习题一

熟悉 Linux

1. 如何在 Ubuntu 中安装软件（命令行界面）？它们通常被安装在什么地方?
    - apt-get，安装的路径通常默认安装在/usr，/var
2. linux 的环境变量是什么？我如何定义新的环境变量?
    - 环境变量是定义在shell 里的变量， 可以直接用 variablename= 来定义
3. linux 根目录下面的目录结构是什么样的？至少说出 3 个目录的用途。
    - /bin 二进制可执行命令
    - /dev 设备特殊文件
    - /etc 系统管理和配置文件
    - home home directory, 用户目录
4. 假设我要给 a.sh 加上可执行权限，该输入什么命令?
    - chmod +x a.sh
5. 假设我要将 a.sh 文件的所有者改成 xiang:xiang，该输入什么命令?
    - chown xiang:xiang a.sh

SLAM 综述文献阅读

1. SLAM 会在哪些场合中用到？至少列举三个方向。
    1. 增强现实
    2. 无人驾驶高精度地图
    3. Robatics
2. SLAM 中定位与建图是什么关系？为什么在定位的同时需要建图?
    1. 在SLAM中定位 建图 是同时进行的， 即在运动过程中建立周围环境的模型，同时估计自己的运动。通过建图能够矫正定位误差偏移的累积， 形成一个闭环。
3. SLAM 发展历史如何？我们可以将它划分成哪几个阶段?

    根据 这篇文章Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age， SLAM 可以分为 classical age (1986-2004)， 第二阶段为 algorithmic-analysis age (2004-2015), 第一阶段主要集中在用概率方式的表达SLAM模型，如使用 Kalman filter。 第二阶段主要集中在SLAM的特性如可见性，一致性等
4. 列举三篇在 SLAM 领域的经典文献。
    1. R. C. Smith and P.Cheeseman, On the representation and estimation of spatial uncertainly
    2. A.Davison, I.Reid, N.molton, and O.Stasse, MonoSLAM: Real-Time Single Camera SLAM

3. R Mur-Artal, ORB-SLAM: A Versatile and Accurate Monocular SLAM System.

CMake 练习

理解 ORB-SLAM2 框架

1. 下载ORB-SLAM2的截图

```
jinxuanw@jinxuanw-server:~/SLAM/week1$ git clone https://github.com/raulmur/ORB_
SLAM2
Cloning into 'ORB_SLAM2'...
remote: Counting objects: 566, done.
remote: Total 566 (delta 0), reused 0 (delta 0), pack-reused 566
Receiving objects: 100% (566/566), 41.44 MiB | 21.38 MiB/s, done.
Resolving deltas: 100% (176/176), done.
Checking connectivity... done.
jinxuanw@jinxuanw-server:~/SLAM/week1$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 16.04.2 LTS
Release:        16.04
Codename:       xenial
jinxuanw@jinxuanw-server:~/SLAM/week1$
```

2. 阅读 ORB-SLAM2 代码目录 下的 CMakeLists.txt，回答问题：
   - ORB-SLAM2 将编译出什么结果？有几个库文件和可执行文件?
     - ORB-SLAM2 将编译生成 6 个可执行文件，分别[rgbd_tum, stereo_kitti,stereo_euroc, mono_tum, mono_kitti, mono_euroc].
     - ORB-SLAM2 将编译生成1个库文件，libORB_SLAM2.so
   - ORB-SLAM2 中的 include, src, Examples 三个文件夹中都含有什么内容?
     - include 里有.h 头文件， src 里有.cpp 的源文件， Examples 里有对不同配置文件，相机标定.yaml 以

