0. 说明

本PDF文档为自动生成，如有遗漏的格式错误请及时告知！

---

1. 熟悉Linux

问1-1 ：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?

答：

- 在线安装
  - 一般使用命令`sudo apt-get install XXX`（XXX为要安装的软件名），有时会先运行`sudo apt-update`更新软件源，这种安装方式的好处是可以将依赖包一并安装。
  - 这种安装方式，软件一般安装到了/usr目录下，usr是Unix Software Resource的缩写。其中/usr/bin用于存放可执行文件，/usr/lib存放库文件，/usr/share存放共享文件，/usr/local一般是用户安装自己下载软件的位置。
- 离线安装
  - 根据文件类型的不同有不同的安装方法，例如rpm、bin、deb、run之类扩展名文件。
  - 这种安装方式一般安装到/opt目录下，当然你也可以指定安装路径。

问1-2：linux 的环境变量是什么?我如何定义新的环境变量?

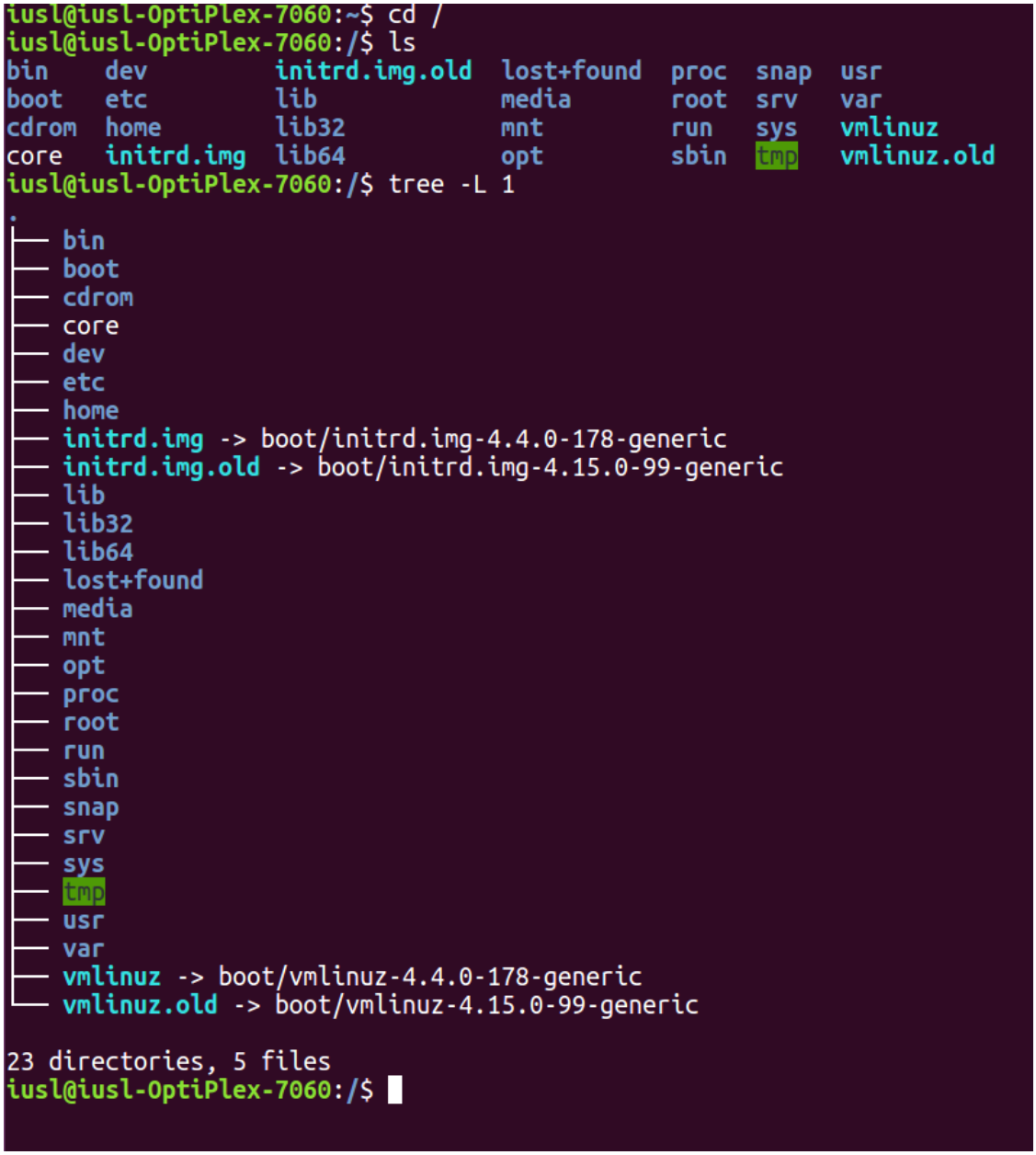
答：

- linux 的环境变量就是指定系统运行环境的参数。可以用`env`命令查看；
- `export`命令可以用来申明环境变量，例如`export SLAM = "Hello SLAM"`；
- 如果需要长期使用该环境变量，需要在配置文件下配置。/etc/profile和/etc/bashrc文件对所有用户有效,~/bashrc和~/.bash_profile只对当前用户有效。

