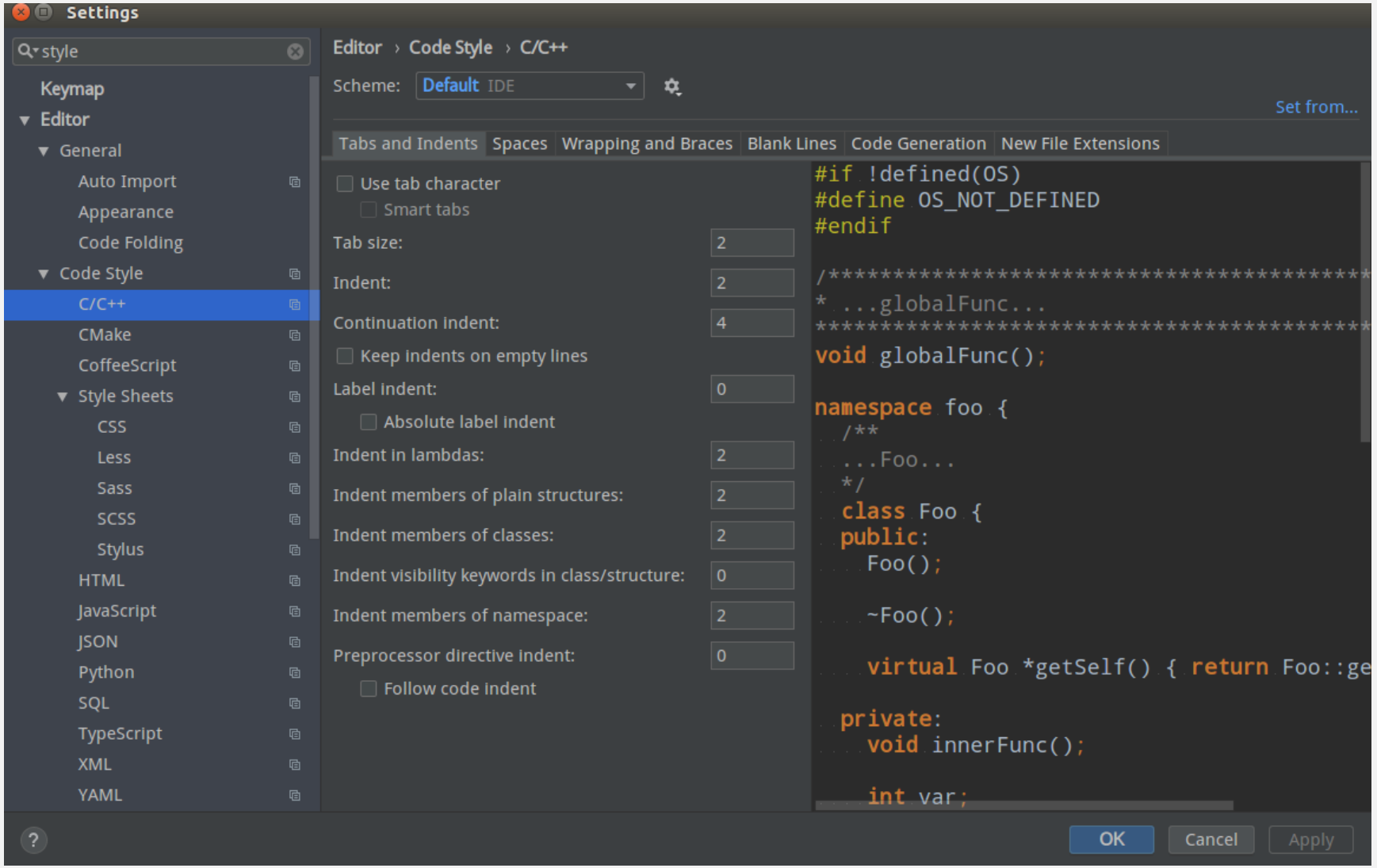
leveldb

接口练习

DaSE
Data Science
&Engineering

CONTENTS

预备工作


Settings
style
Keymap
Editor
General
Auto Import
Appearance
Code Folding
Code Style
C/C++
CMake
CoffeeScript
Style Sheets
CSS
Less
Sass
SCSS
Stylus
HTML
JavaScript
JSON
Python
SQL
TypeScript
XML
YAML
Editor › Code Style › C/C++
Scheme: Default IDE
Set from...
Tabs and Indents
Spaces
Wrapping and Braces
Blank Lines
Code Generation
New File Extensions
Use tab character
Smart tabs
Tab size: 2
Indent: 2
Continuation indent: 4
Keep indents on empty lines
Label indent: 0
Absolute label indent
Indent in lambdas: 2
Indent members of plain structures: 2
Indent members of classes: 2
Indent visibility keywords in class/structure: 0
Indent members of namespace: 2
Preprocessor directive indent: 0
Follow code indent
#if !defined(OS)
#define OS_NOT_DEFINED
#endif
/*****
* ...globalFunc...
*****
void globalFunc();
namespace foo {
/**
...Foo...
*/
class Foo {
public:
Foo();
~Foo();
virtual Foo *getSelf() { return Foo::ge
private:
void innerFunc();
int var;
OK
Cancel
Apply

预备工作

db_test.cc | db_WriteBatch.cc | db_snapshot.cc | write_batch.h | write_batch.cc | CMakeLists.txt

```
242     "${PROJECT_SOURCE_DIR}/db/leveldbutil.cc"
243             )
244     target_link_libraries(leveldbutil leveldb)
245
246     add_executable(db_test
247       "${PROJECT_SOURCE_DIR}/test/db_test.cc"
248             )
249     target_link_libraries(db_test leveldb)
250
251     add_executable(index_test
252       "${PROJECT_SOURCE_DIR}/value_index/index_test.cpp"
253             )
254     target_link_libraries(index_test leveldb)
