Instruction for Preliminary Project 1

Ubuntu and ROS Installation

1. Introduction

You are going to install 'Ubuntu' and 'Robot Operating System (ROS)' for this term assignment. This instruction will provide a guide for installation. We assumed that your main operating system is Windows 10 and explained how to multi-boot Ubuntu 16.04 on Windows 10. If you are using other operating systems, please search the Internet about the installation. In the end of this instruction, there are two screenshots. You will submit your own results compressed on eTL. The file name will be '430.457_StudentNumber_Pre_Project1.zip'.

CAUTION: We highly recommend you to back up your data on Windows before installing Ubuntu. Ubuntu's version must be 16.04, don't install other versions.

2. To install Ubuntu

We'll use Ubuntu 16.04 LTS for the development environment. The installation process is as follows:

2.1 Make USB for booting

2.2 Reduce the size of partition for Windows

2.3 Install Ubuntu 16.04 LTS

2.4 set various environments after the installation

CAUTION: Before installing of Ubuntu, please check if you can divide your drive. Ubuntu needs more than 2 partitions.

2.1 Make USB for booting

To make an USB for booting Ubuntu, please empty the USB at first and install an image file of Ubuntu.

2.1.1 Download Ubuntu 16.04 LTS installation image file

Download an Ubuntu image file(*.iso). Link for Ubuntu 16.04 LTS 64bit is as follows:

https://www.ubuntu.com/download/desktop

2.1.2 Download Universal USB Installer

Use Universal USB Installer for this task. You can download it from the link below.

http://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/

2.1.3 Make booting USB

1. Insert the USB **(Be aware that all the files in the USB will be removed)**
2. Initialize the USB
3. Press Windows key + R and type DISKPART> **diskpart**
4. When Diskpart program execute, type DISKPART> **list disk**
5. Remember disk number of USB that you are going to initialize
6. Type select DISKPART> **select disk** (Number)
7. Type DISKPART> **clean** **(This task eliminate data inside the USB permanently)**
8. Type DISKPART> **create partition primary**
9. Type DISPART> **format quick fs=fat32**
10. Check whether the USB is recognized in My Computer and memorize the letter

