|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | |  |
| **RocksDB 6.10.2** | | |  |
| **安装指南(openEuler 22.03 LTS)** | | |
| **文档版本** | **01** | |
| **发布日期** | **2022-06-30** | |
| 华为网格系统---方案4-032.png | | | | |
|  | 华为技术有限公司 | | 附件1-16K |  |

|  |
| --- |
| 版权所有 © 华为技术有限公司2022。 保留一切权利。  非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。  商标声明  附件3-版权声明页图 和其他华为商标均为华为技术有限公司的商标。  本文档提及的其他所有商标或注册商标，由各自的所有人拥有。  注意  您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。  由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。 |

|  |  |
| --- | --- |
| 华为技术有限公司 | |
| 地址： | 深圳市龙岗区坂田华为总部办公楼 邮编：518129 |
| 网址： | <https://www.huawei.com> |
| 客户服务邮箱： | [support@huawei.com](mailto:support@huawei.com) |
| 客户服务电话： | 4008302118 |

|  |
| --- |
| **RocksDB 6.10.2** |
| **安装指南(openEuler 22.03 LTS)** |

# 介绍

简要介绍

RocksDB 是一个来自Facebook 的可嵌入的支持持久化的 key-value存储系统，也可作为 C/S 模式下的存储数据库。RocksDB 基于LevelDB 构建。

关于RocksDB的更多信息请访问[RocksDB官网](http://rocksdb.org/" \o " )。

**语言**：C++

**一句话描述**：用于快速存储的嵌入式持久键值存储。

**开源协议**：Apache+GPLv2

建议版本

建议使用版本为“RocksDB 6.10.2”。

系统版本为

|  |  |
| --- | --- |
| openEuler | * OS:   22.03 for ARM   * Kernel:   4.19.90 |