255
256     add_executable(db_WriteBatch
257             "${PROJECT_SOURCE_DIR}/test/db_WriteBatch.cc"
258             )
259     target_link_libraries(db_WriteBatch leveldb)
260
261     add_executable(db_snapshot
262             "${PROJECT_SOURCE_DIR}/test/db_snapshot.cc"
263             )
264     target_link_libraries(db_snapshot leveldb)
```


EAST CHINA NORMAL UNIVERSITY
华东师范大学

01 执行过程


DaSE
Data Science
&Engineering

Open db

首先是数据库不存在情况下,则需要新建一个数据;

如果是数据存在的情况下,则需要从数据库恢复数据等

```
Status DB::Open(const Options& options, const std::string& dbname,
                DB** dbptr) {
  *dbptr = NULL;

  DBImpl* impl = new DBImpl(options, dbname);//new一个DB实现类
  impl->mutex_.Lock();
  VersionEdit edit;
  Status s = impl->Recover(&edit); // 这个函数处理的就是判断数据库是否存在
  //以及从磁盘恢复数据
  if (s.ok()) {
    uint64_t new_log_number = impl->versions_->NewFileNumber();
    WritableFile* lfile;
    //新生成一个manifest文件
    s = options.env->NewWritableFile(LogFileName(dbname, new_log_number),
                                     &lfile);
    if (s.ok()) {
      edit.SetLogNumber(new_log_number);//设置新的manifest文件编号,因为
    //impl->Recover函数改变了new_log_number
      impl->logfile_ = lfile;
      impl->logfile_number_ = new_log_number;
      impl->log_ = new log::Writer(lfile);
      //将磁盘数据恢复到内存之后又做了些改变重新写回磁盘,并设置当前版本
      s = impl->versions_->LogAndApply(&edit, &impl->mutex_);
    }
    if (s.ok()) {
      impl->DeleteObsoleteFiles();//删除过期文件,也就是上个版本的文件
      impl->MaybeScheduleCompaction();//可能执行合并
    }
  }
  impl->mutex_.Unlock();
  if (s.ok()) {
    *dbptr = impl;
  } else {
    delete impl;
  }
  return s;
}
```

