Linux安装配置zookeeper

1:查看是否安装java环境:java -version

2:安装java环境:sudo apt-get install openjdk-8-jdk

3:解压:tar -zxvf

**双端编译环境安装**

一：ubuntu安装cmake

查看是否安装:sudo cmake –version

安装cmake: sudo apt-get install -y cmake

二:安装libevent,openssl,zlib

win64:

1.解压zlib-1.2.13.tar,打开vsx64命令输入框,进入解压后目录,创建mybuild,进入mybuild目录,执行 cmake .. 得到zlib.sln,打开编译x64即可

2.安装Strawberry Perl(​ ​<https://www.perl.org/get.html>​​),安装nasm(<https://nasm.us/>​),将系统变量的Path添加Strawberry Perl和nasm路径,解压openssl-1.1.1s.tar,打开vsx64命令输入框,执行

Perl Configure VC-WIN64A –prefix=”D:\openssl”，执行nmake clean， 执行nmake，执行nmake install即可

3:解压libevent-2.1.12-stable.tar,打开vsx64命令输入框,进入解压后目录,创建mybuild目录,执行cmake .. -DCMAKE\_BUILD\_TYPE=Debug -DOPENSSL\_ROOT\_DIR=D:\ -DOPENSSL\_INCLUDE\_DIR=D:\ -DOPENSSL\_LIBRARIES=D:\ -DZLIB\_LIBRARY=D:\ -DZLIB\_INCLUDE\_DIR=D:\ -DZLIB\_INCLUDE\_DIRS=D:\ -DZLIB\_LIBRARIES=D:\ 得到libevent.sln，打开编译x64即可

linux:

1. tar -zxvf zlib-1.2.13.tar.gz

./configure -prefix=/usr/local/zlib

make&make install

1. openssl ubuntu自带版本
2. tar -zxvf libevent-2.1.12-stable.tar.gz

cd libevent-2.1.12-stable

./autogen.sh

Sudo apt install libssl\_dev

./configure -prefix=/usr/local/libevent

make&make install

三:安装jemalloc(windows测试版本,只需安装linux版本)

tar -zxvf jemalloc-5.3.0.tar.gz

cd jemalloc-5.3.0

./configure -prefix=/usr/local/jemalloc

make&make install

四:安装zookeeper(zookeep-3.5.4-beta.tar,用非apache开头版本)

win64:

1.解压zookeep-3.5.4-beta.tar,打开vsx64命令输入框,进入解压后目录,进入src/c目录,先将CMakeLists中WANT\_SYNCAPI改成OFF,再将代码中strock\_r改成strock\_s,localtime\_r改成localtime\_s传参位置互换,创建mybuild,进入mybuild目录,执行 cmake .. 得到zookeeper.sln,打开编译x64即可

五:安装mongo

win64:

1. 解压mongo-c-driver-1.21.2.tar, 打开vsx64命令输入框,进入mongo-c-driver-1.21.2目录,mkdir mybuild目录,进入mybuild目录,运行cmake -DCMAKE\_PREFIX\_PATH=install生成目录 .. ,得到mongo-c-driver.sln,打开编译x64,注意这里不但要编译ALL\_BUILD还有编译INSTALL
2. 解压boost\_1\_79\_0.7z, 打开vsx64命令输入框,进入boost\_1\_79\_0目录,执行bootstrap.bat,然后mkdir mybuild创建目录,执行b2 install –prefix=mybuild绝对路径(库生成路径),即可
3. 解压mongo-cxx-driver-r3.6.7.tar,打开vs64命令输入框,进去mongo-cxx-driver-r3.6.7,mkdir mybuild,进入mybuild目录,

cmake “-DBOOST\_ROOT=boost路径” "-DCMAKE\_PREFIX\_PATH=mongo-c-dirvier INSTALL生成路径" ..

linux:

1: 解压mongo-c-dirver: tar -zxvf mongo-c-driver-1.21.2.tar.gz

进入目录: cd mongo-c-driver-1.21.2/

创建编译目录: mkdir mybuild

进入目录: cd mybuild

执行cmake: cmake .. -DCMAKE\_BUILD\_TYPE=Release -DCMAKE\_INSTALL\_PREFIX=/usr/local/mongo-c-driver

