ubuntu基操

基本操作

基本操作
- 更改apt源
- pip相关操作
- 安装python
- zsh
- 编译opencv
- 编译open3d
- CUDA安装 (reference ymj)
- virtualenvs虚拟环境配置
- github proxy
- ubuntu 更换内核
- py2so
- ubuntu、windows启动盘
- python程序打包
- 搭建c++环境
- 再生龙使用
- 配置c++环境
- gitlab runner
- 终端代理
- VS markdown 配置
- 计算机启动
- Ubuntu启动盘

更改apt源

▼ 展开详情

- 法1:简易

直接去系统设置里面改，具体操作： system setting⇒SoftWare & Updates ⇒ Ubuntu Software ⇒ Download From: mirrors.aliyun.com

- 法2:修改系统文件

备份系统自带源

```
mv /etc/apt/sources.list /etc/apt/sources.list.bak
```

修改/etc/apt/sources.list文件

```
deb-src http://archive.ubuntu.com/ubuntu xenial main restricted #Added by software-properties
deb http://mirrors.aliyun.com/ubuntu/ xenial main restricted
deb-src http://mirrors.aliyun.com/ubuntu/ xenial main restricted multiverse universe #Added by software-properties
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates main restricted
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-updates main restricted multiverse universe #Added by software-properties
deb http://mirrors.aliyun.com/ubuntu/ xenial universe
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates universe
deb http://mirrors.aliyun.com/ubuntu/ xenial multiverse
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates multiverse
deb http://mirrors.aliyun.com/ubuntu/ xenial-backports main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-backports main restricted universe multiverse #Added by software-propertie
deb http://archive.canonical.com/ubuntu xenial partner
deb-src http://archive.canonical.com/ubuntu xenial partner
deb http://mirrors.aliyun.com/ubuntu/ xenial-security main restricted
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-security main restricted multiverse universe #Added by software-properties
deb http://mirrors.aliyun.com/ubuntu/ xenial-security universe
deb http://mirrors.aliyun.com/ubuntu/ xenial-security multiverse```
```

pip相关操作

▼ 展开详情

pip安装

官方文档

```
sudo python get-pip.py
```

get-pip.py: offical github local

如果安装失败，出现internet connect error之类的错误，请在确认联网的情况下用 sudo apt install python-pip python3-pip 安装pip

pip换源

在 ~/.pip/pip.conf (若没有则新建) 下更改pip源为国内源

```
[global]
trusted-host = mirrors.aliyun.com
index-url = http://mirrors.aliyun.com/pypi/simple
```

工程依赖配置

```
pip freeze > requirements.txt #获取当前环境下的pip包及其版本导入到txt文件pip install -r requirement.txt #根据requirement.txt的内容配置
```

Installing from local packages In some cases, you may want to install from local packages only, with no traffic to PyPI.

First, download the archives that fulfill your requirements:

```
pip download --destination-directory DIR -r requirements.txt
```

Note that pip download will look in your wheel cache first, before trying to download from PyPI. If you've never installed your requirements before, you won't have a wheel cache for those items. In that case, if some of your requirements don't come as wheels from PyPI, and you want wheels, then run this instead:

```
pip wheel --wheel-dir DIR -r requirements.txt
```

Then, to install from local only, you'll be using –find-links and –no-index like so:

```
pip install --no-index --find-links=DIR -r requirements.txt
```

通常工程下面会有一个extras文件夹，存放1)依赖安装包、2)requirement.txt、3)安装脚本，脚本示例如下

```
#!/bin/bashCUR_DIR=`dirname $0`pip3 install --no-index --find-links=$CUR_DIR/dependence -r $CUR_DIR/requirement.txt --upgra
```

切换至新工程的虚拟环境后，若无，则根据虚拟环境配置创建新的虚拟环境，一键安装依赖，快速实现工程基本配置。

安装python

▼ 展开详情

安装python 3.6.4

```
tar xf Python-3.6.4.tar.xzcd Python-3.6.4./configure --disable-shared --enable-loadable-sqlite-extensionsmakesudo make inst
```

zsh

▼ 展开详情

- 1、安装:

```
sudo apt install zsh
```

- 2、配置文件，实际上是从git clone一个项目到根目录

```
sh -c "$(curl -fsSL https://raw.github.com/robbyrussell/oh-my-zsh/master/tools/install.sh)"
```