impl->Recover函数执行流程

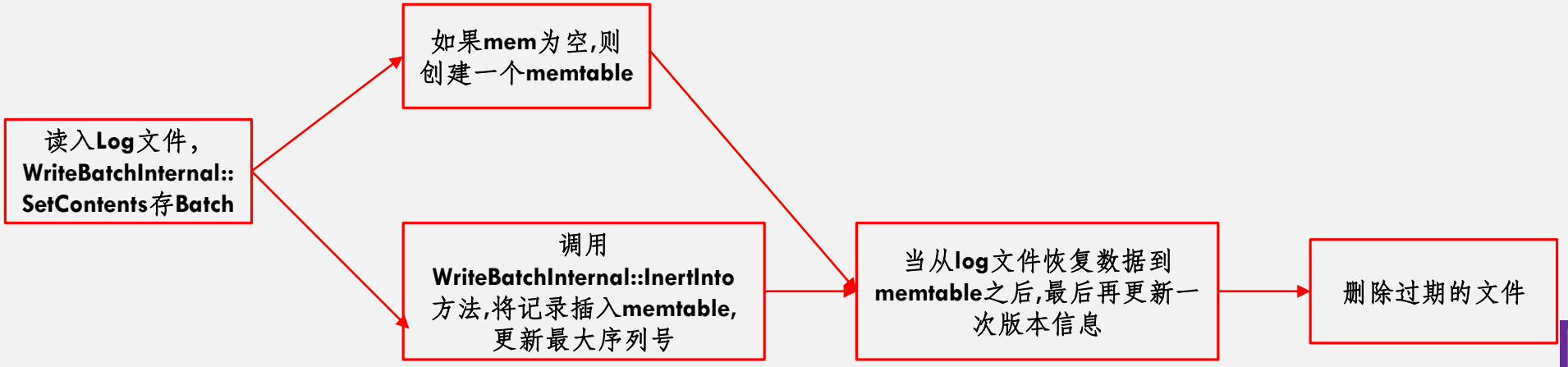
```
Status DBImpl::Recover(VersionEdit* edit) {
mutex_.AssertHeld();

  env_->CreateDir(dbname_);
  assert(db_lock_ == NULL);
  //锁住这个数据库目录
  Status s = env_->LockFile(LockFileName(dbname_), &db_lock_);
  if (!s.ok()) {
    return s;
  }
  //判断文件是否存在
  if (!env_->FileExists(CurrentFileName(dbname_))) {
    if (options_.create_if_missing) {//不存在,且设置了create_if_missing=true,
      s = NewDB();
      if (!s.ok()) {
        return s;
      }
    } else {
      return Status::InvalidArgument(
          dbname_, "does not exist (create_if_missing is false)");
    }
  } else {
    if (options_.error_if_exists) {//如果存在,且设置error_if_exists,则报错
      return Status::InvalidArgument(
          dbname_, "exists (error_if_exists is true)");
    }
  }
```

RocoverLogFile

当初始化版本信息之后,接来就是将log文件的数据恢复到memtable中,收集日志文件,然后按日志文件编号排序,最后调用RecoverLogFile恢复到memtable.

```
std::sort(logs.begin(), logs.end());
for (size_t i = 0; i < logs.size(); i++) {
  s = RecoverLogFile(logs[i], edit, &max_sequence);
  //因为RecoverLogFile这个函数会改变版本的一些属性,所以需要从新
  //设定文件编号
  versions_->MarkFileNumberUsed(logs[i]);
}
```

RecoverLogFile
执行流程




EAST CHINA NORMAL UNIVERSITY
華東師範大學

02 db写入操作


DaSE
Data Science
&Engineering

WriteBatch类

类的成员变量rep_，这个字符串用来存储这批操作的所有记录

```
WriteBatch::rep_:=
        sequence: fixed64
        count: fixed32
        data: record[count]
record: kTypeValue varstring varstring
        kTypeDeletion varstring
varstring:
       len: varint32
       data: uint8[len]
```

