Dobot Magician

Demo 说明（ROS）

文档版本：V1.0

发布日期：2018-08-30

深圳市越疆科技有限公司

版权所有 © 越疆科技有限公司2017。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

免责申明

在法律允许的最大范围内，本手册所描述的产品（含其硬件、软件、固件等）均“按照现状”提供，可能存在瑕疵、错误或故障，越疆不提供任何形式的明示或默示保证，包括但不限于适销性、质量满意度、适合特定目的、不侵犯第三方权利等保证；亦不对使用本手册或使用本公司产品导致的任何特殊、附带、偶然或间接的损害进行赔偿。

在使用本产品前详细阅读本使用手册及网上发布的相关技术文档并了解相关信息，确保在充分了解机器人及其相关知识的前提下使用机械臂。越疆建议您在专业人员的指导下使用本手册。该手册所包含的所有安全方面的信息都不得视为Dobot的保证，即便遵循本手册及相关说明，使用过程中造成的危害或损失依然有可能发生。

本产品的使用者有责任确保遵循相关国家的切实可行的法律法规，确保在越疆机械臂的使用中不存在任何重大危险。

越疆科技有限公司

地址：深圳市南山区同富裕工业城三栋三楼

网址：http://cn.dobot.cc/

前 言

修订记录

时间	修订记录
2018/08/30	第一次发布

符号约定

在本手册中可能出现下列标志，它们所代表的含义如下。

符号	说明
⚠危险	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害
⚠警告	表示有中度或低度潜在危害，如果不能避免，可能导致人员轻微伤害、机械臂毁坏等情况
⚠注意	表示有潜在风险，如果忽视这些文本，可能导致机械臂损坏、数据丢失或不可预知的结果
🕮说明	表示是正文的附加信息，是对正文的强调和补充

目 录

1. 基于 ROS 的 Demo 1
1.1 环境搭建 1
1.2 ROS Demo 说明 5
1.2.1 工程说明 5
1.2.2 代码说明 9

1. 基于 ROS 的 Demo

本文档旨在协助用户熟悉 Dobot Magician 常用 API 和快速搭建开发环境。

1.1 环境搭建

ROS当前仅支持在Linux系统上安装部署，所以ROS Demo基于Ubuntu Linux系统开发。除安装Ubuntu Linux操作系统外，还需安装配套版本的ROS。Ubuntu Linux操作系统可支持在本地安装，也支持在VMware安装，建议选择Ubuntu16.04 LTS版本进行安装，配套的ROS版本为Kinetic。

该ROS Demo所使用的Ubuntu Linux操作系统基于VMware安装，本节不提供详细的安装指导。

如果你已经安装Ubuntu Linux操作系统，请在终端输入**cat /etc/issue**查看操作系统版本，根据操作系统版本选择配套的ROS版本进行安装。Ubuntu Linux操作系统版本与ROS版本对应关系如表 1.1所示。更多信息请参考http://www.ros.org/。

表 1.1 Ubuntu Linux 操作系统版本与 ROS 版本对应关系

ROS版本	对应的Ubuntu版本
Lunar	Ubuntu 17.04
Kinetic（建议选用）	**Ubuntu16.04**
Jade	Ubuntu 15.04
Indigo	Ubuntu 14.04
…	…

⚠注意

如果ROS版本与Ubuntu Linux操作系统不配套，可能会出现各种各样的依赖错误，安装时请确保ROS版本与Ubuntu Linux操作系统配套。

步骤 1 以普通用户登录Ubuntu Linux操作系统。

步骤 2 配置Ubuntu软件仓库。

正式安装前，请先检查Ubuntu初始环境是否配置正确。

1. 在“System Settings… > Software&Updates > Ubuntu Software”页签勾选带有“universe”、“restricted”、“multiverse”关键字的选项，并将“Download from”选择为“Server for United States”，如图 1.1所示。