若连接不上网络，使用本地文件zsh_install.sh，根据以下命令执行。

```
sudo chmod +x xxx.sh
bash xxx.sh
```

- 3、修改默认bash终端为zsh:

```
chsh -s /bin/zsh
```

重启后默认终端就是zsh。

- 4、更多配置 查看zsh配置文件，根据zsh主题设置

```
sudo subl ~/.zshrc
```

- 5、plugins插件

内置git插件，文件路径：~/.oh-my-zsh/plugins/git/git.plugin.zsh, 修改后需要

```
source ~/.zshrc
```

生效

- 6、卸载

```
sudo sh -c "$(curl -fsSL https://raw.github.com/robbyrussell/oh-my-zsh/master/tools/uninstall.sh)"
```

```
chsh -s /bin/bash
```

编译opencv

▼ 展开详情

源码下载: github:opencv github:opencv_contrib

将文件放置如下结构

opencv_src ├── opencv └── opencv_contrib(-3.4.0)

源码编译opencv⇒python接口cv2.xxx.so

```
tar xf vision_tool_opencv.tar.gz
cd vision_tool_opencv/opencv
mkdir build
cd build
```

最主要的是对cmake的配置，需要更为关注python的各项配置路径!

▼ cmake config

```
# cmake配置1
cmake .. \
  -DCMAKE_BUILD_TYPE=Release \
  -DBUILD_SHARED_LIBS=OFF \
  -DENABLE_PIC=ON \
  -DBUILD_JAVA=OFF \
  -DBUILD_opencv_python2=ON \
  -DBUILD_opencv_python3=ON \
  -DBUILD_PERF_TESTS=OFF \
  -DBUILD_TESTS=OFF \
  -DINSTALL_TESTS=OFF \
  -DWITH_MATLAB=OFF \
  -DOPENCV_ENABLE_NONFREE=ON \
  -DENABLE_FAST_MATH=ON \
  -DOPENCV_EXTRA_MODULES_PATH="../../opencv_contrib-3.4.0/modules"\
  -DWITH_LIBV4L=ON \
  -DWITH_V4L=ON \
  -DWITH_IPP=ON \
  -DWITH_TIFF=OFF \
  -DCMAKE_C_FLAGS="-fPIC" \
  -DCMAKE_CXX_FLAGS="-fPIC" \
  -DWITH_TBB=ON \
  -DWITH_CUDA=OFF \
  -DWITH_BIG_ENDIAN=OFF #若该代码，有时候需要打开cmake注释掉大小端测试的检测代码
# 以上步骤也可以使用cmake-gui操作，会有一个用户交互界面，选择合适的依赖包。
# cmake 配置2 opencv3.4.0 4.2.0 已验证
cmake ..\
  -DBUILD_TIFF=ON \
  -DBUILD_opencv_java=OFF \
  -DBUILD_opencv_python3=ON \
  -DWITH_CUDA=OFF \
  -DWITH_OPENGL=ON \
  -DWITH_OPENCL=ON \
  -DWITH_IPP=ON \
  -DWITH_TBB=ON \
  -DWITH_EIGEN=ON \
  -DWITH_V4L=ON \
  -DBUILD_TESTS=OFF \
  -DBUILD_PERF_TESTS=OFF \
  -DCMAKE_BUILD_TYPE=RELEASE \
  -DBUILD_NEW_PYTHON_SUPPORT=ON \
  -DBUILD_PYTHON_SUPPORT=ON \
  -DCMAKE_INSTALL_PREFIX=/usr/local \
  -DPYTHON3_EXECUTABLE=$(which python3.6) \
  -DPYTHON3_INCLUDE_DIR=$(python3.6 -c "from distutils.sysconfig import get_python_inc; print(get_python_inc())") \
  -DPYTHON3_PACKAGES_PATH=$(python3.6 -c "from distutils.sysconfig import get_python_lib; print(get_python_lib())")
```

==========⇒查看cmake的输出信息，检查config后python的路径是否正确,python路径配置错误会导致无法在python的包目录下产生cv2*.so文件。

```
Python 3:
--     Interpreter:                 /usr/local/bin/python3 (ver 3.6.4)
--     Libraries:                   /usr/local/lib/libpython3.6m.a (ver 3.6.4)
--     numpy:                       /usr/local/lib/python3.6/site-packages/numpy/core/include (ver 1.17.2)
--     packages path:               /usr/local/lib/python3.6/site-packages
--
--   Python (for build):            /usr/local/bin/python3
```

```
make -j4
```