1.对于插入的记录，由kTypeValue+key长度+key+value长度+value组成

2.对于删除记录，由kTypeDelete+key长度+key组成

WriteBatch接口

主要是将记录添加进rep_以及从rep_中解析出所有记录。
将记录添加到rep_接口为:

```
1   void WriteBatch::Put(const Slice& key, const Slice& value) {
2     //先将记录数加1
3     WriteBatchInternal::SetCount(this, WriteBatchInternal::Count(this) + 1);
4     rep_.push_back(static_cast<char>(kTypeValue));//添加类型
5     PutLengthPrefixedSlice(&rep_, key);//添加key的长度和值
6     PutLengthPrefixedSlice(&rep_, value);//添加value的长度和值
7   }
8
9   void WriteBatch::Delete(const Slice& key) {
10    //设置记录数加1
11    WriteBatchInternal::SetCount(this, WriteBatchInternal::Count(this) + 1);
12    //添加类型
13    rep_.push_back(static_cast<char>(kTypeDeletion));
14    //添加key以及key值
15    PutLengthPrefixedSlice(&rep_, key);
16  }
```

上层方法调用

```
1   Status DBImpl::Put(const WriteOptions& o, const Slice& key, const Slice& val) {
2     return DB::Put(o, key, val);
3   }
4
5   Status DBImpl::Delete(const WriteOptions& options, const Slice& key) {
6     return DB::Delete(options, key);
7   }
8
9   Status DB::Put(const WriteOptions& opt, const Slice& key, const Slice& value) {
10    WriteBatch batch;
11    batch.Put(key, value);
12    return Write(opt, &batch);
13  }
14
15  Status DB::Delete(const WriteOptions& opt, const Slice& key) {
16    WriteBatch batch;
17    batch.Delete(key);
18    return Write(opt, &batch);
19  }
```

