交叉编译

任务目标：

将直接编译结果是x86_64-linux的应用编译到riscv-linux上

目标应用：**LMDB**

LMDB（Lightning Memory-Mapped Database）是一个轻量级、高性能的键值存储库，支持事务和多线程，适用于嵌入式设备和高性能服务器等场景。

嵌入式内存数据库：嵌入式内存数据库是指可以在嵌入式设备中运行的内存数据库，通常具有轻量级、高性能、低功耗、易于集成等特点。

C/S内存数据库：Redis等

实现过程：

1. 预先构建riscv交叉编译工具链，并将交叉编译工具链添加到环境变量：

   export PATH=$PATH:/opt/riscv/bin/

   (编译安装交叉编译工具链时：./configure --prefix=/opt/riscv后，再make linux)

2. 下载 LMDB 源码

   wget https://github.com/LMDB/lmdb/archive/refs/tags/LMDB_0.9.29.tar.gz

   tar -zxvf LMDB_0.9.29.tar.gz

   mv lmdb-LMDB_0.9.29 lmdb # 将lmdb-LMDB_0.9.29目录重命名为lmdb

   cd lmdb/libraries/liblmdb

3. 配置交叉编译选项(修改Makefile文件)

```
CC       = /opt/riscv/bin/riscv64-unknown-linux-gnu-gcc
AR       = ar
W        = -W -Wall -Wno-unused-parameter -Wbad-function-cast -Wuninitialized

THREADS = -pthread
OPT = -O2 -g
CFLAGS  = -static
LDLIBS  = -lpthreada
```

a. `CC` 指定交叉编译器，`CFLAGS` 指定交叉编译选项，`LDLIBS` 用于指定链接时需要的库的名称。
b. 由于项目使用了pthread库，这个库的riscv版本在riscv交叉编译工具链的目录下的sysroot/lib目录下(在安装交叉编译工具链指定--prefix=/opt/riscv后只需要在/opt/riscv/sysroot/lib/下)

4. 添加环境变量(export PATH=$PATH:/opt/riscv/bin/)，在目录lmdb/libraries/liblmdb下执行:

```
./configure  --prefix=/opt/lmdb
```

```
make
```

```
sudo make install
```

在**x86**下验证

1. 测试程序

```
#include <stdio.h>
#include <string.h>
#include <lmdb.h>

int main(int argc, char* argv[]){
    int rc;
    MDB_env *env;
    MDB_dbi dbi;
    MDB_val key, data;
    MDB_txn *txn;
    MDB_cursor *cursor;
    char strKey[50];
    char strValue[50];
    printf("lmdb version:%s\n",mdb_version(0, 0, 0));
    rc = mdb_env_create(&env);
    if(rc){
        printf("mdb_env_create error,detail:%s\n",
mdb_strerror(rc));
        return -1;
    }
//打开数据库，如果目录为空，将在该目录内初始化一个数据库
rc = mdb_env_open(env, "./", 0, 0644);
/*if(rc){
    printf("mdb_env_open error,detail:%s\n",
mdb_strerror(rc));
    return -1;
```

```
}*/

rc = mdb_txn_begin(env, NULL, 0, &txn);
if(rc){
    printf("mdb_txn_begin error,detail:%s\n",
mdb_strerror(rc));
    return -1;
}

rc = mdb_dbi_open(txn, NULL, 0, &dbi);
if(rc){
    printf("mdb_dbi_open error,detail:%s\n",
mdb_strerror(rc));
    return -1;
}

sprintf(strKey, "%s", "key");
sprintf(strValue, "%s", "value");
key.mv_size = strlen(strKey)*sizeof(char);
key.mv_data = strKey;
data.mv_size = strlen(strValue)*sizeof(char);
data.mv_data = strValue;

//写入key-value数据
rc = mdb_put(txn, dbi, &key, &data, 0);

//提交
rc = mdb_txn_commit(txn);
if (rc) {
    fprintf(stderr, "mdb_txn_commit: (%d) %s\n", rc,
mdb_strerror(rc));
    goto leave;
}

rc = mdb_txn_begin(env, NULL, MDB_RDONLY, &txn);
rc = mdb_cursor_open(txn, dbi, &cursor);

//遍历数据库中所有key-value数据
while ((rc = mdb_cursor_get(cursor, &key, &data, MDB_NEXT))
== 0) {
    memset(strKey, 0, sizeof(strKey));
    memset(strValue, 0, sizeof(strValue));
```

```
    strncpy(strKey, (const char*)key.mv_data,
(int)key.mv_size);
    strncpy(strValue, (const char*)data.mv_data,
(int)data.mv_size);
    printf("key:%s, value:%s\n",strKey, strValue);
}

mdb_cursor_close(cursor);
mdb_txn_abort(txn);
leave:
    mdb_dbi_close(env, dbi);
    mdb_env_close(env);
    return 0;
}
```

2. 编译指令

   `/opt/riscv/bin/riscv64-unknown-linux-gnu-gcc test.cpp -o test -I/opt/rv64_lmdb/include -L/opt/rv64_lmdb/lib -llmdb -lpthread --static`

3. 执行

```
lmdb version:LMDB 0.9.29: (March 16, 2021)
key:key, value:value
```

4. 可能存在的错误

```
mdb_env_open(env, "./testdb", MDB_FIXEDMAP , 0664): No such file or directory
```

   原因mdb_env_open()函数指定的目录不存在，因为该函数第二个参数是指定数据库的目录不带名字

5. 在gem5/RISCV上测试失败

```
lmdb version:LMDB 0.9.29: (March 16, 2021)
mdb_env_open error,detail:Operation not permitted
Exiting @ tick 7065500 because exiting with last active thread context
```

6.