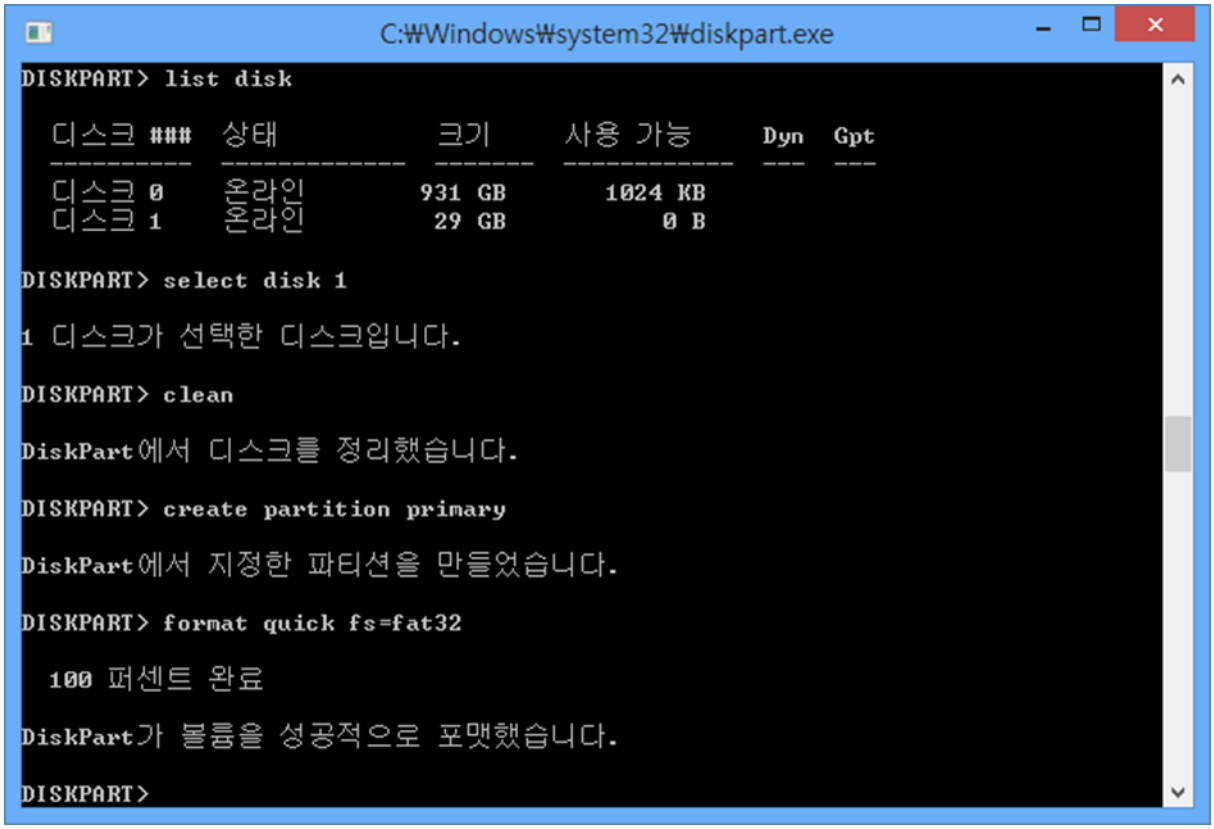

Figure 1 Prior setting for making booting USB

11. Make booting USB using Universal USB Installer
12. Run Universal USB installer
13. Select Ubuntu, Image file, USB Drive letter (Check in the My Computer)
14. Press 'Create' button to start making booting USB
15. Press 'Yes' to continue
16. Booting USB making is completed

2.2 Reduce the size of partition for Windows

To secure space for installation of Ubuntu, you need to reduce the size of partition for Windows

1. Widows key + R and type **compmgmt.msc**
2. Select a partition that you want to reduce
3. Right click on the selected partition and click 'Reduce Volume(볼륨축소)'
4. Set volume to reduce (same as volume for Linux) and press 'Reduce'
5. Check the result after the task is done

2.3 Install Ubuntu 16.04 LTS

We are going to use booting USB made in the previous section. Turn off the computer and insert USB and turn on the computer. Press F2 or Del to enter the BIOS/EFI when the logo of computer maker shows on the screen. After entering the BIOS/EFI, find out Boot Priority menu and move USB to the top.

After boot priority setup, the computer will boot from the USB and you can see the black screen with the white icon at the bottom. Press any key to move next and select your language and select Install Ubuntu(I).

When you see the welcome screen, select your language and press continue. You don't need to connect to the internet. Press continue until you encounter installation mode(설치 형식). Select etc(기타) and move to the next. If you follow the instruction well, you can see the screen like Figure 2.