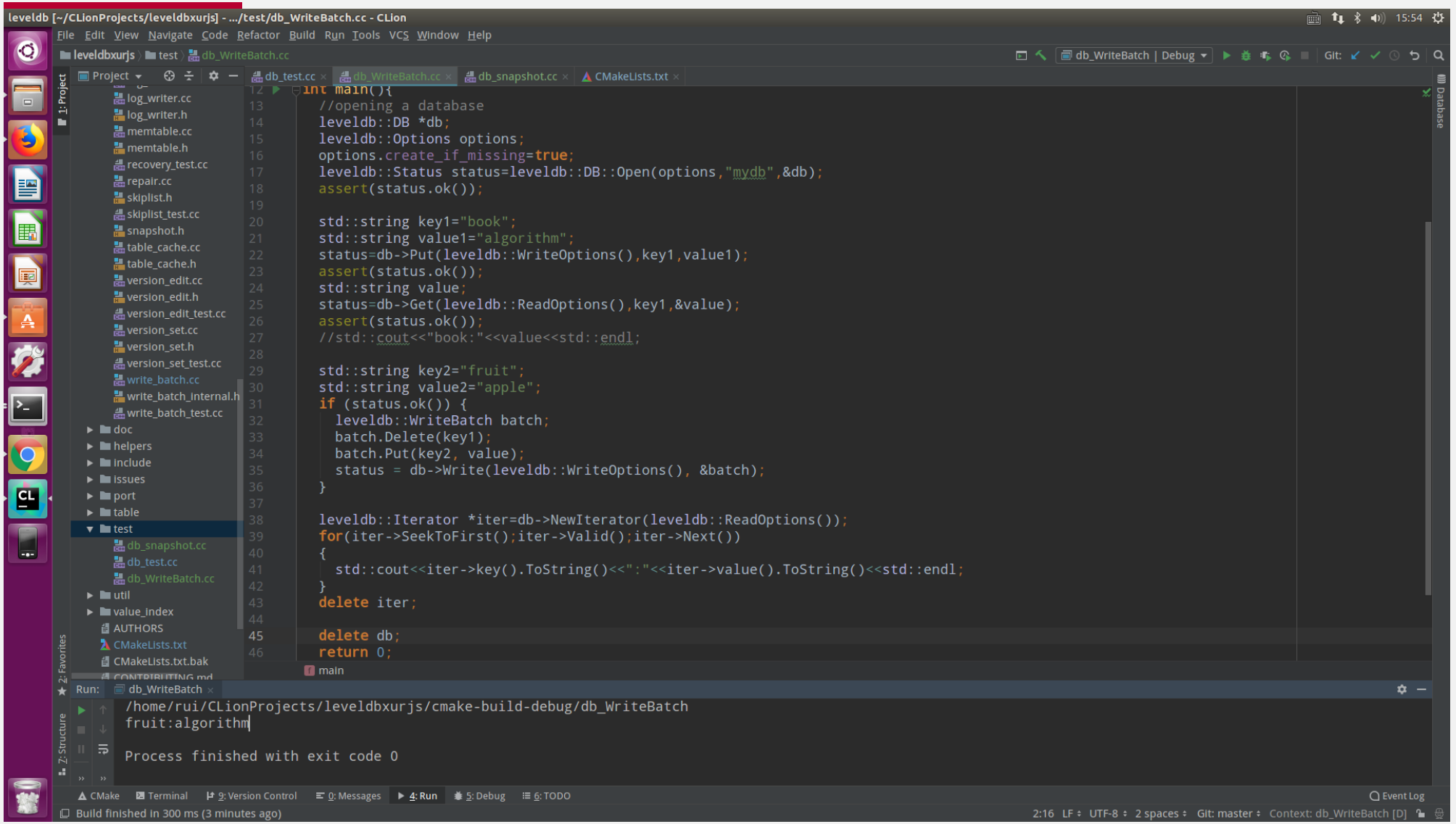
03 接口练习

DaSE
Data Science
&Engineering

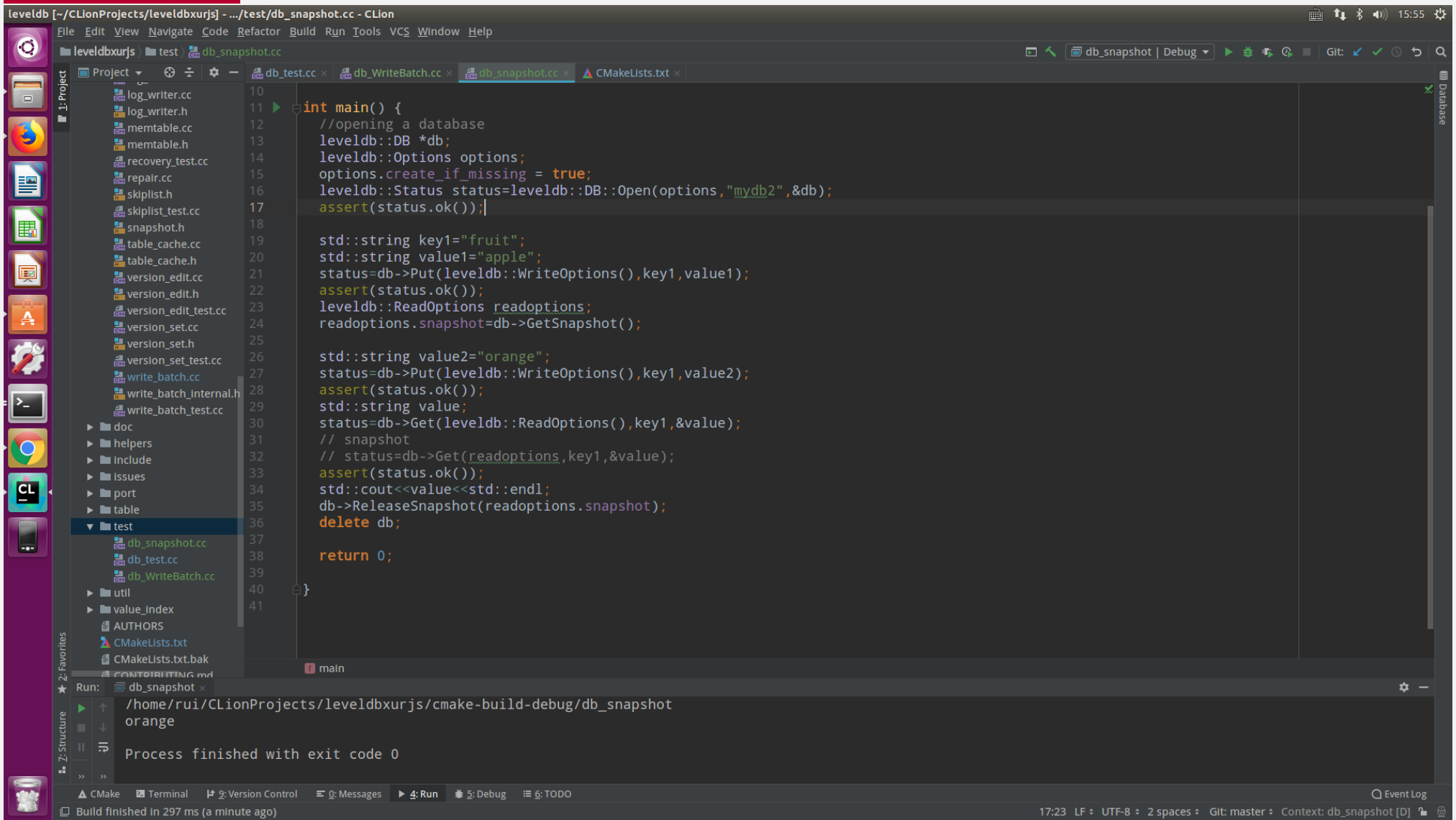
PutGet

```cpp
#include "leveldb/db.h"
#include <cstdio>
#include <iostream>

using namespace std;
using namespace leveldb;

int main() {
  //opening a database
  leveldb::DB* db;
  leveldb::Options options;
  options.create_if_missing = true;
  leveldb::Status status = leveldb::DB::Open(options, "testdb", &db);
  assert(status.ok());

  std::string key1="book";
  std::string value1="algorithm";
  std::string value;
  db->Put(WriteOptions(),key1,value1);
  db->Get(ReadOptions(),key1,&value);
  std::cout<<"key:"<<key1<<",value:"<<value<<std::endl;


  delete db;

  return 0;
}
```