问1-3：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。

答：

-



- 部分目录用途
  - /bin ：存放所有二进制命令（用户）
  - /boot：Linux内核及引导系统程序所需的目录
  - /dev：所有设备文件的目录（如声卡、磁盘、光驱）
  - /etc：配置文件默认路径，服务启动命令存放目录
  - /home：普通用户的家目录默认数据存放目录
  - /proc：进程及内核信息存放目录

问1-4：假设我要给 a.sh 加上可执行权限,该输入什么命令?

答：

```
#Linux/Unix 的文件调用权限分为三级 ： 文件拥有者、群组、其他
#u表示该文件的拥有者，g表示与该文件的拥有者属于同一个群体(group)者，o表示其他以外的人，a表示这三者皆是
#+ 表示增加权限、- 表示取消权限、= 表示唯一设定权限
#r 表示可读取，w 表示可写入，x 表示可执行
chmod a+x   a.sh
```

问1-5：假设我要将 a.sh 文件的所有者改成 xiang:xiang,该输入什么命令?

答：

```
chown xiang:xiang a.sh
```

---

2. SLAM 综述文献阅读

问2-1：SLAM 会在哪些场合中用到?至少列举三个方向。

答：

SLAM可以用于那些需要实时构建地环境信息的场合。

- 无人驾驶

  辅助无人车行驶。

- AR

  SLAM可以在未知环境中定位自身方位并同时构建环境三维地图, 从而保证叠加的虚拟与现实场景在几何上的一致性。

- 机器人自主导航

  危险场景下、GPS失效的室内、太空中、水下等进行探测、定位和建图。

.....

问2-2：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?

答：

准确的定位需要用到精确的环境信息（建图），精确的建图也需要知道相机的准确位置。

问2-3：SLAM 发展历史如何?我们可以将它划分成哪几个阶段?

答：

- **classical age(1986~2004)**：

  引入了SLAM的主要概率公式，包括基于扩展卡尔曼滤波器、RaoBlackwelled粒子滤波器和最大似然估计的方法 。

- **algorithmic-analysis age(2004~2015)**：

  研究SLAM的基础性质，包括可观性、收敛性和一致性。这一时期出现了很多开源SLAM库。

- **robust-perception age**：

  希望进一步提高SLAM的可用性，包括更加鲁棒，能提供更高层次的环境信息（理解环境）等。

问2-4：列举三篇在 SLAM 领域的经典文献。

答：

- Engel J , Schps T , Cremers D . LSD-SLAM: Large-scale direct monocular SLAM[J]. 2014.

  直接法单目 SLAM

- Mur-Artal R , Montiel J M M , Tardos J D . ORB-SLAM: A Versatile and Accurate Monocular SLAM System[J]. Robotics IEEE Transactions on, 2015, 31(5):1147-1163.

ORB-SLAM

- Alex Kendall, Matthew Grimes, Roberto Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization", in ICCV 2015

  深度学习+SLAM

---

3. CMake 练习

```
cmake_minimum_required(VERSION 2.8.3)
SET(CMAKE_BUILD_TYPE "Release")
PROJECT (HELLO)

INCLUDE_DIRECTORIES(${PROJECT_SOURCE_DIR}/include)

ADD_LIBRARY(hello SHARED ${PROJECT_SOURCE_DIR}/src/hello.cpp)

SET(SRC_LIST ${PROJECT_SOURCE_DIR}/src/useHello.cpp)
ADD_EXECUTABLE(sayhello ${SRC_LIST})

TARGET_LINK_LIBRARIES(sayhello hello)

INSTALL(TARGETS hello LIBRARY DESTINATION /usr/local/lib)
INSTALL(FILES ${PROJECT_SOURCE_DIR}/include/hello.h DESTINATION
/usr/local/include)
```

```
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM/Hello/build$ cmake ..
-- The C compiler identification is GNU 5.4.0
-- The CXX compiler identification is GNU 5.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/iusl/lyq/github/SLAM/Hello/build
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM/Hello/build$ make
Scanning dependencies of target hello
[ 25%] Building CXX object CMakeFiles/hello.dir/src/hello.cpp.o
[ 50%] Linking CXX shared library libhello.so
[ 50%] Built target hello
Scanning dependencies of target sayhello
[ 75%] Building CXX object CMakeFiles/sayhello.dir/src/useHello.cpp.o
[100%] Linking CXX executable sayhello
[100%] Built target sayhello
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM/Hello/build$ sudo make install
[ 50%] Built target hello
[100%] Built target sayhello
Install the project...
-- Install configuration: ""
-- Installing: /usr/local/lib/libhello.so
-- Up-to-date: /usr/local/include/hello.h
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM/Hello/build$ ./sayhello
Hello SLAM
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM/Hello/build$
```