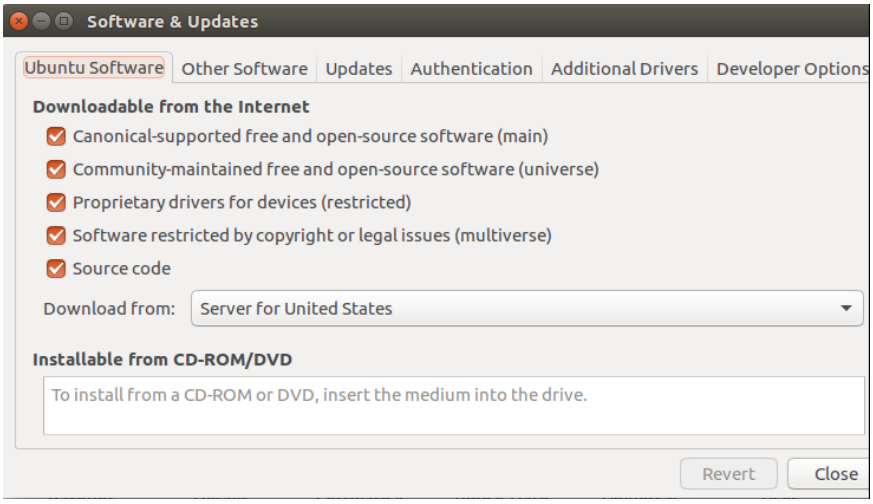

图 1.1 Ubuntu 软件设置

2. 在“System Settings… > Software&Updates > Others Software”页签将设置修改成如图 1.2所示。

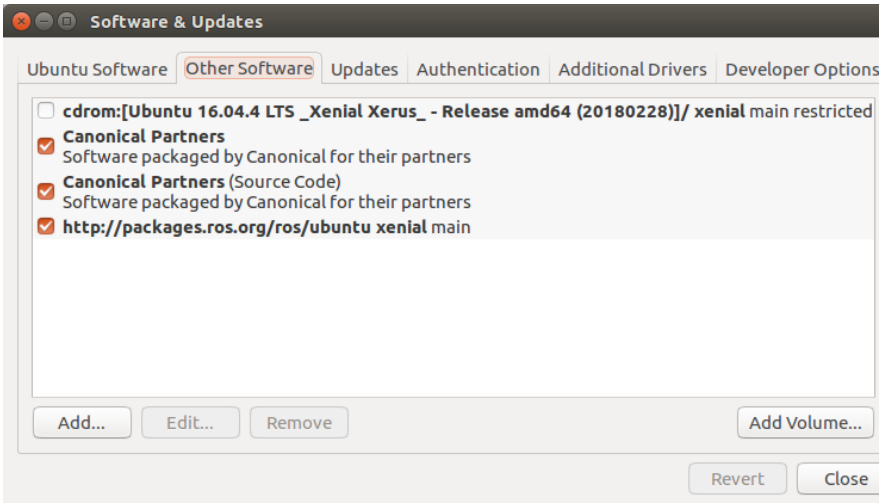

图 1.2 其他软件设置

3. 单击“Close”，等待约30秒后缓存更新完成。

步骤 3 配置 ROS 的 apt 源。

1. 添加“sources.list”，将镜像添加到Ubuntu Linux系统源列表中。

   建议使用国内镜像源，保证下载速度。本例使用的是国内USTC源。

- 方式一：官方源

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

- 方式二：国内USTC源

```
$ sudo sh -c '. /etc/lsb-release && echo "deb http://mirrors.ustc.edu.cn/ros/ubuntu/ $DISTRIB_CODENAME main" > /etc/apt/sources.list.d/ros-latest.list'
```

- 方式三：新加坡源

```
$ sudo sh -c '. /etc/lsb-release && echo "deb http://mirror-ap.packages.ros.org/ros/ubuntu/
$DISTRIB_CODENAME main" > /etc/apt/sources.list.d/ros-latest.list'
```

2. 添加公钥。公钥是Ubuntu Linux操作系统的一种安全机制，也是ROS安装中不可缺的一部分。

```
$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key
421C365BD9FF1F717815A3895523BAEEB01FA116
```

3. 更新系统。

```
$ sudo apt-get update && sudo apt-get upgrade
```

更新系统后，确保Debian软件包和索引是最新的。

4. 安装ROS软件包。

ROS官网提供了四种默认的安装方式，这四种方式包括桌面完整版安装、桌面版安装、基础版安装、单独软件包安装。不同的安装方式，ROS提供的函数库和工具不同。

该Demo推荐安装桌面完整版安装（包含ROS、rqt、rviz、通用机器人函数库、2D/3D仿真器、导航以及2D/3D感知功能），以Ubuntu 16.04安装Kinetic版本为例，在终端输入：

```
$ sudo apt-get install ros-kinetic-desktop-full
```

如果用户使用的Ubuntu和ROS为其他版本，则需将命令中的“ros-kinetic”改为对应版本的命令，例如，若用户在Ubuntu 14.04安装Indigo版本，则需在终端输入：

```
$ sudo apt-get install ros-indigo-desktop-full
```

用户也可根据自己的需求采用其他的安装方式：

- 桌面版安装：包含ROS、rqt、rviz以及通用机器人函数库。

```
$ sudo apt-get install ros-kinetic-desktop
```

- 基础版安装：包含ROS核心软件包、构建工具以及通信相关的程序库。

```
$ sudo apt-get install ros-kinetic-ros-base
```

- 单独软件包安装：安装指定的功能包，当运行ROS时缺少某些系统依赖的功能包时需用到。其中，*PACKAGE*为功能名，请根据实际情况替换*PACKGAGE*。

```
$ sudo apt-get install ros-kinetic-PACKAGE
```

例如系统提示找不到slam-gmapping，在终端输入：

```
$ sudo apt-get install ros-kinetic-slam-gmapping
```