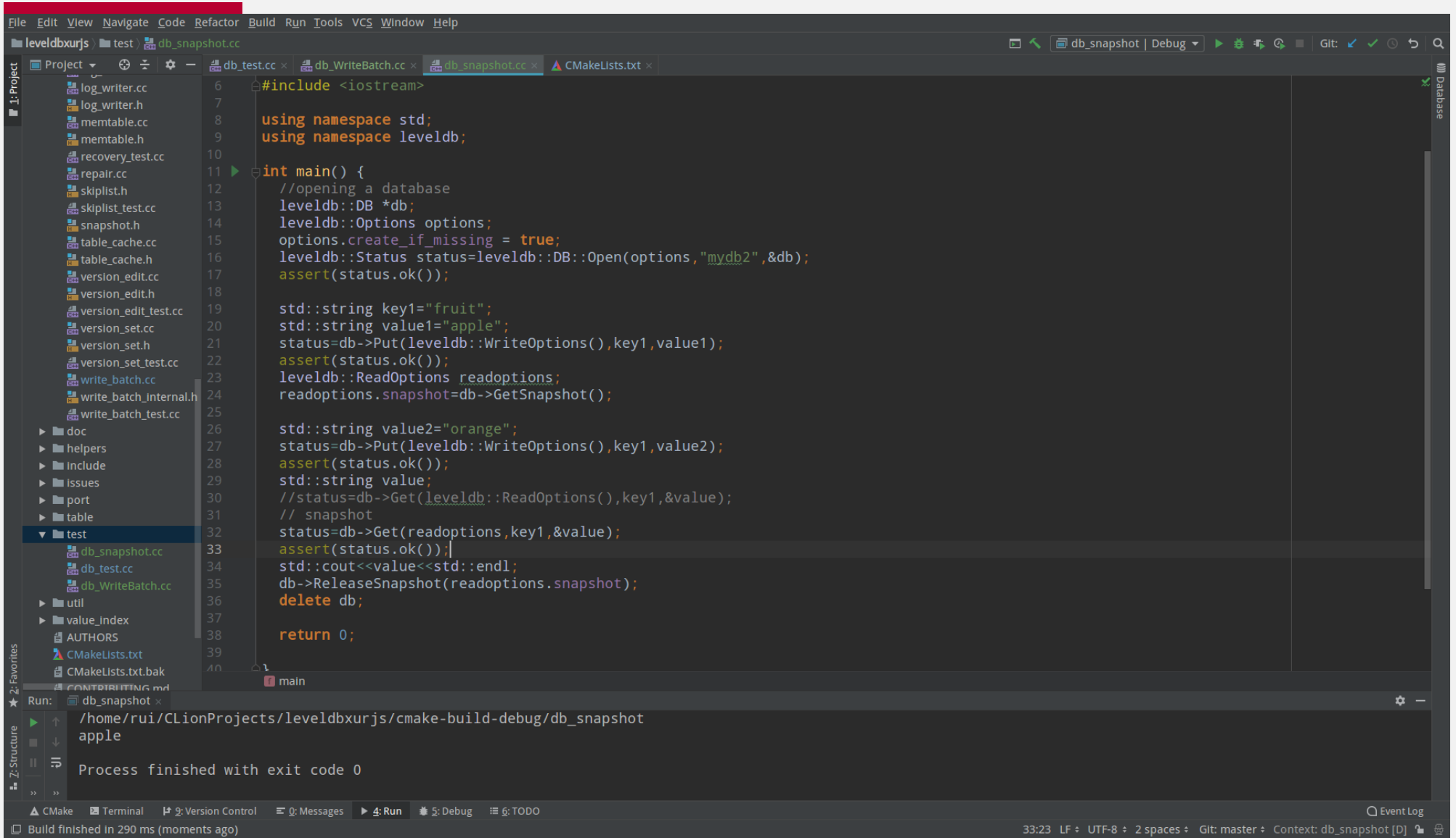
WriteBatch


leveldb [~/CLionProjects/leveldbxurjs] - .../test/db_WriteBatch.cc - CLion
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
leveldbxurjs test db_WriteBatch.cc
db_WriteBatch | Debug
Git:
Project
db_test.cc
db_WriteBatch.cc
db_snapshot.cc
CMakeLists.txt
log_writer.cc
log_writer.h
memtable.cc
memtable.h
recovery_test.cc
repair.cc
skiplist.h
skiplist_test.cc
snapshot.h
table_cache.cc
table_cache.h
version_edit.cc
version_edit.h
version_edit_test.cc
version_set.cc
version_set.h
version_set_test.cc
write_batch.cc
write_batch_internal.h
write_batch_test.cc
doc
helpers
include
issues
port
table
test
db_snapshot.cc
db_test.cc
db_WriteBatch.cc
util
value_index
AUTHORS
CMakeLists.txt
CMakeLists.txt.bak
int main(){
//opening a database
leveldb::DB *db;
leveldb::Options options;
options.create_if_missing=true;
leveldb::Status status=leveldb::DB::Open(options,"mydb",&db);
assert(status.ok());
std::string key1="book";
std::string value1="algorithm";
status=db->Put(leveldb::WriteOptions(),key1,value1);
assert(status.ok());
std::string value;
status=db->Get(leveldb::ReadOptions(),key1,&value);
assert(status.ok());
//std::cout<<"book:"<<value<<std::endl;
std::string key2="fruit";
std::string value2="apple";
if (status.ok()) {
leveldb::WriteBatch batch;
batch.Delete(key1);
batch.Put(key2, value);
status = db->Write(leveldb::WriteOptions(), &batch);
}
leveldb::Iterator *iter=db->NewIterator(leveldb::ReadOptions());
for(iter->SeekToFirst();iter->Valid();iter->Next())
{
std::cout<<iter->key().ToString()<<":"<<iter->value().ToString()<<std::endl;
}
delete iter;
delete db;
return 0;
main
Run: db_WriteBatch
/home/rui/CLionProjects/leveldbxurjs/cmake-build-debug/db_WriteBatch
fruit:algorithm
Process finished with exit code 0
CMake
Terminal
9: Version Control
0: Messages
4: Run
5: Debug
6: TODO
Event Log
Build finished in 300 ms (3 minutes ago)
2:16 LF UTF-8 2 spaces Git: master Context: db_WriteBatch [D]