在make结束后可以看到这样一句话就说明你的so已经生成了:[100%] Linking CXX shared module ../../lib/python3/cv2.cpython-36m-x86_64-linux-gnu.so 即在source/build/lib/python3下可以看到供python3调用的cv2*.so

```
sudo make install
find /usr/local/lib/ -type f -name "cv2*.so" #查看编译生成的so文件在哪里
ln -s /usr/local/lib/python3.6/site-packages/cv2.cpython-36m-x86_64-linux-gnu.so ~/.virtualenvs/visiontool/lib/python3.6/
#创建软连接，或者直接复制.so文件到虚拟环境的文件夹
#opencv 4.+会多一层目录，在/usr/local/lib/python3.6/site-packages/cv2/python3.6/cv2*.so
```

验证

```
>>> cv2.__file__
'/usr/local/lib/python3.6/site-packages/cv2.cpython-36m-x86_64-linux-gnu.so'
>>> cv2.__version__
'3.4.0'
```

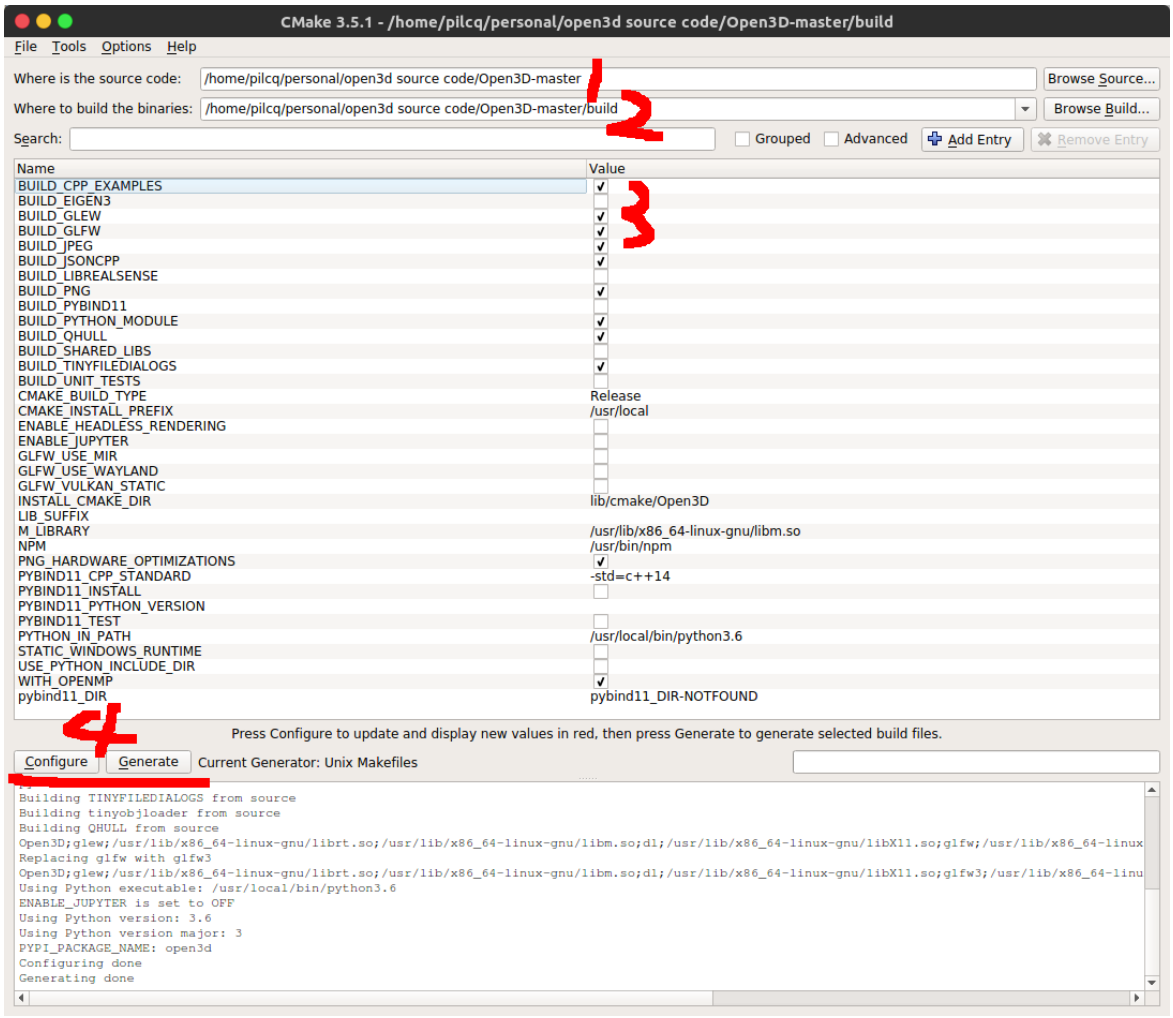
公司电脑编译4.2.0ok，且也会产生.so Python 3: Interpreter: /usr/local/bin/python3 (ver 3.6.4) Libraries: /usr/local/lib/libpython3.6m.a (ver 3.6.4) numpy: /usr/local/lib/python3.6/site-packages/numpy/core/include (ver 1.17.2) install path: /usr/local/lib/python3.6/site-packages/cv2/python-3.6