# 源码编译安装和卸载

## 配置安装环境

### yum安装依赖包

执行以下命令安装依赖包。

**yum install -y vim gcc-c++ snappy snappy-devel zlib zlib-devel bzip2 bzip2-devel lz4 lz4-devel**

----结束

### 源码安装zstd

操作步骤

进入“/home”目录。

**cd /home**

获取zstd源码包并修改包名。

**wget https://codeload.github.com/facebook/zstd/tar.gz/refs/tags/v1.1.3**

**mv v1.1.3 zstd-1.1.3.tar.gz**

解压源码包。

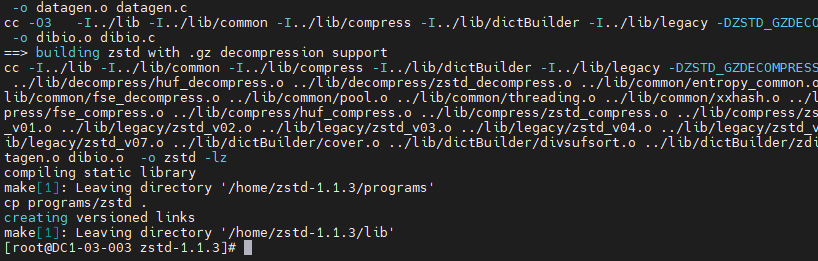
**tar -zxvf zstd-1.1.3.tar.gz**

进入解压目录。

**cd zstd-1.1.3/**

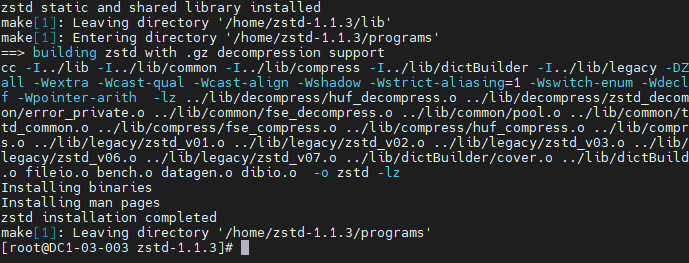
编译。

**make -j8**



安装。

**make install**



环境变量配置。

**export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/usr/local/lib**

**export CPLUS\_INCLUDE\_PATH=$CPLUS\_INCLUDE\_PATH:/usr/local/include/**

----结束

### 源码安装cmake



cmake版本最低要求为3.5.1，建议安装3.10.0版本。

操作步骤

进入“/home”目录。

**cd /home**

获取源码。

**wget https://github.com/Kitware/CMake/releases/download/v3.10.0/cmake-3.10.0.tar.gz**

解压源码包。

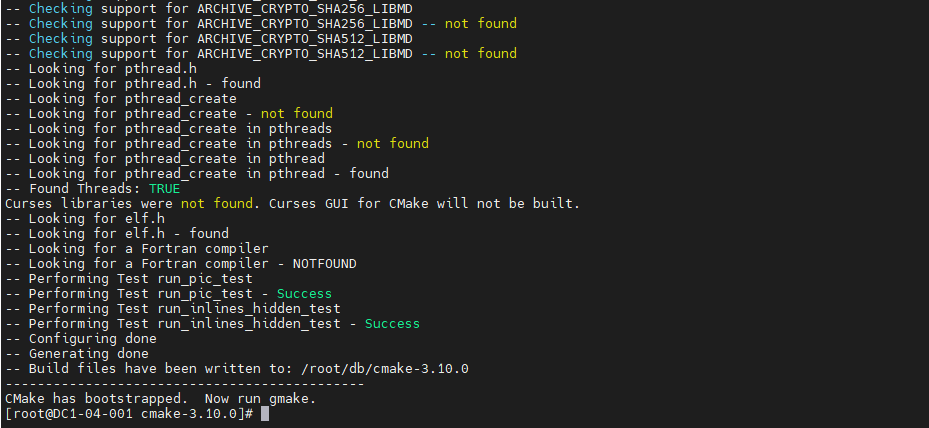
