如果只获取特定操作系统的本地驱动，则使用如：

<dependencies>

<dependency>

<groupId>org.fusesource.leveldbjni</groupId>

<artifactId>leveldbjni-linux64</artifactId>

<version>1.8</version>

</dependency>

</dependencies>

如果本地已经有DLL了，则只需要依赖Java启动。

<dependency>

<groupId>org.fusesource.leveldbjni</groupId>

<artifactId>leveldbjni</artifactId>

<version>1.8</version>

</dependency>

一个完全独立的另一个项目，是LevelDB的Java实现，其接口与LevelDB JNI一致。

<https://github.com/dain/leveldb>

API参考：

<https://github.com/fusesource/leveldbjni>

Using a memory pool to make native memory allocations more efficient:

JniDBFactory.pushMemoryPool(1024 \* 512);

try {

// .. work with the DB in here,

} finally {

JniDBFactory.popMemoryPool();

}

# RocksDB

## RocksDB由C++语言编写

<https://github.com/facebook/rocksdb>

A library that provides an embeddable, persistent key-value store for fast storage.

Facebook开发和维护，建立在LevelDB工作的基础上。

RocksDB可用作快速Key-Value服务器的核心构建块，特别适合在Flash驱动上存数据。

使用了LSM树设计，灵活平衡WAF（Write-Amplification-Factor）、RAF和SAF。

Compact是多线程的，使得特别适合在单个DB上存储数TB的数据。

## RocksDB JNI

<https://github.com/fusesource/rocksdbjni>

A Java JNI driver to rocksdb.

<dependency>  
 <groupId>org.rocksdb</groupId>  
 <artifactId>rocksdbjni</artifactId>  
 <version>5.15.10</version>  
</dependency>

提供的操作API与LevelDB兼容，通过指定的类和方法可以转换为RocksDB的。但不同版本略有不同，不细叙。

*\*/*public class User {  
 static {  
 RocksDB.*loadLibrary*();  
 }  
  
 public static void main(String[] args) {  
 Options options = new Options();  
 options.setCreateIfMissing(true);  
 options.setCompressionType(CompressionType.*NO\_COMPRESSION*);  
 options.setBottommostCompressionType(CompressionType.*ZSTD\_COMPRESSION*);  
 options.setLevelCompactionDynamicLevelBytes(true);  
 options.setMaxOpenFiles(32);  
 options.setIncreaseParallelism(1);  
 options.useFixedLengthPrefixExtractor(16);  
  
 BlockBasedTableConfig blockBasedTableConfig = new BlockBasedTableConfig();  
 blockBasedTableConfig.setBlockSize(16 \* 1024);  
 blockBasedTableConfig.setBlockCacheSize(32 \* 1024 \* 1024);  
 blockBasedTableConfig.setCacheIndexAndFilterBlocks(true);  
 blockBasedTableConfig.setPinL0FilterAndIndexBlocksInCache(true);  
 blockBasedTableConfig.setFilter(new BloomFilter(10, false));  
 options.setTableFormatConfig(blockBasedTableConfig);  
  
 RocksDB db = null;  
 try {  
 db = RocksDB.*open*(options, "C:\\Users\\chenjw\\Desktop\\MyRocksDB");  
  
  
 db.put("db".getBytes(StandardCharsets.*UTF\_8*), "rocksdb".getBytes(StandardCharsets.*UTF\_8*));  
 System.*out*.println("db: " + new String(db.get("db".getBytes(StandardCharsets.*UTF\_8*)), StandardCharsets.*UTF\_8*));  
 } catch (RocksDBException e) {  
 e.printStackTrace();  
 } finally {  
 if (db != null) {  
 db.close();  
 }  
 }  
 }  
  
 private static void backup(RocksDB db) {  
 BackupableDBOptions backupableDBOptions = new BackupableDBOptions("C:\\Users\\chenjw\\Desktop\\MyRocksDBBackUp");  
 BackupEngine backupEngine = null;  
 try {  
 backupEngine = BackupEngine.*open*(Env.*getDefault*(), backupableDBOptions);  
 backupEngine.createNewBackup(db, true);  
 } catch (RocksDBException e) {  
 e.printStackTrace();  
 } finally {  
 if (backupEngine != null) {  
 backupEngine.close();  
 }  
 backupableDBOptions.close();  
 }  
 }  
  
 private static void bulkRead(RocksDB db) {  
 RocksIterator iterator = db.newIterator();  
 try {  
 for (iterator.seekToFirst(); iterator.isValid(); iterator.next()) {  
 System.*out*.println(new String(iterator.key(), StandardCharsets.*UTF\_8*));  
 }  
 } finally {  
 iterator.close();  
 }  
 }  
  
 private byte[] prefixLoopup(RocksDB db, byte[] prefixBytes) {  
 ReadOptions readOpts = null;  
 try {  
 readOpts = new ReadOptions();  
 readOpts.setPrefixSameAsStart(true);  
 readOpts.setVerifyChecksums(false);  
  
 RocksIterator it = db.newIterator(readOpts);  
 it.seek(prefixBytes);  
 if (it.isValid()) {  
 return it.value();  
 }  
 } finally {  
 if (readOpts != null) {  
 readOpts.close();  
 }  
 }  
 return null;  
 }  
  
 private static void batchWrite(RocksDB db) {  
 WriteBatch writeBatch = new WriteBatch();  
 WriteOptions writeOptions = new WriteOptions();  
 try {  
 writeBatch.delete("db".getBytes(StandardCharsets.*UTF\_8*));  
 writeBatch.put("db".getBytes(StandardCharsets.*UTF\_8*), "rocksdb".getBytes(StandardCharsets.*UTF\_8*));  
 db.write(writeOptions, writeBatch);  
 } catch (RocksDBException e) {  
 e.printStackTrace();  
 } finally {  
 writeOptions.close();  
 writeBatch.clear();  
 }  
 }  
}

运行结果：