编译open3d

▼ 展开详情

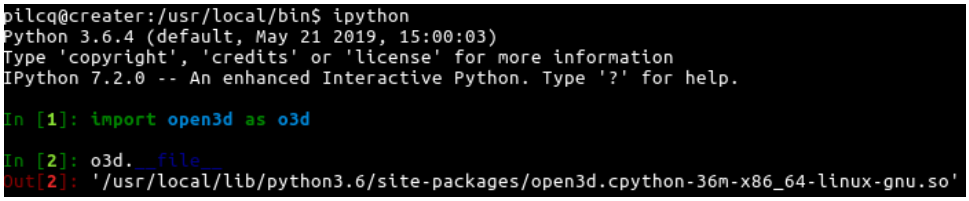
由于pip安装的open3d不能*tab*出函数名与查看函数细节，采用源码编译的方法。

```
git clone https://github.com/intel-isl/Open3D.git #拉取源码cd Open3D #切换工作目录"""补充拓展库，由于Open3D/3rdpart/下有许多拓展库，拉
```



cmake-open3d.png

```
"""cmake-gui操作：1、选择源码文件夹2、新创建的二进制文件3、工程配置（参考官方）4、Configure 配置，Generate生成5、生成的.so文件在/build/lib/pyth
```



CUDA安装 (reference ymj)

▼ 展开详情

torch==1.3,cuda==10.1,cudnn==7.6.4

1.安装驱动

```
sudo add-apt-repository ppa:graphics-drivers
sudo apt-get update
sudo apt-get install nvidia-418
```

2.安装cuda, 官网地址,

```
sudo sh cuda_10.1.105_418.39_linux.run
```

添加系统环境到`.bashrc`

```
export PATH="/usr/local/cuda-10.1/bin:$PATH"
export LD_LIBRARY_PATH="/usr/local/cuda-10.1/lib64:$LD_LIBRARY_PATH"
```

使用`nvcc -V`验证结果

3.安装cuDNN==7.6.4, 官网地址, 下载`cuDNN Library for Linux`版本

解压只有出现`cuda`文件夹

```
sudo cp cuda/include/cudnn.h  /usr/local/cuda-10.1/include
sudo cp cuda/lib64/libcudnn*  /usr/local/cuda-10.1/lib64
sudo chmod a+r /usr/local/cuda-10.0/include/cudnn.h  /usr/local/cuda-10.1/lib64/libcudnn*
```

virtualenvs虚拟环境配置

▼ 展开详情

虚拟环境配置并更改虚拟环境下的默认python版本

安装虚拟环境依赖

- 法1

运行安装脚本install_virtualenv.sh，自动安装依赖包

- 法2

先安装虚拟环境的依赖库：

```
sudo pip install virtualenvsudo pip install virtualenvwrapper
```

创建虚拟环境管理目录：

```
mkdir ~/.virtualenvssudo vim ~/.bashrc#在.bashrc的末尾增加下面两行内容export WORKON_HOME=$HOME/.virtualenvs  # 所有虚拟环境存储的目录
```

当安装完虚拟环境的依赖后进行下一步创建虚拟环境

创建虚拟环境

```
>>>mkvirtualenv new_vir --python=python3#创建新的虚拟环境为new_vir
# todo
>>>mkvirtualenv -p /usr/bin/python3 new_vir #制定虚拟环境的python为local的python3

>>>workon new_vir #切换到new_vir虚拟环境
>>>deactivate #退出当前虚拟环境
>>>rmvirtualenv new_vir #删除new_vir虚拟环境
```

在虚拟环境中挂载硬盘

首先安装虚拟机的⇒设置⇒硬件高级提升，在设置里建立共享文件夹，路径是真机的一个共享文件夹路径；

打开虚拟机，挂载文件夹(在设置中选择了自动挂载则不需要)

```
sudo mount -t vboxsf path_you_share_file_name path_you_share_file
```

挂载之后即可看见共享文件

github proxy

```
git config --global http.https://github.com.proxy socks5://127.0.0.1:1080
git config --global https.https://github.com.proxy socks5://127.0.0.1:1080
```

ubuntu 更换内核

▼ 展开详情

copy from ubuntu16.04更改启动内核

看到也有网友遇到相似问题，并提出了解决方法（http://askubuntu.com/questions/809199/failed-to-start-load-kernel-modules-ubuntu-16-04）。然而我的问题并未如愿解决。然后想到了修改启动配置文件，更改系统默认加载的内核。

```
grep menuentry /boot/grub/grub.cfg
```