**tar -zxvf cmake-3.10.0.tar.gz**

进入解压包。

**cd cmake-3.10.0**

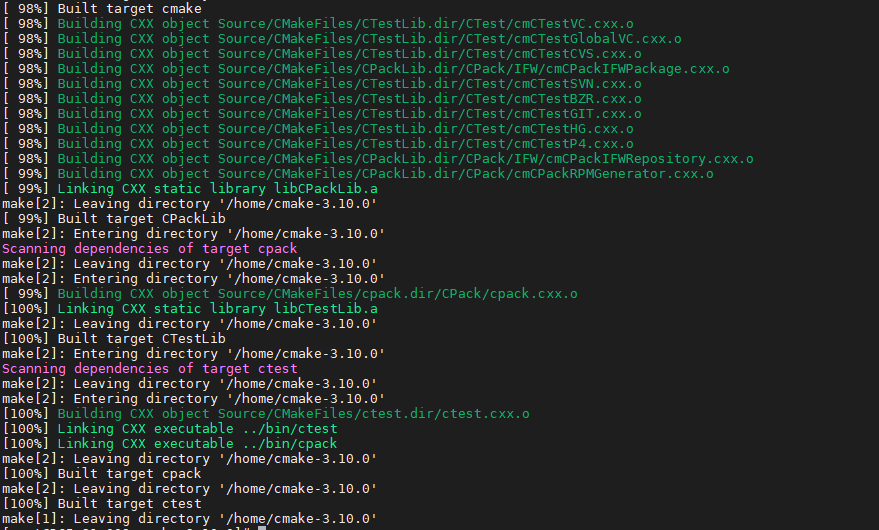
编译配置。

**./bootstrap --prefix=/usr/local/cmake**



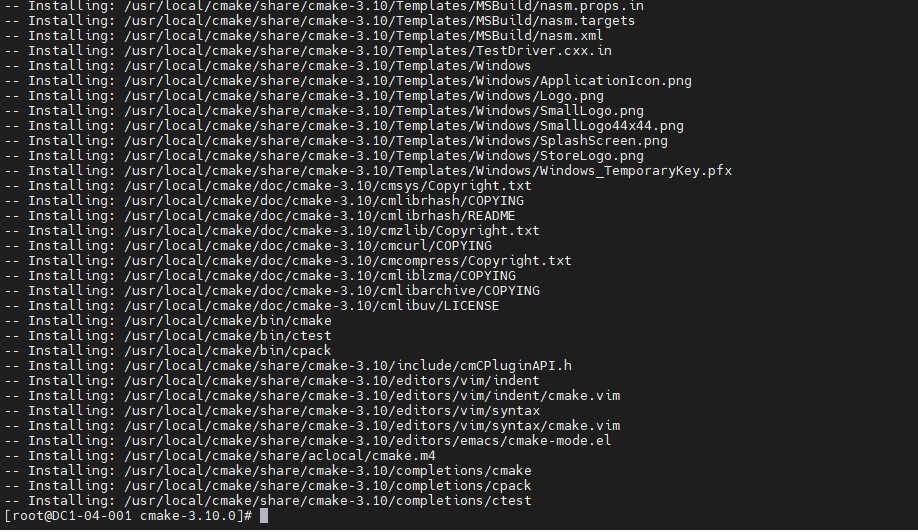
编译。

**make -j8**



安装。

**make install**

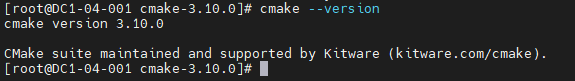


执行以下命令设置cmake的环境变量。

**export PATH=$PATH:/usr/local/cmake/bin/**

查看版本验证是否安装成功。

**cmake --version**



如果查看的版本不是3.10.0，参考[5.1章节](#_cmake 版本未生效)。

----结束

### 源码安装gflags

操作步骤

进入“/home”目录。

**cd /home**

获取源码包并修改包名。

**wget https://github.com/gflags/gflags/archive/v2.2.2.tar.gz**

**mv v2.2.2.tar.gz gflags-2.2.2.tar.gz**

解压源码包。

**tar -zxvf gflags-2.2.2.tar.gz**

进入解压包。

**cd gflags-2.2.2**

新建目录。

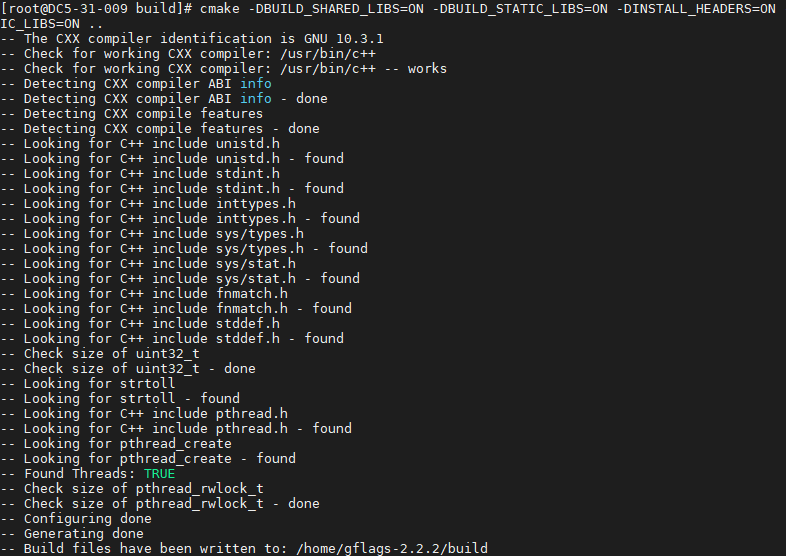
**mkdir build/**

进入新建目录。

**cd build/**

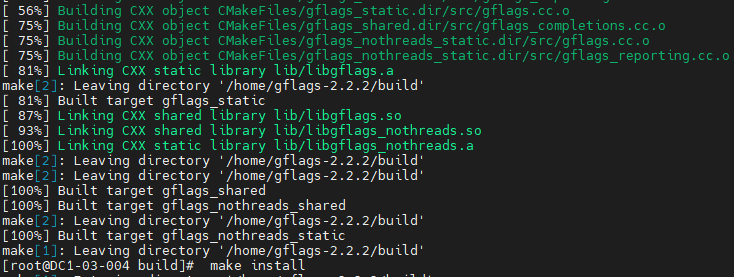
编译配置。

**cmake -DBUILD\_SHARED\_LIBS=ON -DBUILD\_STATIC\_LIBS=ON -DINSTALL\_HEADERS=ON -DINSTALL\_SHARED\_LIBS=ON -DINSTALL\_STATIC\_LIBS=ON ..**