如果用户需要要查找可用的软件包，请在终端输入：

```
$ apt-cache search ros-kinetic
```

如果在安装过程中出现如下问题，则可能是ROS版本不兼容，或者镜像源没有更新。请参考**步骤 1**检查Ubuntu初始环境是否配置正确。如果仍无法解决，可访问http://wiki.ros.org/ROS/查看具体问题的解决方案。

```
下列软件包有未满足的依赖关系： ros-kinetic-desktop-full :
依赖: ros-kinetic-desktop 但是它将不会被安装；
```

```
依赖: ros-kinetic-perception 但是它将不会被安装；
依赖: ros-kinetic-simulators 但是它将不会被安装；
E: 无法修正错误， 因为您要求某些软件包保持现状， 就是它们破坏了软件包间的依赖关系。
```

步骤 4 配置ROS。

1. 初始化rosdep。

```
$ sudo rosdep init && rosdep update
```

初始化rosdep，是使用ROS之前的必要一步。rosdep可以方便在你需要编译某些源码的时候为其安装一些系统依赖， 同时也是某些ROS核心功能组件所必需用到的工具。

2. 配置ROS环境。

```
#For Ubuntu 16.04
$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

```
#For Ubuntu 14.04
$ echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

3. 安装rosinstall。

rosinstall是ROS中一个独立的常用命令行工具，只需一个命令即可轻松的下载ROS程序包所需的资源树。

```
$ sudo apt-get install python-rosinstall
```