编译安装: make -j8&make install

2: 解压boost\_1\_79\_0.7z: 7z x boost\_1\_79\_0.7z

进入目录: cd boost\_1\_79\_0/

运行脚本: ./bootstrap.sh –with-libraries=all –with-toolset=gcc

编译代码: ./b2 toolset=gcc

安装路径: ./b2 install –prefix=/usr/local/boost

3: 解压mongo-cxx-driver-r3.6.7.tar.gz: tar -zxvf mongo-cxx-driver-r3.6.7.tar.gz

进入目录: cd mongo-cxx-driver-r3.6.7/

创建编译目录: mkdir mybuild

进入创建目录: cd mybuild

cmake: cmake -DBOOST\_ROOT=/root/Duncan/lib/boost\_1\_65\_1 -DCMAKE\_PREFIX\_PATH=/usr/local/mongo-c-driver -DCMAKE\_BUILD\_TYPE=Release -DCMAKE\_INSTALL\_PREFIX=/usr/local/mongo-cxx-driver -DBSONCXX\_POLY\_USE\_BOOST=1 ..(让stdx::optional用宏控制调用boost库)

编译代码: make -j8

安装:make install

4:ubuntu安装mongodb

六:安装curl

win64:

1.解压curl-8.1.2.tar,打开vsx64命令输入框,进入解压后curl-8.1.2目录,创建mybuild,进入mybuild目录,执行 cmake .. 得到CURL.sln,打开编译x64即可

linux:

1. 解压curl-8.1.2.tar: tar -zxvf curl-8.1.2.tar.gz
2. 进入目录: cd curl-8.1.2/
3. 配置路径: ./configure –prefix=/usr/local/curl –with-openssl
4. make -j8& make install

七:安装mysql&mysql++

win64:

1.解压mysql-connector-c-6.1.11-src.tar,打开vsx64命令输入框,进入解压后mysql-connector-c-6.1.11-src目录,(注意这里cmake/install\_macros.cmake文件第365行有错误,需要将365行ENDIF和ENDFOREACH位置互换)创建mybuild,进入mybuild目录,执行 cmake .. 得到libMySQL.sln,打开编译x64即可

2.首先要完成1,然后解压mysql++-3.3.0.tar,进入解压后mysql++-3.3.0,打开vc2008,运行mysql++.sln(注意需要包含mysql-connector-c-6.1.11-src下面的头文件和链接库),编译mysqlpp即可

linux:

1. 解压: tar -zxvf mysql-connector-c-6.1.11-src.gz.tar
2. 进入目录: cd mysql-connector-c-6.1.11/
3. 创建编译目录: mkdir mybuild
4. 进入编译目录: cd mybuild
5. 修复cmake错误: vim ../cmake/install\_macros.cmake将365和366行互换
6. cmake: cmake -DCMAKE\_PREFIX\_PATH=/usr/local/mysql ..
7. 编译: make -j8&make install

八:安装protobuf

win64:

1: 解压protobuf-master,打开vsx64命令输入框,进入解压后protobuf-master目录,再进入cmake目录,创建mybuild,进入mybuild目录,执行 cmake .. 得到protobuf.sln,打开编译x64即可

(注意port\_indef.inc里面的max宏会和mongo申明冲突,暂时注释处理)

linux:

1:解压: unzip protobuf-master.zip

2:进入目录:cd protobuf-master

3:进入cmake目录:cd cmake

4:cmake . -Dprotobuf\_BUILD\_TESTS=OFF

5:make&make install DESTDIR=/usr/local/protobuf

九:安装json

win64:

1:解压jsoncpp-master.zip, 开vsx64命令输入框,进入解压后jsoncpp-master目录,创建mybuild,进入mybuild目录,执行 cmake -DCMAKE\_INSTALL\_PREFIX=./ .. 得到jsoncpp.sln,打开编译x64即可

九:安装behaviac

1: 解压behaviac-master,打开vsx64命令输入框,进入解压后behaviac-master目录,再进入cmake文件目录,创建mybuild,进入mybuild目录,执行 cmake .. 得到ALL\_BUILD.sln,打开编译x64即可(注意降低警告等级和生成.lib静态库设置)

**Python环境安装**

Windows:

1: python解释器安装<https://www.python.org/downloads/>

2: 安装第三方包:

pip install flake8

pip install yapf

jango的坑:

高版本jango和低版本mysql:

打开python311/lib/site\_packagres/django/db/backends/base/base.py注释代码self.check\_database\_version\_supported

orm生成数据表:

python manage.py makemigrations

python manage.py migrate

python manage.py createsuperuser