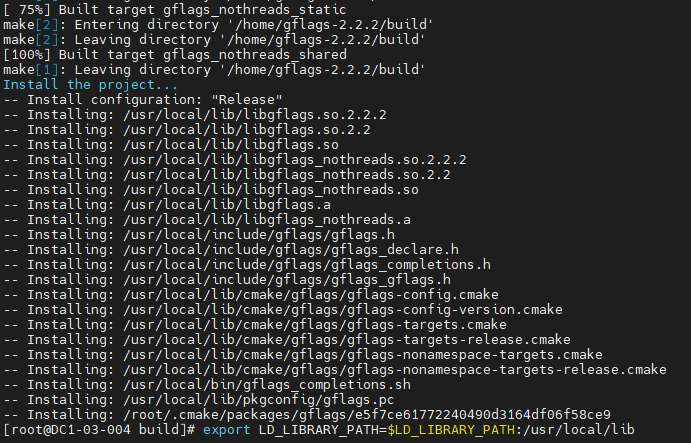
编译。

**make -j8**



安装。

**make install**



执行以下命令设置gflags的环境变量。

**export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/usr/local/lib**

**export CPLUS\_INCLUDE\_PATH=$CPLUS\_INCLUDE\_PATH:/usr/local/include/gflags**

----结束

## 源码编译安装



获取源码的方式有两种，文档中默认第一种方式。

1. 若服务器可以访问网络，则可以直接在服务器上使用wget命令下载源码。
2. 若服务器无法访问外网，则可以通过浏览器下载源码包，并上传至服务器“/home”目录。
3. 源码下载地址参考[第2章节环境要求](#_环境要求)部分。

操作步骤

进入“/home”目录。

**cd /home/**

获取源码包并修改包名。

**wget** [**https://github.com/facebook/rocksdb/archive/refs/tags/v6.10.2.tar.gz**](https://github.com/facebook/rocksdb/archive/refs/tags/v6.10.2.tar.gz)

**mv v6.10.2.tar.gz rocksdb-6.10.2.tar.gz**

解压RocksDB安装包。

**tar -zxvf rocksdb-6.10.2.tar.gz**

进入解压目录。

**cd rocksdb-6.10.2/**

修改cmake编译配置文件。

**vim CMakeLists.txt**



按照源码包中的配置文件进行编译，会在生成benchmark工具时，出现所需静态库找不到的错 误。需修改CMakeList.txt配置文件可解决该问题。具体操作如下。

在1186行添加如下代码段：

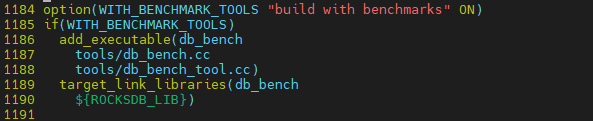
**add\_subdirectory(third-party/gtest-1.8.1/fused-src/gtest)**

**add\_library(testharness STATIC**

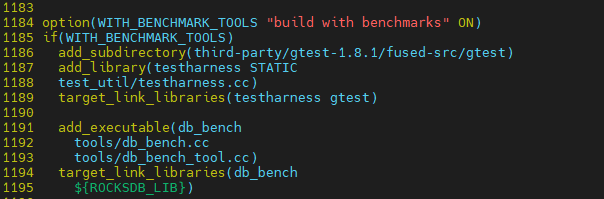
**test\_util/testharness.cc)**

**target\_link\_libraries(testharness gtest)**

编辑前：



编辑后：



创建目录。

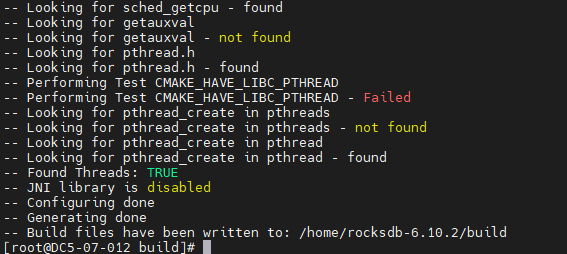
**mkdir build**

进入创建的目录。

**cd build**

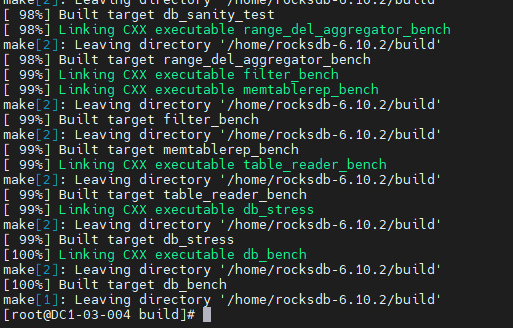
编译配置。

**cmake -DCMAKE\_INSTALL\_PREFIX=/usr/local/rocksdb -DWITH\_SNAPPY=1 -DWITH\_ZLIB=1 -DWITH\_LZ4=1 -DWITH\_ZSTD=1 -DWITH\_BZ2=1 ..**