步骤 5 检查ROS是否能正常运行。

在终端输入如下命令启动ROS。

```
$ roscore
```

如果出现如图 1.3所示的状态，则说明ROS启动正常。

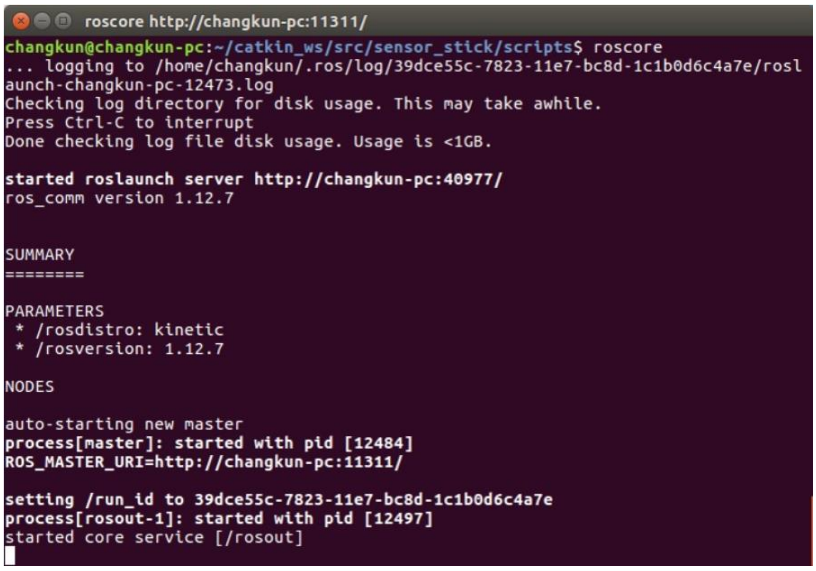

图 1.3 ROS 启动成功

1.2 ROS Demo 说明

1.2.1 工程说明

步骤 1 将获取的“dobot_ws”Demo包解压至“/home”目录下，如图 1.4所示。

⚠注意

请确保在Linux环境下解压，否则会出现编译错误。

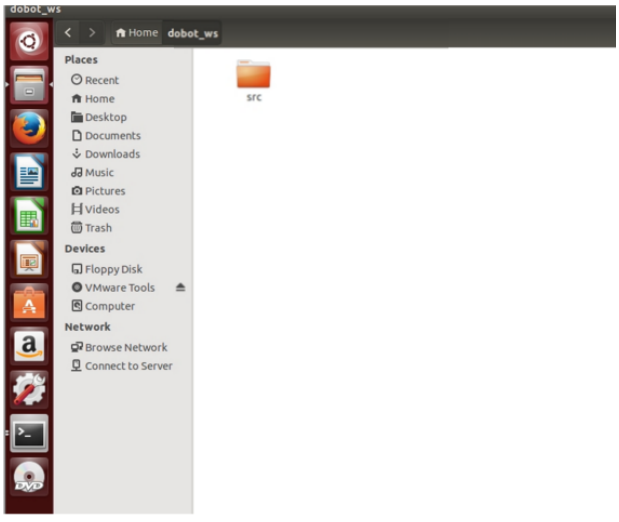

图 1.4 解压 Demo 包

步骤 2 设置普通用户的串口读写权限。

```
$ sudo usermod -a -G dialout username
```

其中，*username*为普通用户名，请根据实际情况替换。

步骤 3 修改“dobot_ws”目录权限。

```
$ cd /home/dobot-ws
$ sudo chmod 777 ./* -R
```

步骤 4 编译ROS Demo。

```
$ cd /home/dobot-ws
$ catkin_make
```

编译完成后，显示如图 1.5所示内容。

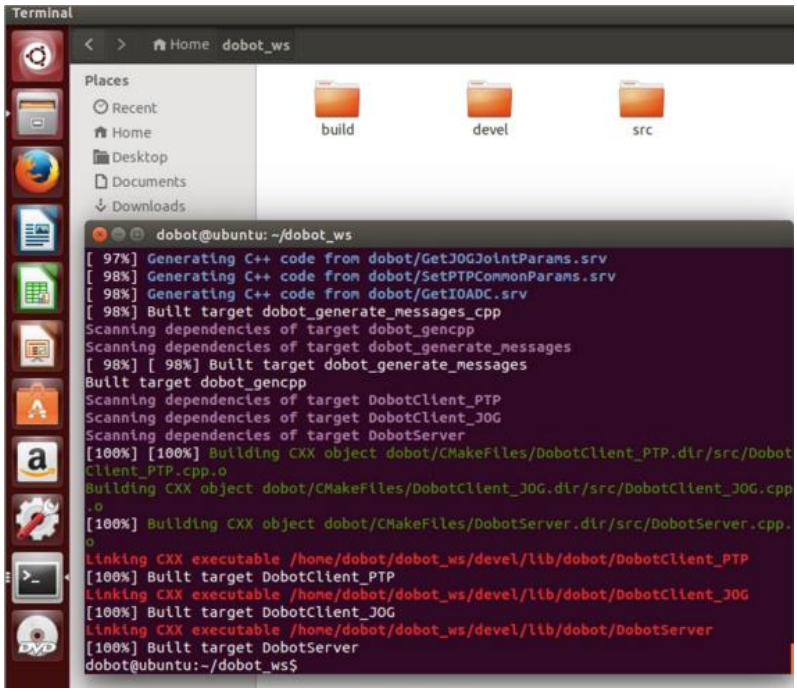

图 1.5 编译完成

步骤 5 添加dobot_ws环境变量到~/.bashrc文件中。

```
$ echo "source /home/dobot_ws/devel/setup.bash " >> ~/.bashrc
$ source ~/.bashrc
```

步骤 6 启动ROS。

```
$ roscore
```

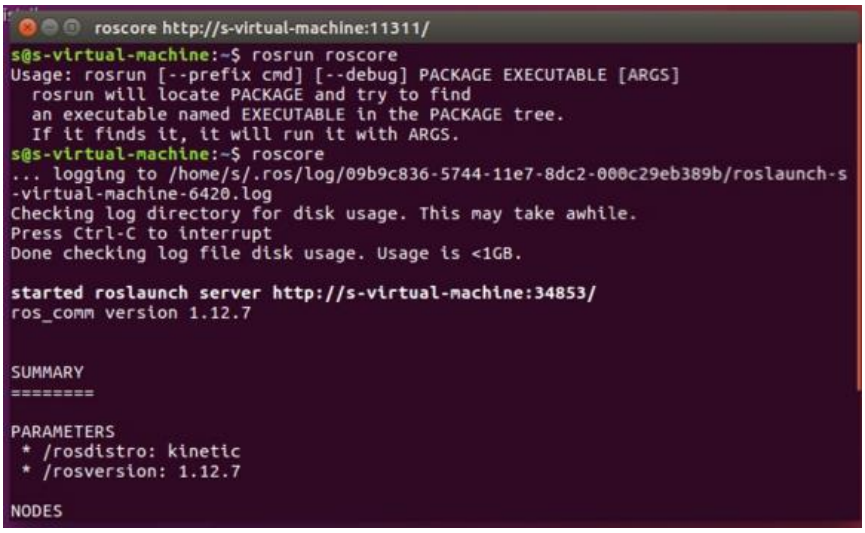

图 1.6 启动 ROS

步骤 7 重新打开一个终端作为服务端节点，使用rosrun命令运行dobot包中的DobotServer指令，即连接Dobot。

其中，*Port*为Dobot使用的串口，可执行**ls /dev -l**命令查看实际串口，USB串口命名形式为“ttyUSB*X*”。*X*表示串号号，请根据实际情况替换。

```
$ rosrun dobot DobotServer Port
```

