Ubuntu环境配置

一、关闭自动更新：

1.界面操作

系统设置-->软件和更新-->更新，将“自动检查更新”和“有新版本时通知我”设置为“从不“，关闭对话框完成设置。



2.修改配置文件

修改配置文件/etc/apt/apt.conf.d/10periodic。

将“1”改为“0”，保存即可。

vi /etc/apt/apt.conf.d/10periodic

APT::Periodic::Update-Package-Lists "０";

APT::Periodic::Download-Upgradeable-Packages "0";

APT::Periodic::AutocleanInterval "0";

二、安装sshd

1.Ubuntu主机安装ssh相关服务

openssh-client

openssh-server

2方法：

sudo apt-get install openssh-client openssh-server

检测：

ps -e | grep sshd

异常：当执行上述指令后未发现sshd服务在运行，可尝试如下命令

sudo service ssh restart 或

sudo /etc/init.d/ssh restart

# 三、Ubuntu 更换源

使用Ubuntu 的apt-get来安装软件是总是因为官方源的速度太慢而抓狂。

#### 1）备份

sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak

#### 2）修改

sudo vim /etc/apt/sources.list

将source.list文件内容替换成下面的(xenial对应的是16.04版本的)

# 默认注释了源码镜像以提高 apt update 速度，如有需要可自行取消注释

deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial main restricted universe multiverse

# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial main restricted universe multiverse

deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-updates main restricted universe multiverse

# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-updates main restricted universe multiverse

deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-backports main restricted universe multiverse

# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-backports main restricted universe multiverse

deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-security main restricted universe multiverse

# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-security main restricted universe multiverse

# 预发布软件源，不建议启用

# deb https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-proposed main restricted universe multiverse

# deb-src https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ xenial-proposed main restricted universe multiverse

#### 3）更新

sudo apt-get update

四、下载源码编译CMake

1.到cmake官网下载最新的cmake

<https://cmake.org/download/>

2.下载后解压,然后进入目录执行:

./bootstrap

make -j8

sudo make install

3.验证版本

cmake --version

cmake version 3.9.0

五、降级安装gcc/g++版本为4.9.x

(1). 下载gcc/g++ 4.9.x

$ sudo apt-get install -y gcc-4.9

$ sudo apt-get install -y g++-4.9

(2). 链接gcc/g++实现降级

$ cd /usr/bin

$ sudo rm gcc

$ sudo ln -s gcc-4.9 gcc

$ sudo rm g++

$ sudo ln -s g++-4.9 g++

完成，降级之后的版本为4.9.4。

检查版本可在终端输入gcc -v查看。

# 六、ubuntu16.04 + OpenCv 2.4.13安装

## 安装包下载

<https://pan.baidu.com/s/1_H5tYfC0iwyOWZJAIX6-NA>

### 下载安装一些一类包 以及编译工具

安装编译工具

sudo apt-get install build-essential

安装依赖包

sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev

安装可选包

sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev

### 下载opencv

https://github.com/Itseez/opencv/archive/2.4.13.zip

### 进入下载目录解压

### 编译安装

打开文件夹"opencv-2.4.13"：

cd opencv-2.4.13

新建一个文件夹用于存放临时文件：

mkdir release

切换到该临时文件夹：

cd release

开始编译：

cmake -D CMAKE\_BUILD\_TYPE=RELEASE -D CMAKE\_INSTALL\_PREFIX=/usr/local ..

make -j4 *//开启线程 按照自己的配置*

sudo make install

### 相关配置

配置环境

将opencv的库加入到路径，从而让系统可以找到

sudo gedit /etc/ld.so.conf.d/opencv.conf

末尾加入/usr/local/lib，保存退出

sudo ldconfig 使配置生效

sudo gedit /etc/bash.bashrc

末尾加入

PKG\_CONFIG\_PATH=$PKG\_CONFIG\_PATH:/usr/local/lib/pkgconfig

export PKG\_CONFIG\_PATH

保存退出

sudo source /etc/bash.bashrc *#使配置生效*

（该步骤可能会报错找不到命令，原因是source为root命令

su（进入root权限）

输入密码

source /etc/bash.bashrc

Ctrl+d（推迟root）

sudo updatedb *#更新database*

### 测试实例

*//文件名字Dis.cpp*

#include <stdio.h>

#include <opencv2/opencv.hpp>

using namespace cv;

int main( )

{

Mat image;

image = imread("/home/elijah/lena.jpg", 1 );*//目录按照自己的目录*

if ( !image.data )

{

printf("No image data \n");

return -1;

}

namedWindow("Display Image", WINDOW\_AUTOSIZE );

imshow("Display Image", image);

waitKey(0);

return 0;

}

*运行g*++ *Dis*.*cpp* -*o* *Dis*.*o* *`pkg*-*config* --*cflags* --*libs* *opencv`*

七、kcov的安装和使用方法

1.Kcov安装

a)   下载kcov源代码：

<https://github.com/SimonKagstrom/kcov/archive/v30.tar.gz>

b)   安装依赖包：

sudo apt-get install libcurl4-nss-dev binutils-dev libdw-dev

c)   mkdir build && cd build

d)   cmake .. && make

e)   sudo make install

2. Kcov使用：

a) g++ -g（需要为debug模式，release模式无法测试覆盖率）

b) LD\_LIBRARY\_PATH=XXX kcov --include-path=include,src cov\_report test.exe

覆盖率报告在cov\_report下

# 八、Linux下编译安装GCC 4.9.4

# 下载GCC

<ftp://ftp.gnu.org/gnu/gcc/gcc-4.9.4/gcc-4.9.4.tar.gz> 官方FTP  
[http://ftpmirror.gnu.org](http://ftpmirror.gnu.org/) 自动选择速度较快的镜像站