1.该命令显示内核的启动顺序

▼ 比如：

```
zgw@zgw-ThinkPad:~$ grep menuentry /boot/grub/grub.cfg
if [ x"${feature_menuentry_id}" = xy ]; then
  menuentry_id_option="--id"
  menuentry_id_option=""
export menuentry_id_option
menuentry 'Ubuntu' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_option 'gnulinux-simple-5bce37
submenu 'Ubuntu 高级选项' $menuentry_id_option 'gnulinux-advanced-5bce3795-da96-4c6f-bed2-67d37185a77d' {
    menuentry 'Ubuntu, Linux 4.8.0-26-lowlatency' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id
    menuentry 'Ubuntu, with Linux 4.8.0-26-lowlatency (upstart)' --class ubuntu --class gnu-linux --class gnu --class c
    menuentry 'Ubuntu, with Linux 4.8.0-26-lowlatency (recovery mode)' --class ubuntu --class gnu-linux --class gnu --c
    menuentry 'Ubuntu, Linux 4.8.0-26-generic' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_op
    menuentry 'Ubuntu, with Linux 4.8.0-26-generic (upstart)' --class ubuntu --class gnu-linux --class gnu --class os $
    menuentry 'Ubuntu, with Linux 4.8.0-26-generic (recovery mode)' --class ubuntu --class gnu-linux --class gnu --clas
    menuentry 'Ubuntu, Linux 4.4.0-21-generic' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_op
    menuentry 'Ubuntu, with Linux 4.4.0-21-generic (upstart)' --class ubuntu --class gnu-linux --class gnu --class os $
    menuentry 'Ubuntu, with Linux 4.4.0-21-generic (recovery mode)' --class ubuntu --class gnu-linux --class gnu --clas
menuentry 'Memory test (memtest86+)' {
menuentry 'Memory test (memtest86+, serial console 115200)' {
```

2.假设你要以4.4.0-21内核版本启动，则将文件/etc/default/grub中

```
GRUB_DEFAULT=0
```

改为

```
GRUB_DEFAULT=6
```

或者改为

```
GRUB_DEFAULT="Ubuntu，Linux 4.4.0-21-generic"
```

保存后

3.然后使用命令sudo update-grub

4.重新启动后输入uname -r查看，内核即为想要的内核。

然而我的没成功，没办法，只能卸载内核了==========》》看到有牛人的方法（http://www.linuxidc.com/Linux/2016-05/131143.htm），顺便就使用了，在此不再做赘述。

我在恢复模式用低版本内核进入系统后，检查不到4.8.0-26-generic内核，不知道是什么原因。命令如下：

```
 dpkg -l | tail -n +6| grep -E 'linux-image-[0-9]+'| grep -Fv $(uname -r)
```

最后没办法，抱着死马当活马医的想法。

5.直接把/boot/中4.8.0-26相关的文件及文件夹全部删除。命令如下：

```
sudo rm -rf *4.8.0-26*
```

6.然后修改了配置文件：/boot/grub/grub.cfg

```
sudo cp /etc/boot/grub/grub.cfg /etc/boot/grub/grub.cfg.bak.zgw
sudo vim  /etc/boot/grub/grub.cfg
```

7.找到如下代码块（我的为148,149行）：

```
linux    /vmlinuz-4.8.0-26-generic root=UUID=5bce3795-da96-4c6f-bed2-67d37185a77d ro  quiet splash $vt_handoff
    initrd    /initrd.img-4.8.0-26-generic
```

将其改为自己想使用的内核，我的如下：

```
linux    /vmlinuz-4.4.0-45-generic root=UUID=5bce3795-da96-4c6f-bed2-67d37185a77d ro  quiet splash $vt_handoff
    initrd    /initrd.img-4.5.0-45-generic
```

8.然后重启电脑就可以了，如若不行，请找如下代码块（我的为151行）。

```
submenu 'Ubuntu 高级选项' $menuentry_id_option 'gnulinux-advanced-5bce3795-da96-4c6f-bed2-67d37185a77d'
```