如果连接Dobot Magician成功，则如图 1.7所示。

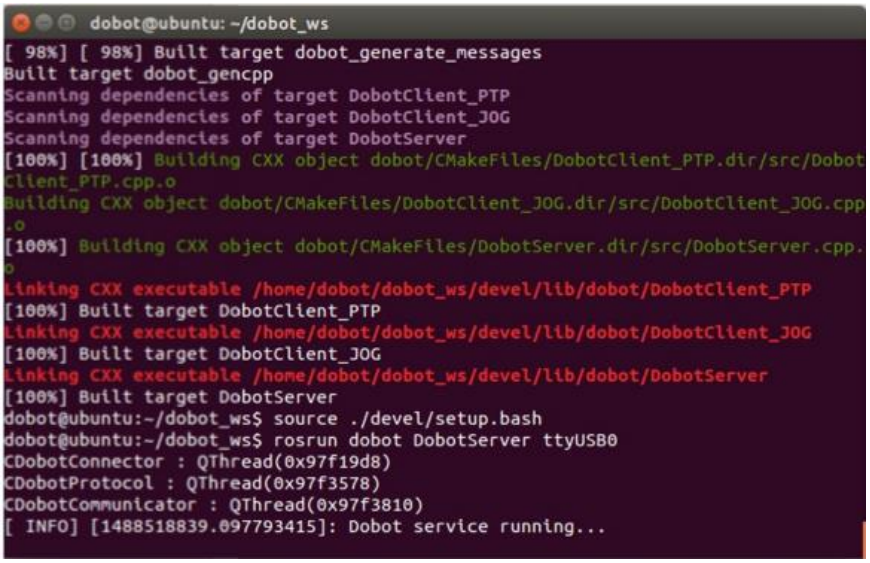

图 1.7 Dobot Magician 连接成功

如果执行指令后出现乱码，则可能是连接的串口不对，请执行**ls /dev -l**命令查看实际串口。

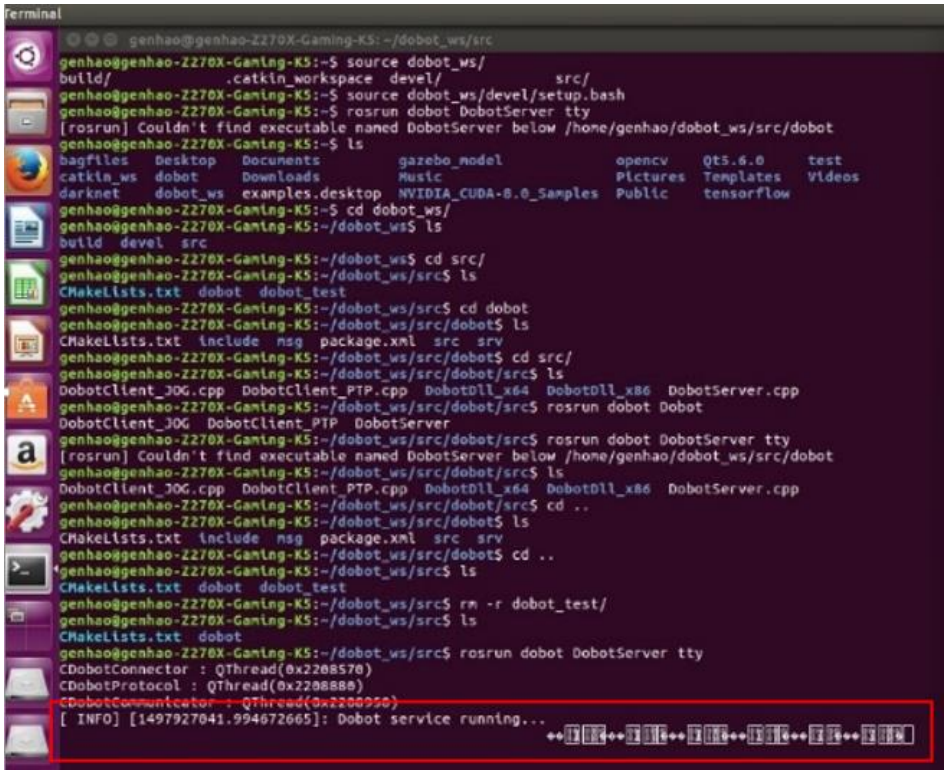

图 1.8 串口连接错误

步骤 8 DobotServer成功运行后，重新打开一个终端作为客户端节点，使用rosrun命令运行dobot包中的DobotClient_JOG指令。

执行成功后，可以使用键盘控制Dobot，如表 1.2所示

表 1.2 按键说明

按键	方向
W	向前
S	向后
A	向左
D	向右
U	向上
I	向下
J	逆时针旋转
K	顺时针旋转
其他按键	停止

```
$ rosrun dobot DobotClient_JOG
```

如果执行点动命令后，按“W”、“A”、“S”、“D”等按键无任何反应，可能是键盘按键的ASCII码值与Demo中设置的ASCII码值不一致，可以在src文件更改相应按键的ASCII码值，如图 1.9所示.