# 下载依赖包

* [GMP](https://gmplib.org/)
* [MPC](http://www.mpfr.org/)
* [MPFR](http://www.multiprecision.org/)

可通过两种方式下载：

* 自动下载，运行GCC源码目录内置脚本：
* # cd gcc-4.9.4
* # sh ./contrib/download\_prerequisites
* 手动下载、解压，然后执行命令：
* # cd gcc-4.9.4
* # ln -sf path/to/gmp-x.x.x gmp
* # ln -sf path/to/mpc-x.x.x mpc
* # ln -sf path/to/mpfr-x.x.x mpfr

这样，编译GCC同时自动构建上述3个库。

注意要保证3个包的源码目录是干净的，否则可能报错，必要时可执行make distclean

版本要求参考脚本./contrib/download\_prerequisites，以4.9.4为例

# Necessary to build GCC.

MPFR=mpfr-2.4.2

GMP=gmp-4.3.2

MPC=mpc-0.8.1

# 编译安装

官方文档强烈建议在源码目录之外，新建一个临时目录，用于编译；

个人建议设置安装目录，便于GCC多版本管理；

configure选项，可以根据需求自行配置。

# mkdir build-gcc-4.9.4

# cd build-gcc-4.9.4

# ../gcc-4.9.4/configure --prefix=/usr/local/gcc-4.9.4/ --enable-checking=release --enable-languages=c,c++ --disable-multilib

# make -j4

# make install

# 设置环境变量

优先使用指定版本的GCC，可将下行代码添加到~/.bash\_profile

export PATH=/usr/local/gcc-4.9.4/bin:$PATH

# 配置libstdc++.so.6

在gcc编译目录下找到最新的libstdc++.so，拷贝至/usr/lib64/，运行命令ldconfig

# ll /usr/lib64/libstdc++.so.6

lrwxrwxrwx. 1 root root 19 Nov 20 15:59 /usr/lib64/libstdc++.so.6 -> libstdc++.so.6.0.13

# find build-gcc-4.9.4/ -name libstdc++.so.\* |xargs ls -l

lrwxrwxrwx. 1 root root 19 Aug 30 2016 build-gcc-4.9.4/prev-x86\_64-unknown-linux-gnu/libstdc++-v3/src/.libs/libstdc++.so.6 -> libstdc++.so.6.0.20

-rwxr-xr-x. 1 root root 6781772 Aug 30 2016 build-gcc-4.9.4/prev-x86\_64-unknown-linux-gnu/libstdc++-v3/src/.libs/libstdc++.so.6.0.20

lrwxrwxrwx. 1 root root 19 Aug 30 2016 build-gcc-4.9.4/stage1-x86\_64-unknown-linux-gnu/libstdc++-v3/src/.libs/libstdc++.so.6 -> libstdc++.so.6.0.20

-rwxr-xr-x. 1 root root 6781772 Aug 30 2016 build-gcc-4.9.4/stage1-x86\_64-unknown-linux-gnu/libstdc++-v3/src/.libs/libstdc++.so.6.0.20

lrwxrwxrwx. 1 root root 19 Aug 30 2016 build-gcc-4.9.4/x86\_64-unknown-linux-gnu/libstdc++-v3/src/.libs/libstdc++.so.6 -> libstdc++.so.6.0.20

-rwxr-xr-x. 1 root root 6781772 Aug 30 2016 build-gcc-4.9.4/x86\_64-unknown-linux-gnu/libstdc++-v3/src/.libs/libstdc++.so.6.0.20

# cp build-gcc-4.9.4/x86\_64-unknown-linux-gnu/libstdc++-v3/src/.libs/libstdc++.so.6.0.20 /usr/lib64/

# ldconfig

# ll /usr/lib64/libstdc++.so.6

lrwxrwxrwx. 1 root root 19 Nov 20 16:12 /usr/lib64/libstdc++.so.6 -> libstdc++.so.6.0.20

好了，开车上路吧

# 九、linux下编译安装glibc

Glibc 包含了linux一些主要的C库，用于分配内存、搜索目录、打开关闭文件、读写文件、字串处理、模式匹配、数学计算等。

centos默认的glibc版本比较低，不支持一些新的应用，所以对glibc进行更新，发现编译glibc报一个错误：

Makeconfig:42: \*\*\* missing separator. Stop

看了glibc文档发现，glibc不能在源代码目录编译，建议在源码目录上层新建一个目录glibc-build，用于编译

$ wget http://ftp.gnu.org/gnu/glibc/glibc-2.19.tar.gz

$ tar -zxf glibc-2.19.tar.gz

$ mkdir glibc-build

$ cd glibc-build

$ ../glibc-2.19/configure --prefix=/usr/lib64/glibc-2.19

$ make

$ make install