---

4. 理解 ORB-SLAM2 框架

4-1：下载 ORB-SLAM2 的代码

```
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM$ git clone https://github.com/raulmur/ORB_SLAM2.git --recursive
Cloning into 'ORB_SLAM2'...
remote: Enumerating objects: 566, done.
remote: Total 566 (delta 0), reused 0 (delta 0), pack-reused 566
Receiving objects: 100% (566/566), 41.41 MiB | 4.53 MiB/s, done.
Resolving deltas: 100% (177/177), done.
Checking connectivity... done.
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM$
```

4-2 ：阅读 ORB-SLAM2 代码目录下的 CMakeLists.txt

- (a)ORB-SLAM2 将编译出什么结果?有几个库文件和可执行文件?
  - 除去生成的中间文件，我们主要关心ORB-SLAM2编译之后会生成可执行文件和库文件；
  - 除去第三方库文件和中间生成的链接文件，ORB-SLAM2生成的库文件是libORB_SLAM2.so；
  - 可执行文件为：rgbd_tum、stereo_kitti、stereo_euroc、mono_tum、mono_kitti、mono_euroc 。
- (b)ORB-SLAM2 中的 include, src, Examples 三个文件夹中都含有什么内容?

  这里不讨论文件具体内容含义，只说明文件功能。

- Examples中存放的分别是基于单目、双目、RGBD的实例程序（基于TUM、KITTI 、EuRoC数据库），另外还有一个ROS版本的ORB-SLAM2；
- include文件夹中存放的是程序需要用到的一些头文件；
- src文件夹存放的是ORB-SLAM2的程序源码，一堆.cc文件，这些文件最终一起生成了libORB_SLAM2.so库文件。

- (c)ORB-SLAM2 中的可执行文件链接到了哪些库?它们的名字是什么?

```
add_library(${PROJECT_NAME} SHARED
src/System.cc
src/Tracking.cc
src/LocalMapping.cc
src/LoopClosing.cc
src/ORBextractor.cc
src/ORBmatcher.cc
src/FrameDrawer.cc
src/Converter.cc
src/MapPoint.cc
src/KeyFrame.cc
src/Map.cc
src/MapDrawer.cc
src/Optimizer.cc
src/PnPsolver.cc
src/Frame.cc
src/KeyFrameDatabase.cc
src/Sim3Solver.cc
src/Initializer.cc
src/Viewer.cc
)

target_link_libraries(${PROJECT_NAME}
${OpenCV_LIBS}
${EIGEN3_LIBS}
${Pangolin_LIBRARIES}
${PROJECT_SOURCE_DIR}/Thirdparty/DBoW2/lib/libDBoW2.so
${PROJECT_SOURCE_DIR}/Thirdparty/g2o/lib/libg2o.so
)
```