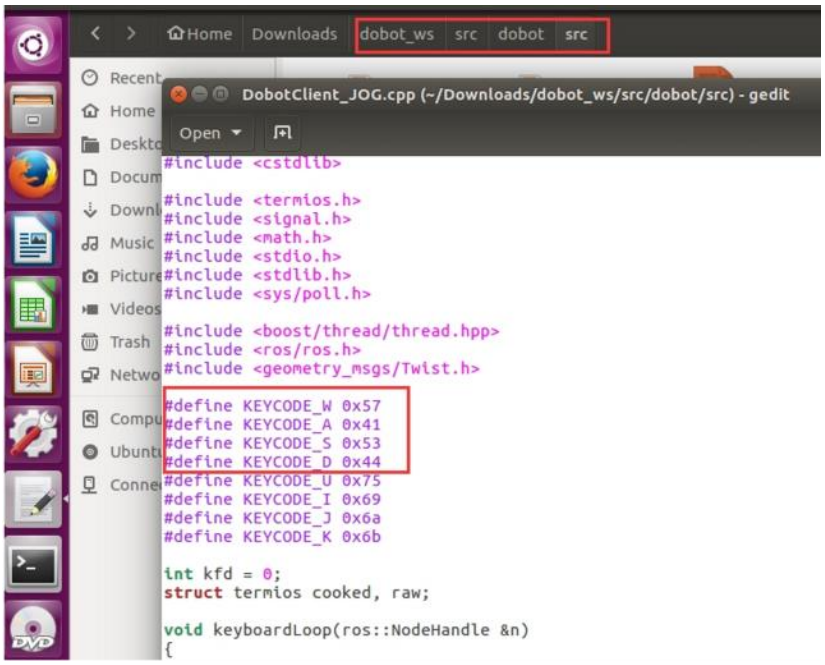

图 1.9 修改按键ASCII码

你也可以在客户端按Ctrl＋C停止JOG程序后，测试其他指令，例如执行PTP指令。

```
$ rosrun dobot DobotClient_PTP
```

执行该指令后，Dobot Magician会自动沿X轴来回运动。

1.2.2 代码说明

- 本示例采用指令队列模式。Demo的目录结构如图 1.10所示。

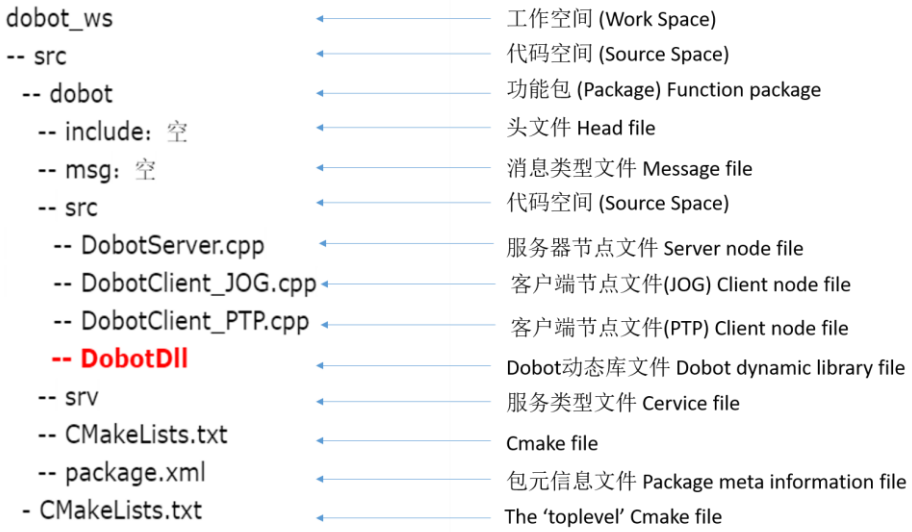

图 1.10 Demo 目录结构

- 在ROS中常用的通信方式有四种：
  - Topic
  - Service
  - Parameter Service
  - Actionlib

  本示例采用Service服务(查询式的通信模型)。Service通信是双向的，采用Request/Reply模式，即通过“请求/应答”机制完成服务通信，其通信方式如图1.11所示。

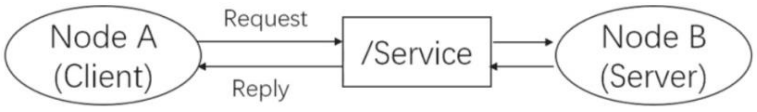

图 1.11 Service 服务通信方式

其中，Node A作为客户端，Node B作为服务端，并提供Service接口。客户端发送Request后，服务端处理Request并给出一个Reply，客户端会收到该Reply。该方式可以实现客户端的阻塞，也能保证消息通信的反馈。