9.将此行代码下的与4.8.0-26相关的代码全部删除（我的为152~263行）然后重启就可以了。

▼ 具体信息较长，展开查看

```
menuentry 'Ubuntu，Linux 4.8.0-26-generic' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_option
        recordfail
        load_video
        gfxmode $linux_gfx_mode
        insmod gzio
        if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
        insmod part_msdos
        insmod ext2
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,m
        else
          search --no-floppy --fs-uuid --set=root 88421677-a988-4ff9-bf29-6c56aa4a9027
        fi
        echo    '载入 Linux 4.8.0-26-generic ...'
```

```
        linux    /vmlinuz-4.8.0-26-generic root=UUID=5bce3795-da96-4c6f-bed2-67d37185a77d ro  quiet splash $vt_handoff
        echo    '载入初始化内存盘...'
        initrd    /initrd.img-4.8.0-26-generic
    }
    menuentry 'Ubuntu, with Linux 4.8.0-26-generic (upstart)' --class ubuntu --class gnu-linux --class gnu --class os $
        recordfail
        load_video
        gfxmode $linux_gfx_mode
        insmod gzio
        if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
        insmod part_msdos
        insmod ext2
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,m
        else
          search --no-floppy --fs-uuid --set=root 88421677-a988-4ff9-bf29-6c56aa4a9027
        fi
        echo    '载入 Linux 4.8.0-26-generic ...'
        linux    /vmlinuz-4.8.0-26-generic root=UUID=5bce3795-da96-4c6f-bed2-67d37185a77d ro  quiet splash $vt_handoff
        echo    '载入初始化内存盘...'
        initrd    /initrd.img-4.8.0-26-generic
    }
    menuentry 'Ubuntu, with Linux 4.8.0-26-generic (recovery mode)' --class ubuntu --class gnu-linux --class gnu --clas
        recordfail
        load_video
        insmod gzio
        if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
        insmod part_msdos
        insmod ext2
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,m
        else
          search --no-floppy --fs-uuid --set=root 88421677-a988-4ff9-bf29-6c56aa4a9027
        fi
        echo    '载入 Linux 4.8.0-26-generic ...'
        linux    /vmlinuz-4.8.0-26-generic root=UUID=5bce3795-da96-4c6f-bed2-67d37185a77d ro recovery nomodeset
        echo    '载入初始化内存盘...'
        initrd    /initrd.img-4.8.0-26-generic
    }
    menuentry 'Ubuntu，Linux 4.8.0-22-generic' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_op
        recordfail
        load_video
        gfxmode $linux_gfx_mode
        insmod gzio
        if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
        insmod part_msdos
        insmod ext2
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,m
        else
          search --no-floppy --fs-uuid --set=root 88421677-a988-4ff9-bf29-6c56aa4a9027
        fi
        echo    '载入 Linux 4.8.0-22-generic ...'
        linux    /vmlinuz-4.8.0-22-generic root=UUID=5bce3795-da96-4c6f-bed2-67d37185a77d ro  quiet splash $vt_handoff
        echo    '载入初始化内存盘...'
        initrd    /initrd.img-4.8.0-22-generic
    }
    menuentry 'Ubuntu, with Linux 4.8.0-22-generic (upstart)' --class ubuntu --class gnu-linux --class gnu --class os $
        recordfail
        load_video
        gfxmode $linux_gfx_mode
        insmod gzio
        if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
        insmod part_msdos
        insmod ext2
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,m
        else
          search --no-floppy --fs-uuid --set=root 88421677-a988-4ff9-bf29-6c56aa4a9027
        fi
        echo    '载入 Linux 4.8.0-22-generic ...'
        linux    /vmlinuz-4.8.0-22-generic root=UUID=5bce3795-da96-4c6f-bed2-67d37185a77d ro  quiet splash $vt_handoff
        echo    '载入初始化内存盘...'
        initrd    /initrd.img-4.8.0-22-generic
    }
    menuentry 'Ubuntu, with Linux 4.8.0-22-generic (recovery mode)' --class ubuntu --class gnu-linux --class gnu --clas
        recordfail
        load_video
        insmod gzio
        if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
        insmod part_msdos
        insmod ext2
        set root='hd0,msdos1'
```

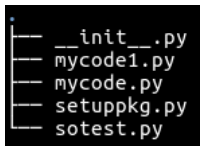
```
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,m
        else
          search --no-floppy --fs-uuid --set=root 88421677-a988-4ff9-bf29-6c56aa4a9027
        fi
        echo    '载入 Linux 4.8.0-22-generic ...'
        linux    /vmlinuz-4.8.0-22-generic root=UUID=5bce3795-da96-4c6f-bed2-67d37185a77d ro recovery nomodeset
        echo    '载入初始化内存盘...'
        initrd    /initrd.img-4.8.0-22-generic
    }
```

py2so

▼ 展开详情

对ubuntu环境下 py⇒so，对文件加密

文件结构：



▼ mycode.py

```
import datetime

class Test(object):
    def __init__(self):
        print('init')

    def say(self):
        print('hello')

    def get_time(self):
        print(datetime.datetime.now())
```