---

5. 使用摄像头或视频运行 ORB-SLAM2

下面记录我在运行ORB-SLAM2时碰到的一些需要注意的点与运行结果：

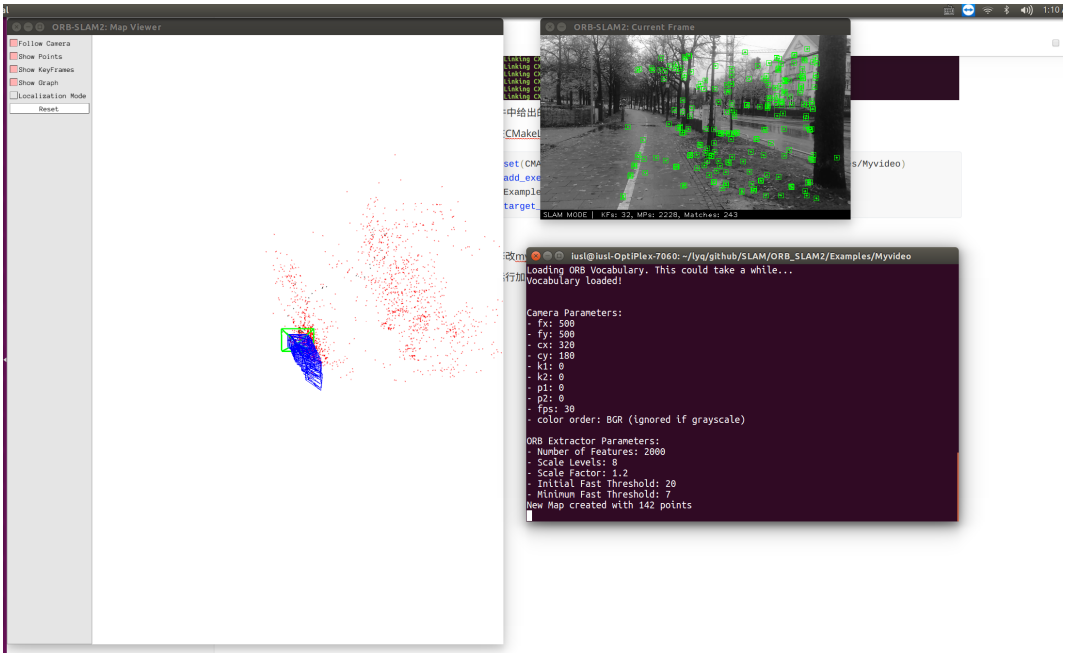
1. 安装ORB-SLAM2的依赖项pangolin库时，也需要安装一些pangolin的依赖项，其中有些python库，安装时候注意python版本；
2. 编译ORB-SLAM2结果：

```
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM/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/iusl/lyq/github/SLAM/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/iusl/lyq/github/SLAM/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/iusl/lyq/github/SLAM/ORB_SLAM2/build
Scanning dependencies of target ORB_SLAM2
[  6%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/System.cc.o
[  6%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/Tracking.cc.o
[  9%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/LocalMapping.cc.o
[ 12%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/LoopClosing.cc.o
[ 18%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/ORBmatcher.cc.o
[ 15%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/ORBextractor.cc.o
[ 21%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/Converter.cc.o
[ 25%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/FrameDrawer.cc.o
[ 28%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/MapPoint.cc.o
[ 34%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/KeyFrame.cc.o
[ 34%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/MapDrawer.cc.o
[ 37%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/Map.cc.o
[ 43%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/PnPsolver.cc.o
[ 43%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/Sim3Solver.cc.o
[ 50%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/Optimizer.cc.o
[ 50%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/Frame.cc.o
[ 53%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/KeyFrameDatabase.cc.o
[ 56%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/Viewer.cc.o
[ 59%] Building CXX object CMakeFiles/ORB_SLAM2.dir/src/Initializer.cc.o
[ 62%] Linking CXX shared library ../lib/libORB_SLAM2.so
[ 62%] Built target ORB_SLAM2
Scanning dependencies of target rgbd_tum
Scanning dependencies of target mono_euroc
Scanning dependencies of target mono_kitti
Scanning dependencies of target stereo_euroc
Scanning dependencies of target stereo_kitti
Scanning dependencies of target mono_tum
[ 68%] Building CXX object CMakeFiles/mono_tum.dir/Examples/Monocular/mono_tum.cc.o
[ 68%] Building CXX object CMakeFiles/mono_kitti.dir/Examples/Monocular/mono_kitti.cc.o
[ 71%] Building CXX object CMakeFiles/stereo_kitti.dir/Examples/Stereo/stereo_kitti.cc.o
[ 75%] Building CXX object CMakeFiles/stereo_euroc.dir/Examples/Stereo/stereo_euroc.cc.o
[ 78%] Building CXX object CMakeFiles/mono_euroc.dir/Examples/Monocular/mono_euroc.cc.o
[ 81%] Building CXX object CMakeFiles/rgbd_tum.dir/Examples/RGB-D/rgbd_tum.cc.o
[ 84%] Linking CXX executable ../Examples/Monocular/mono_kitti
[ 87%] Linking CXX executable ../Examples/Monocular/mono_euroc
[ 90%] Linking CXX executable ../Examples/Monocular/mono_tum
[ 93%] Linking CXX executable ../Examples/RGB-D/rgbd_tum
[ 96%] Linking CXX executable ../Examples/Stereo/stereo_kitti
[100%] Linking CXX executable ../Examples/Stereo/stereo_euroc
```

3. 以课件中给出的视频资料为例子，运行ORB-SLAM2
   - 在CMakeLists.txt文件中添加以下命令：

```
set(CMAKE_RUNTIME_OUTPUT_DIRECTORY
${PROJECT_SOURCE_DIR}/Examples/Myvideo)
add_executable(myvideo
Examples/Myvideo/myvideo.cpp)
target_link_libraries(myvideo ${PROJECT_NAME})
```

   - 修改myvideo.cpp文件，将参数文件、字典文件和视频文件的地址修改成你电脑上的这些文件的路径；
   - 运行加载Vocabulary时，如果内存溢出了，建议关闭部分应用程序，清空内存。
   - 运行截图：



- 体会：
  - 有趣；
  - 疑惑和强烈的想弄明白其中的原理；
  - 如果是实际使用，需要一定的工程实践能力；
  - 这周时间不足，有工作而且下周还要出差一周，所以无法进行摄像头的实验验证；
  - 奥利给！！！