- 本示例中的Service服务编程流程如图 1.12所示。

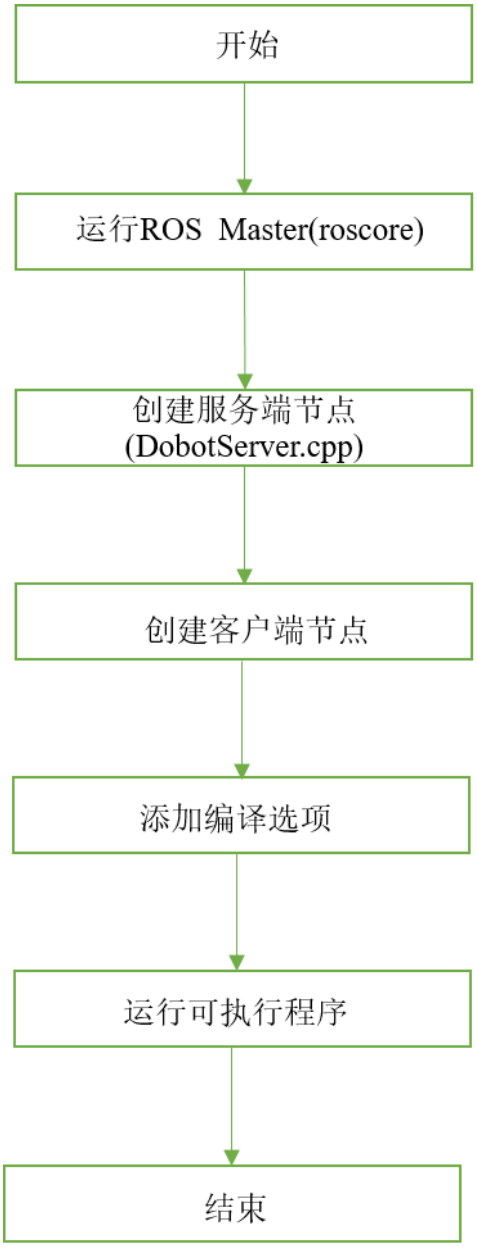

图 1.12 Service 服务编程流程

- 创建服务端文件(DobotServer.cpp)流程图如图 1.13所示。

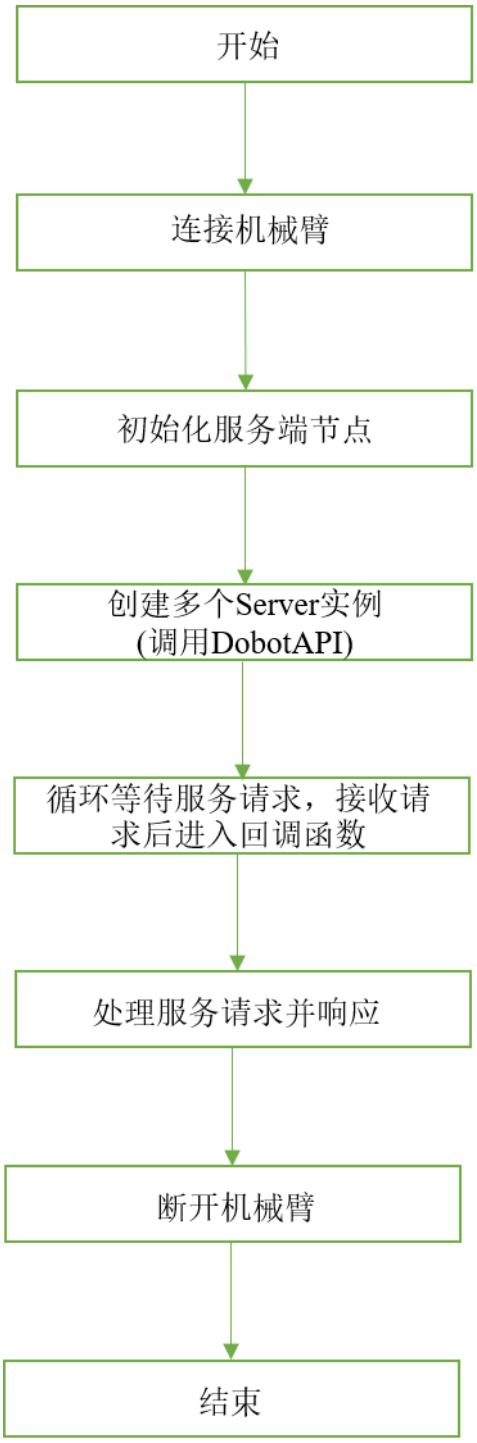

图 1.13 创建服务端流程

（1） 加载 ROS 和 Dobot 动态库。

程序 1.1 加载 ROS&DobotDLL 头文件

```
#include "ros/ros.h"     //加载 ROS 头文件
#include "std_msgs/String.h“
```

```
#include "std_msgs/Float32MultiArray.h“
#include "DobotDll.h"              //加载 Dobot 动态库头文件
```

（2） 连接机械臂。

程序 1.2 连接机械臂

```
if (argc < 2)
    {
        ROS_ERROR("[USAGE]Application portName");
        return -1;
    }
     printf("---------%s",argv[1]);
    //根据获取的串口信息连接机械臂
    int result = ConnectDobot(argv[1], 115200, 0, 0);
```