▼ mycode1.py

```
import os
import sys
sys.path.append(os.path.abspath('../../../'))
from mycodes.py2so.demo1.mycode import Test


if __name__ == "__main__":
    test = Test()
    test.say()
    test.get_time()
```

▼ sotest.py

```
import os
import sys
sys.path.append(os.path.abspath('../../../'))
from mycodes.py2so.demo1.mycode import Test

test = Test()
test.get_time()
test.say()
```

▼ setuppkg.py

```
from distutils.core import setup
from Cython.Build import cythonize
```

```
setup(ext_modules=cythonize(["mycode.py"]))
```

▼ 在当前目录的终端下运行如下代码字段

```
$ python setuppkg.py build_ext
# 再查看目录结构
$ tree
.
├── build
│   ├── lib.linux-x86_64-3.6
│   │   └── mycodes
│   │       └── py2so
│   │           └── demo1
│   │               └── mycode.cpython-36m-x86_64-linux-gnu.so
│   └── temp.linux-x86_64-3.6
│       └── mycode.o
├── __init__.py
├── mycode1.py
├── mycode.c
├── mycode.py
├── setuppkg.py
└── sotest.py

# 将源文件改名
$ mv mycode.py mycodebackup.py

# 执行报错，因为还未将so拷贝至当前目录
$ python sotest.py
Traceback (most recent call last):
  File "sotest.py", line 12, in <module>
    from mycodes.py2so.demo1.mycode import Test
ModuleNotFoundError: No module named 'mycodes.py2so.demo1.mycode'

# 拷贝so至当前目录
$ cp ./build/lib.linux-x86_64-3.6/mycodes/py2so/demo1/mycode.cpython-36m-x86_64-linux-gnu.so .
```

ubuntu、windows启动盘

python程序打包

▼ 展开详情

ubuntu环境下

对于一些工具型代码，我们希望在给别人用的时候又不需要安装复杂的依赖环境，这个时候就可以考虑吧代码打包成可执行文件，pyinstaller可以把python代码打包发布.

１．安装

```
sudo pip install pyinstaller
```

２．使用

```
pyinstaller xxx.py
```

默认参数会把python文件的依赖打包出来，并放在build目录下面.

其他可选参数：

-F 必须大写，打包成单一可执行文件，目标文件放在dist目录下

-w 不带termial窗口，加入这个参数后台运行信息将会被隐藏，否则会在终端中显示源码中的print信息

-i 为可执行文件添加图标

```
pyinstaller -F -w -i xxx.ico xxx.py
```

３．参考链接

https://pyinstaller.readthedocs.io/en/stable/

windows环境下

pyinstaller 打包的时候会自动包含依赖

搭建c++环境

▼ 展开详情

Clion

建立工程后，添加CMakeLists.txt，修改其内容为:

```
cmake_minimum_required(VERSION 3.14)
project(untitled)

find_package(OpenCV REQUIRED)
message(STATUS "OpenCV library status:")
message(STATUS "    version: ${OpenCV_VERSION}")
message(STATUS "    libraries: ${OpenCV_LIBS}")
message(STATUS "    include path: ${OpenCV_INCLUDE_DIRS}")
include_directories(${OpenCV_INCLUDE_DIRS})

set(CMAKE_CXX_STANDARD 11)

add_executable(main showimg/main.cpp)
target_link_libraries(main ${OpenCV_LIBS})
```

VS code

再生龙使用

remarkersys依赖过大，暂时抛弃。

配置c++环境

gitlab runner

参考教程

目前问题:服务器的 git -V 确实是指向python3的版本，但是runner里的pip list与服务器本地的pip list不一样。

终端代理

alias setproxy="export ALL_PROXY=socks5://127.0.0.1:1080" alias unsetproxy="unset ALL_PROXY"

alias ip="curl cip.cc"

VS markdown 配置

File → Preferences → Settings → User Setting → Extensions → markdownlint → Edit in setting.json

```
{    "window.zoomLevel": 0,    "markdownlint.config": {      "MD033": false,      "MD007": { "indent": 4 },    },    "files.eol
```

也可以直接编辑配置文件：`$HOME/.config/Code/User/settings.json`

计算机启动

▼ 展开详情

提供有关启动过程的一般概述（从BIOS到启动的内核以及已根安装） 使数十种安装方法中的一些黑魔法变得神秘 提供一些高级启动方案的线索 链接到更具体的说明和方法 启动阶段 启动系统分为四个阶段：