Snapshot


leveldb [~/CLionProjects/leveldbxurjs] - .../test/db_snapshot.cc - CLion
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
db_test.cc
db_WriteBatch.cc
db_snapshot.cc
CMakeLists.txt
int main() {
//opening a database
leveldb::DB *db;
leveldb::Options options;
options.create_if_missing = true;
leveldb::Status status=leveldb::DB::Open(options,"mydb2",&db);
assert(status.ok());
std::string key1="fruit";
std::string value1="apple";
status=db->Put(leveldb::WriteOptions(),key1,value1);
assert(status.ok());
leveldb::ReadOptions readoptions;
readoptions.snapshot=db->GetSnapshot();
std::string value2="orange";
status=db->Put(leveldb::WriteOptions(),key1,value2);
assert(status.ok());
std::string value;
status=db->Get(leveldb::ReadOptions(),key1,&value);
// snapshot
// status=db->Get(readoptions,key1,&value);
assert(status.ok());
std::cout<<value<<std::endl;
db->ReleaseSnapshot(readoptions.snapshot);
delete db;
return 0;
}
Run: db_snapshot
/home/rui/CLionProjects/leveldbxurjs/cmake-build-debug/db_snapshot
orange
Process finished with exit code 0
Build finished in 297 ms (a minute ago)

Snapshot


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
leveldbxurjs test db_snapshot.cc
db_snapshot | Debug
Git:
Project
db_test.cc
db_WriteBatch.cc
db_snapshot.cc
CMakeLists.txt
log_writer.cc
log_writer.h
memtable.cc
memtable.h
recovery_test.cc
repair.cc
skiplist.h
skiplist_test.cc
snapshot.h
table_cache.cc
table_cache.h
version_edit.cc
version_edit.h
version_edit_test.cc
version_set.cc
version_set.h
version_set_test.cc
write_batch.cc
write_batch_internal.h
write_batch_test.cc
doc
helpers
include
issues
port
table
test
db_snapshot.cc
db_test.cc
db_WriteBatch.cc
util
value_index
AUTHORS
CMakeLists.txt
CMakeLists.txt.bak
#include <iostream>
using namespace std;
using namespace leveldb;
int main() {
//opening a database
leveldb::DB *db;
leveldb::Options options;
options.create_if_missing = true;
leveldb::Status status=leveldb::DB::Open(options,"mydb2",&db);
assert(status.ok());
std::string key1="fruit";
std::string value1="apple";
status=db->Put(leveldb::WriteOptions(),key1,value1);
assert(status.ok());
leveldb::ReadOptions readoptions;
readoptions.snapshot=db->GetSnapshot();
std::string value2="orange";
status=db->Put(leveldb::WriteOptions(),key1,value2);
assert(status.ok());
std::string value;
//status=db->Get(leveldb::ReadOptions(),key1,&value);
// snapshot
status=db->Get(readoptions,key1,&value);
assert(status.ok());
std::cout<<value<<std::endl;
db->ReleaseSnapshot(readoptions.snapshot);
delete db;
return 0;
}
main
Run: db_snapshot
/home/rui/CLionProjects/leveldbxurjs/cmake-build-debug/db_snapshot
apple
Process finished with exit code 0
CMake Terminal 9: Version Control 0: Messages 4: Run 5: Debug 6: TODO Event Log
Build finished in 290 ms (moments ago)
33:23 LF UTF-8 2 spaces Git: master Context: db_snapshot [D]

资料

https://zh-google-styleguide.readthedocs.io/en/latest/google-cpp-styleguide/

DaSE
Data Science
&Engineering