（3） 初始化服务端节点。

程序 1.3 初始化服务端节点

```
ros::init(argc, argv, "DobotServer");   //初始化服务端节点并命名为 DobtoServer
ros::NodeHandle n;        //创建节点句柄
```

（4） 创建 Server 实例。

程序 1.4 创建多个 Server 实例

```
    //创建 Server 容器
    std::vector<ros::ServiceServer> serverVec;
    //初始化 Server 实例，并在对应的初始化函数中注册回调函数
    InitCmdTimeoutServices(n, serverVec);
    InitDeviceInfoServices(n, serverVec);
    InitPoseServices(n, serverVec);
    InitAlarmsServices(n, serverVec);
    InitHOMEServices(n, serverVec);
    InitTRIGServices(n, serverVec);
    InitEIOServices(n, serverVec);
    InitQueuedCmdServices(n, serverVec);
    … ....
    void InitQueuedCmdServices(ros::NodeHandle &n, std::vector<ros::ServiceServer> &serverVec)
    {
```

```
ros::ServiceServer server;
//注册回调函数
server = n.advertiseService("/DobotServer/SetQueuedCmdStartExec", SetQueuedCmdStartExecService);
serverVec.push_back(server);
server = n.advertiseService("/DobotServer/SetQueuedCmdStopExec", SetQueuedCmdStopExecService);
serverVec.push_back(server);
…. ….
}
```

（5） 循环等待服务请求消息，待接收到消息后进入回调函数。

程序 1.5 进入回调函数

```
ROS_INFO("Dobot service running...");
ros::spin();
```

（6） 处理服务请求并响应。

程序 1.6 处理服务请求并响应

```
bool SetQueuedCmdStartExecService(dobot::SetQueuedCmdStartExec::Request &req,
dobot::SetQueuedCmdStartExec::Response &res)
{
    res.result = SetQueuedCmdStartExec();
    return true;
}
```

（7） 断开机械臂。

程序 1.7 断开机械臂

```
ROS_INFO("Dobot service exiting...");
DisconnectDobot();
```

- 创建客户端文件(以DobotClient_PTP.cpp为例)流程图如图 1.14所示。

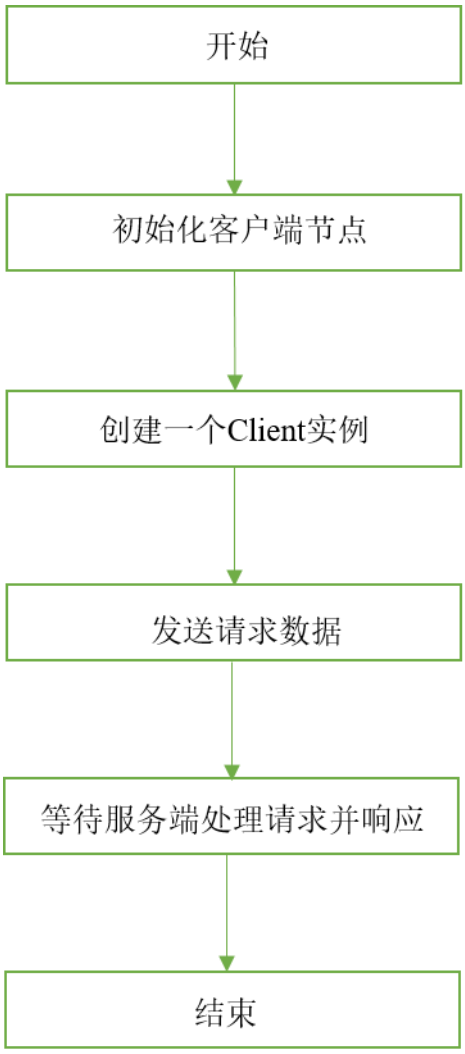

图 1.14 创建客户端流程

（1） 初始化客户端节点。

程序 1.8 初始化客户端节点

```
ros::init(argc, argv, "DobotClient");  //初始化客户端节点并命名为 DobtoClient
ros::NodeHandle n;    //创建节点句柄
```

（2） 创建 Client 实例，发出服务请求，等待服务端处理请求并响应。

程序 1.9 创建 Client 实例并完成服务请求处理

```
ros::ServiceClient client;  //创建 Client 实例：client
… …
// Set PTP common parameters
do {
```

```
        //创建 Client，请求 DobotServer 节点下的 SetPTPCommonParams 服务
        //服务消息类型为 dobot::SetPTPCommonParams
        client = n.serviceClient<dobot::SetPTPCommonParams>("/DobotServer/SetPTPCommonParams");
        dobot::SetPTPCommonParams srv;

        srv.request.velocityRatio = 50;
        srv.request.accelerationRatio = 50;
        //发送服务请求，等待 Server 处理并响应
        client.call(srv);
    } while (0);
```