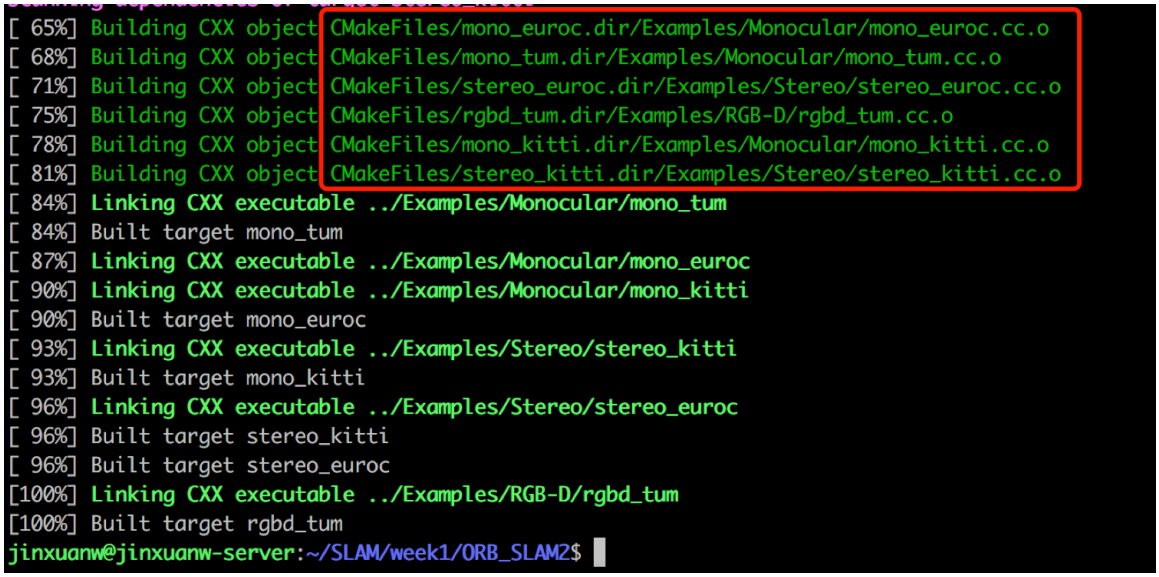
```
jinxuanw@jinxuanw-server:~/SLAM/week1/tmp/ORB_SLAM2$ tree -L 2 src/ include/ Examples/
src/
├── Converter.cc
├── Frame.cc
├── FrameDrawer.cc
├── Initializer.cc
├── KeyFrame.cc
├── KeyFrameDatabase.cc
├── LocalMapping.cc
├── LoopClosing.cc
├── Map.cc
├── MapDrawer.cc
├── MapPoint.cc
├── Optimizer.cc
├── ORBextractor.cc
├── ORBmatcher.cc
├── PnPsolver.cc
├── Sim3Solver.cc
├── System.cc
```

```
├── Tracking.cc
└── Viewer.cc
include/
├── Converter.h
├── FrameDrawer.h
├── Frame.h
├── Initializer.h
├── KeyFrameDatabase.h
├── KeyFrame.h
├── LocalMapping.h
├── LoopClosing.h
├── MapDrawer.h
├── Map.h
├── MapPoint.h
├── Optimizer.h
├── ORBextractor.h
├── ORBmatcher.h
├── ORBVocabulary.h
├── PnPsolver.h
├── Sim3Solver.h
├── System.h
├── Tracking.h
└── Viewer.h
Examples/
├── Monocular
│   ├── EuRoC_TimeStamps
│   ├── EuRoC.yaml
│   ├── KITTI00-02.yaml
│   ├── KITTI03.yaml
│   ├── KITTI04-12.yaml
│   ├── mono_euroc.cc
│   ├── mono_kitti.cc
│   ├── mono_tum.cc
│   ├── TUM1.yaml
│   ├── TUM2.yaml
│   └── TUM3.yaml
├── RGB-D
│   ├── associations
│   ├── rgbd_tum.cc
│   ├── TUM1.yaml
│   ├── TUM2.yaml
│   └── TUM3.yaml
├── ROS
│   └── ORB_SLAM2
└── Stereo
    ├── EuRoC_TimeStamps
    ├── EuRoC.yaml
    ├── KITTI00-02.yaml
    ├── KITTI03.yaml
    ├── KITTI04-12.yaml
    ├── stereo_euroc.cc
    └── stereo_kitti.cc

8 directories, 59 files
jinxuanw@jinxuanw-server:~/SLAM/week1/tmp/ORB_SLAM2$
```