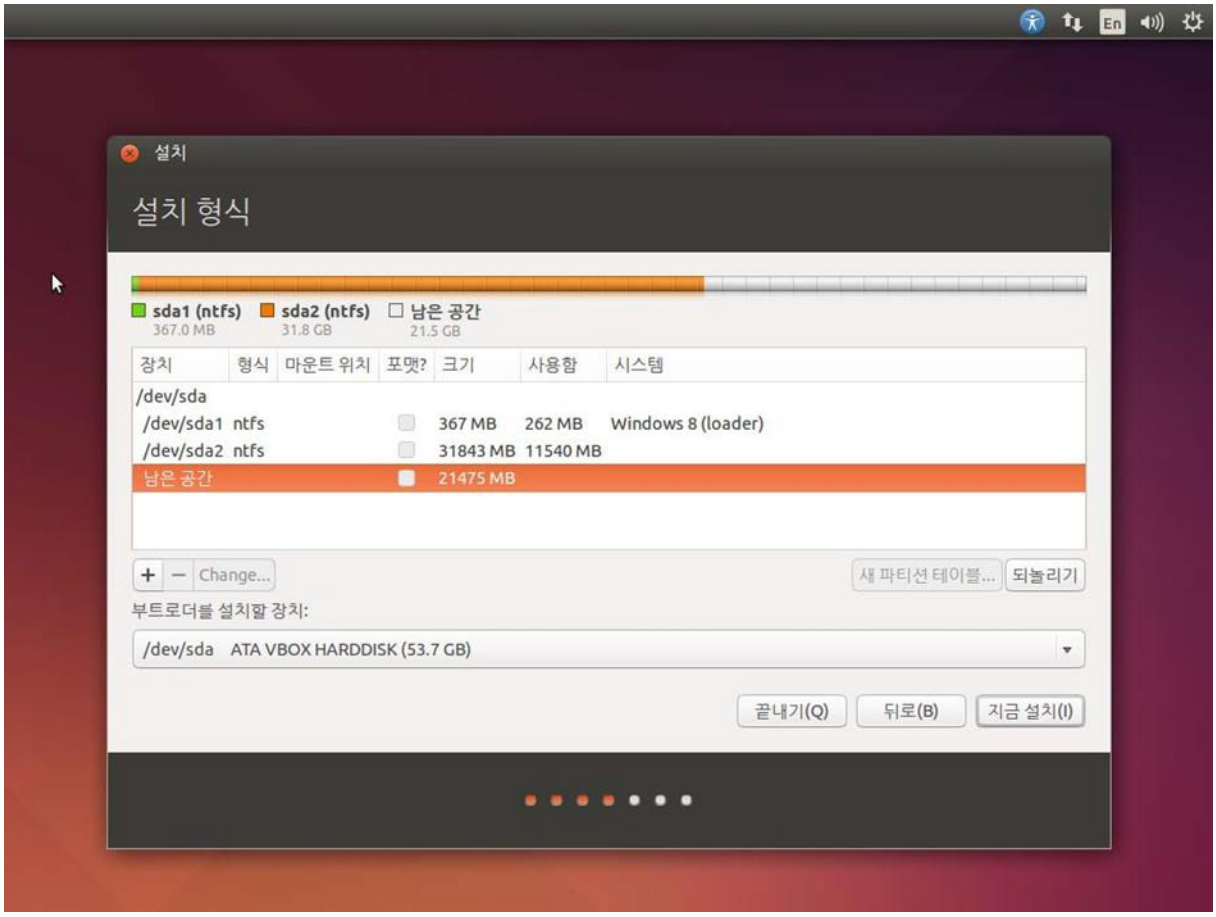

Figure 2 Installation Mode of Ubuntu 16.04 LTS

The bar graph shows the volume of the empty space in which the Ubuntu can be installed. You have to allocate two partitions: root partition and swap partition.

Allocation of root partition: Click 'Remaining Space' → Click '+' → Set volume(크기) → Select 'Primary partition' → select 'The starting point of this space' → Purpose: EXT4 Journaling file system → Mount point: /

Allocation of swap partition: Click '+' → Set volume(크기) → Select 'swap area' → select 'The starting point of this space'