执行编译命令。

**make -j8**

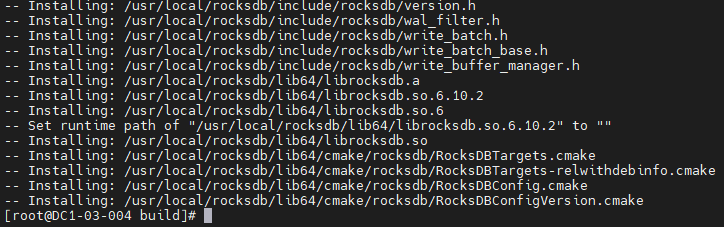




如果编译报错提示“Cannot allocate memory”，请参考[5.2章节](#_编译rocksdb提示Cannot allocate memory)。

执行安装命令。

**make install**



将工具移动到安装目录。

**cp ../utilities/merge\_operators.h /usr/local/rocksdb/include/rocksdb/utilities/**

**mkdir -p /usr/local/rocksdb/tools**

**cp -p ./tools/ldb /usr/local/rocksdb/tools**

**cp -p ./tools/sst\_dump /usr/local/rocksdb/tools**

查看安装目录。

**ls /usr/local/rocksdb**



环境变量设置。

**export CPLUS\_INCLUDE\_PATH=$CPLUS\_INCLUDE\_PATH:/usr/local/rocksdb/include/**

**export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/usr/local/rocksdb/lib64/**

**export LIBRARY\_PATH=$LIBRARY\_PATH:/usr/local/rocksdb/lib64/**

**export PATH=$PATH:/usr/local/rocksdb/tools**

----结束

## 接口**Get、Put、Delete、Merge**验证



1. 通过ldb工具验证put、get、delete三个接口的正常操作。
2. 通过代码简单验证merge接口的正常操作。

### put接口

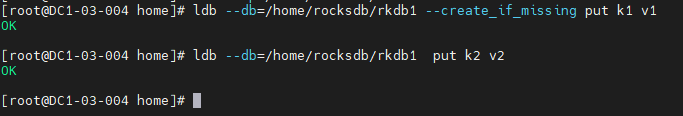
创建数据库存储目录。

**mkdir -p /home/rocksdb/rkdb1**

添加数据。

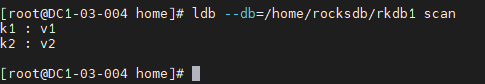
**ldb --db=/home/rocksdb/rkdb1 --create\_if\_missing put k1 v1**