BIOS ### 引导加载程序 ### 内核 ### 新贵（管理系统任务和服务）

由新贵启动的一些核心启动任务是

普利茅斯-图形化启动动画和记录器 mountall-挂载在/ etc / fstab上定义的所有文件系统 网络*-网络相关服务 显示管理器（GDM，KDM，XDM等） （启动任务/服务在/ etc / init中配置）

BIOS阶段

当计算机开始执行时，它将通过执行此代码（也称为固件）开始，因为它通常以永久形式的内存（例如ROM）存储在计算机主板上。在Macintosh计算机上，这是打开固件。

此代码必须初始化CPU以外的硬件，并获取下一步所需的代码，即引导加载程序。现代计算机为引导加载程序提供了多种可能性，并且通常在BIOS启动屏幕上进行选择。

引导加载程序阶段

引导加载程序有几种可能的类型，以及BIOS获取引导加载程序的方式。

A.存储在硬盘的第一扇区，主引导记录或MBR中的引导加载程序。这可能是GRUB或LILO或yaboot或其他。

B.存储在某些其他存储设备（例如CDR或USB闪存驱动器）上的引导加载程序。

C.使用网络的引导加载程序，例如预执行环境（PXE）。此代码通常存储在网卡本身的ROM中。

在存储介质的第一部分需要引导加载程序代码的初始部分，这说明了为什么某些硬盘驱动器是“可启动的”而其他硬盘驱动器却不是。

引导加载程序的工作是开始下一阶段，加载内核和初始ram磁盘文件系统。

内核阶段

内核是操作系统的核心代码，可提供对硬件和其他服务的访问。引导加载程序将启动内核运行。为了使内核保持合理的大小并允许为单独的硬件提供单独的模块，现代内核还使用内存中存在的文件系统，该文件系统称为“初始ram磁盘”的“initrd”。

通常将要加载的内核文件和初始ram磁盘都指定为引导加载程序的选项。

内核在initrd文件系统内启动init脚本，该脚本加载硬件驱动程序并找到根分区。

系统启动

内核运行后，其余的操作系统将联机。

首先，找到，检查并挂载根分区和文件系统。 接下来，启动初始化进程，该进程运行初始化脚本。 这些脚本涉及不同的/etc / rc脚本和暴发户事件， 最终会为您提供带有登录屏幕的随时可用的计算机。

引导组件 MBR（IBM兼容PC） 主引导记录是磁盘上的第一个扇区，通常包含磁盘的分区表和简单的引导加载程序。在大多数情况下，此简单的引导加载程序仅在同一磁盘上查找活动分区，然后跳转到该分区上的引导扇区。引导扇区将包含实际的引导加载程序。

GRUB引导程序 因为GRUB引导加载程序提供了选择菜单，并且可以处理许多不同形式的硬件，所以它比单个MBR中可以容纳的代码更大。它具有3个阶段：MBR中的阶段1，磁盘第一个圆柱面的其余部分中的阶段1.5和磁盘中文件中的阶段2。

Grub将找到/boot/grub/menu.lst来配置其交互式菜单。menu.lst以及stage1.5和stage2文件的位置在安装到引导扇区时会被硬编码到grub中。Grub使用BIOS调用及其对文件系统的内置识别功能来定位并加载内核和initrd（这要归功于不同的stage1.5部分）。最后启动内核。

在某些情况下，操作系统会划分为多个分区（例如/usr），并且引导脚本会尽快挂载这些分区。

成功的条件 首先，BIOS必须找到引导加载程序，这取决于您的硬件功能。 其次，引导加载程序必须找到内核和initrd。它可能会使用BIOS调用，因此这又取决于您的BIOS。 最后，内核将启动，并且必须在initrd的帮助下找到根分区 查找根分区 操作系统的根分区可以与内核完全不同，例如在另一个驱动器或远程计算机上。 在某些情况下，内核可能找不到磁盘上的根分区，因为initrd缺少访问分区的模块。如果是这种情况，请重建initrd以包括缺少的模块（请参阅man mkinitramfs和man update-initramfs）。

参考来源 Booting

Ubuntu启动盘

grldr menu下新增内容

```
title 安装ubuntu16.04LTS_64位
      find --set-root /.images/ubuntu16/vmlinuz.efi
      kernel /.images/ubuntu16/vmlinuz.efi boot=casper iso-scan/filename=/.images/ubuntu16/ubuntu16.04.iso ro quiet splash  loc
      initrd /.images/ubuntu16/initrd.lz

title 安装windows10
      SISO RUN /.images/win10/windows_10.iso
```