- ORB-SLAM2 中的可执行文件链接到了哪些库？它们的名字是什么？

  如下图所示， 以及 libORB_SLAM2.so



使用摄像头或视频运行 ORB-SLAM2

1. 顺利编译 ORBSLAM2 的截图:

```
jinxuanw@jinxuanw-server:~/SLAM/week1/ORB_SLAM2$ ./build.sh
Configuring and building Thirdparty/DBoW2 ...
mkdir: cannot create directory ‘build’: File exists
-- Configuring done
-- Generating done
-- Build files have been written to: /home/jinxuanw/SLAM/week1/ORB_SLAM2/Thirdparty/DBoW2/build
[100%] Built target DBoW2
Configuring and building Thirdparty/g2o ...
mkdir: cannot create directory ‘build’: File exists
-- BUILD TYPE:Release
-- Compiling on Unix
-- Configuring done
-- Generating done
-- Build files have been written to: /home/jinxuanw/SLAM/week1/ORB_SLAM2/Thirdparty/g2o/build
[100%] Built target g2o
Uncompress vocabulary ...
Configuring and building ORB_SLAM2 ...
mkdir: cannot create directory ‘build’: File exists
Build type: Release
-- Using flag -std=c++11.
-- Configuring done
-- Generating done
-- Build files have been written to: /home/jinxuanw/SLAM/week1/ORB_SLAM2/build
[  3%] Linking CXX shared library ../lib/libORB_SLAM2.so
[ 62%] Built target ORB_SLAM2
Scanning dependencies of target mono_euroc
Scanning dependencies of target stereo_euroc
Scanning dependencies of target mono_tum
Scanning dependencies of target mono_kitti
Scanning dependencies of target rgbd_tum
Scanning dependencies of target stereo_kitti
[ 65%] Building CXX object CMakeFiles/mono_euroc.dir/Examples/Monocular/mono_euroc.cc.o
[ 68%] Building CXX object CMakeFiles/mono_tum.dir/Examples/Monocular/mono_tum.cc.o
[ 71%] Building CXX object CMakeFiles/stereo_euroc.dir/Examples/Stereo/stereo_euroc.cc.o
[ 75%] Building CXX object CMakeFiles/rgbd_tum.dir/Examples/RGB-D/rgbd_tum.cc.o
[ 78%] Building CXX object CMakeFiles/mono_kitti.dir/Examples/Monocular/mono_kitti.cc.o
[ 81%] Building CXX object CMakeFiles/stereo_kitti.dir/Examples/Stereo/stereo_kitti.cc.o
[ 84%] Linking CXX executable ../Examples/Monocular/mono_tum
[ 84%] Built target mono_tum
[ 87%] Linking CXX executable ../Examples/Monocular/mono_euroc
[ 90%] Linking CXX executable ../Examples/Monocular/mono_kitti
[ 90%] Built target mono_euroc
[ 93%] Linking CXX executable ../Examples/Stereo/stereo_kitti
[ 93%] Built target mono_kitti
[ 96%] Linking CXX executable ../Examples/Stereo/stereo_euroc
[ 96%] Built target stereo_kitti
[ 96%] Built target stereo_euroc
[100%] Linking CXX executable ../Examples/RGB-D/rgbd_tum
[100%] Built target rgbd_tum
jinxuanw@jinxuanw-server:~/SLAM/week1/ORB_SLAM2$
```

2. 将 myslam.cpp 或 myvideo.cpp 加入到 ORB-SLAM2 工程中, CMakeLists.txt 修改方案： 添加可执行文件 并连接上

```
set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${PROJECT_SOURCE_DIR}/code)

add_executable(myvideo
code/myvideo.cpp)
target_link_libraries(myvideo ${PROJECT_NAME})
```

3. 运行结果如图