**ldb --db=/home/rocksdb/rkdb1 put k2 v2**



查看结果。

**ldb --db=/home/rocksdb/rkdb1 scan**



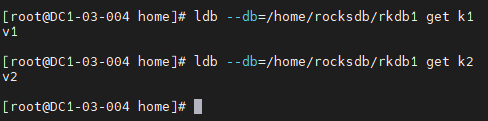
----结束

### Get接口

读取数据。

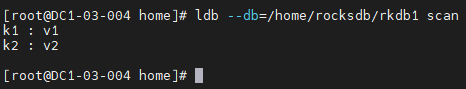
**ldb --db=/home/rocksdb/rkdb1 get k1**

**ldb --db=/home/rocksdb/rkdb1 get k2**



查看结果。

**ldb --db=/home/rocksdb/rkdb1 scan**



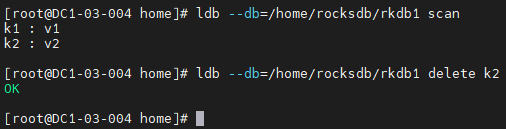
----结束

### Delete接口

删除数据。

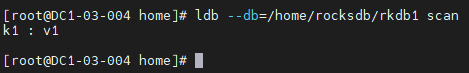
**ldb --db=/home/rocksdb/rkdb1 scan**

**ldb --db=/home/rocksdb/rkdb1 delete k2**



查看结果。

**ldb --db=/home/rocksdb/rkdb1 scan**



----结束

### Merge接口

进入“/home”目录。

**cd /home/**

创建如下源代码。

**vim rocksdb\_test\_merge.cpp**

添加代码内容。

**#include <iostream>**

**#include <vector>**

**#include <rocksdb/db.h>**

**#include <rocksdb/options.h>**

**#include <rocksdb/table.h>**

**#include <rocksdb/merge\_operator.h>**

**#include <rocksdb/filter\_policy.h>**

**#include <rocksdb/perf\_context.h>**

**#include <rocksdb/iostats\_context.h>**

**#include <rocksdb/trace\_reader\_writer.h>**

**#include "rocksdb/utilities/merge\_operators.h"**

**using namespace rocksdb;**

**using namespace std;**

**int main()**

**{**

**DB\* db;**

**Options options;**

**options.create\_if\_missing = true;**

**//no compression**

**options.compression = kNoCompression;**

**BlockBasedTableOptions table\_options;**

**table\_options.no\_block\_cache = true;**

**table\_options.cache\_index\_and\_filter\_blocks = false;**

**options.table\_factory.reset(NewBlockBasedTableFactory(table\_options));**

**options.merge\_operator = MergeOperators::CreateStringAppendOperator();**

**Status s=rocksdb::DB:: Open(options,"/home/rocksdb/rkdb2",&db);**

**if(!s.ok()){**

**cout<<"open rocksdb failed:"<<s.ToString()<<endl;**

**delete db;**

**exit(-1);**

**}**

**cout<<"open rocksdb success!"<<endl;**

**//Merge operator**

**int j = 0;**

**string key = to\_string(j);**

**string value;**

**char buf[8];**

**//EncodeFixed64(buf,2);**

**s = db->Merge(WriteOptions(),key,"1");**

**//need change**

**if(!s.ok()){**

**cout << "Merge value failed:" << s.ToString()<<endl;**

**delete db;**

**exit(-1);**

**}**

**s = db->Merge(WriteOptions(),key,"2");**

**db->Flush(FlushOptions());**

**if(!s.ok()){**

**cout << "Merge value failed:" << s.ToString()<<endl;**

**delete db;**

**exit(-1);**

**}**

**s = db->Get(ReadOptions(),key,&value);**

**if(!s.ok()){**

**cout << "Merge value failed:" << s.ToString()<<endl;**

**delete db;**

**exit(-1);**

**}**

**cout << "Get merge value: " << value << "\n value of size:" << value.size() << endl;**

**}**

创建数据库目录。

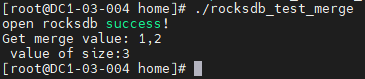
**mkdir -p /home/rocksdb/rkdb2**

执行以下命令编译代码。

**g++ -o rocksdb\_test\_merge --std=c++11 rocksdb\_test\_merge.cpp -lrocksdb -ldl**

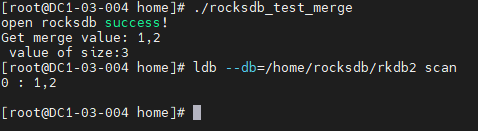
执行可执行文件。

**./rocksdb\_test\_merge**



查询库中的数据。

**ldb --db=/home/rocksdb/rkdb2 scan**



----结束

# 故障排除

[5.1 cmake 版本未生效](#_cmake 版本未生效" \o " )

[5.2 编译rocksdb提示Cannot allocate memory](#_编译rocksdb提示Cannot allocate memory" \o " )

## cmake 版本未生效

问题现象描述

cmake-3.10.0编译安装成功后，查看版本号，该版本未生效。显示为其他版本号。

关键过程、根本原因分析

环境中已存在其它cmake版本，优先级比编译安装的cmake-3.10.0高。

结论、解决方案及效果

查看环境中所有的cmake。

find / -name cmake | grep bin

查看cmake版本。

/usr/bin/cmake --version

/usr/local/cmake/bin/cmake --version

让cmake-3.10.0生效。

cd /usr/bin/   
mv cmake cmake-3.12.1   
hash -r   
cmake --version

----结束

## 编译rocksdb提示Cannot allocate memory

问题现象描述

编译过程中报错：Cannot allocate memory

关键过程、根本原因分析

执行make -j编译会使用所有cpu核数，导致内存不够。

结论、解决方案及效果

重新编译配置。

rm -rf \*   
cmake -DCMAKE\_INSTALL\_PREFIX=/usr/local/rocksdb -DWITH\_SNAPPY=1 -DWITH\_ZLIB=1 -DWITH\_LZ4=1 -DWITH\_ZSTD=1 -DWITH\_BZ2=1 ..

查看cpu核数。

lscpu

再次执行编译命令（减少cpu核数）。

make -j 8

----结束

1. 修订记录

| 发布日期 | 修订记录 |
| --- | --- |
| 2022-06-30 | 第一次正式发布 |