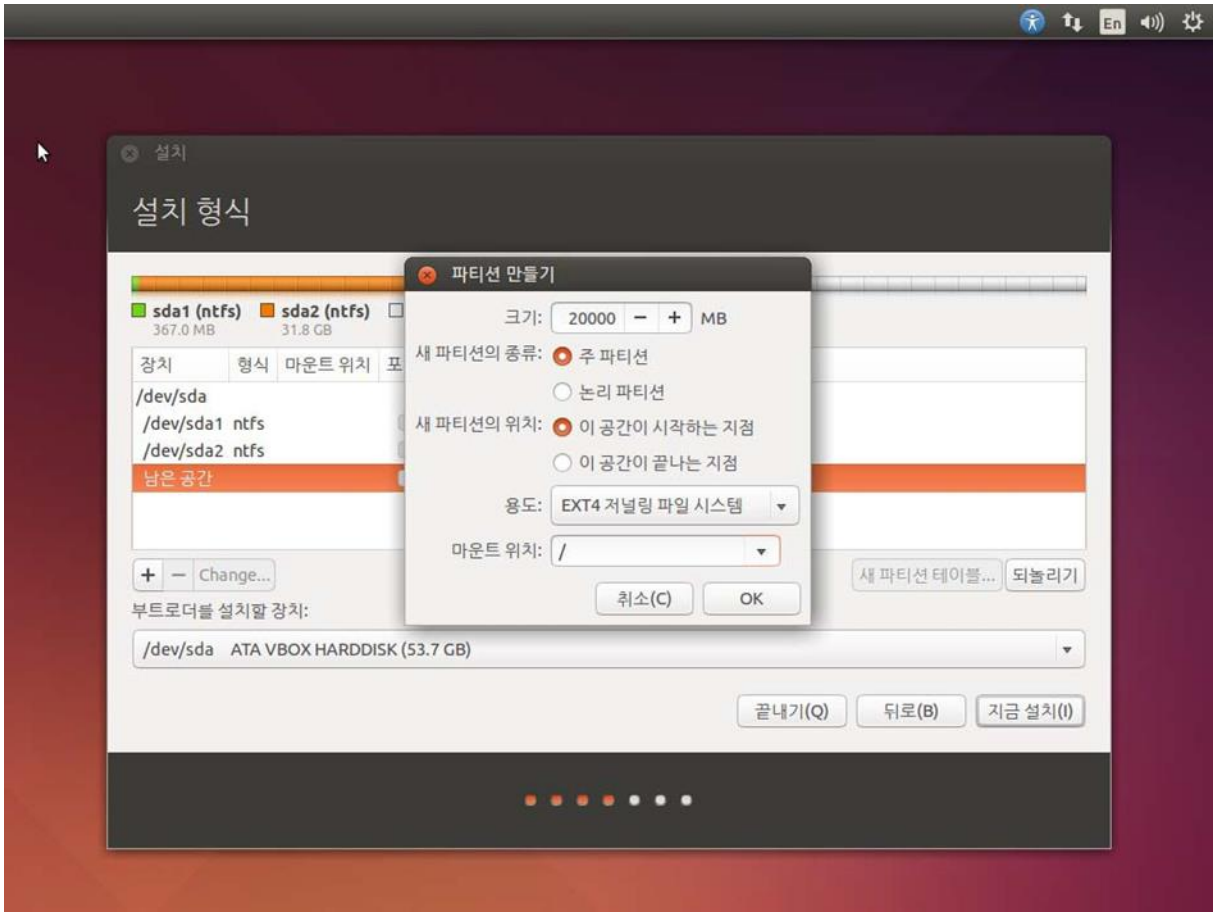

Figure 3 Root partition allocation for Installation of Ubuntu 16.04 LTS

After allocating partition, click 'Install now' button. You will select city, language, set user name and password in turn. You can see the screen like Figure 4.

CAUTION: Please set your user name as your name that TA can recognize you when you submit screenshots as an assignment.

Figure 4 Proceeding installation

After installation, the Computer will start rebooting. GRUB boot loader will show at the screen. Select the Ubuntu.

3. ROS installation

Most of this section referred to a tutorial in ROS wiki (http://wiki.ros.org/kinetic/Installation/Ubuntu). If you don't understand this instruction or have any problem, see the tutorial. This instruction follows the tutorial from section 3.1.

3.1 Set up your sources.list

Setup your computer to accept software from packages.ros.org. ROS Kinetic **ONLY** supports Wily (Ubuntu 15.10), Xenial (Ubuntu 16.04) and Jessie (Debian 8) for Debian packages.

Ubuntu 16.04 (Xenial)

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

3.2 Set up your keys

```
wget https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -O - | sudo apt-key add -
```

3.3 Installation

First, make sure your Debian package index is up-to-date:

```
sudo apt-get update
```

Desktop-Full Install: ROS, rqt, rviz, robot-generic libraries, 2D/3D simulators, navigation and 2D/3D perception

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

3.4 Initialize rosdep

Before you can use ROS, you need to initialize rosdep. rosdep enables you to easily install system dependencies for source you want to compile and is required to run some core components in ROS.

```
sudo rosdep init
rosdep update
```

3.5 Environment setup

It's convenient if the ROS environment variables are automatically added to your bash session every time a new shell is launched:

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc

source ~/.bashrc
```

If you have more than one ROS distribution installed, ~/.bashrc must only source the setup.bash for the version you are currently using.

If you just want to change the environment of your current shell, type this:

```
source /opt/ros/kinetic/setup.bash
```

3.6 Getting rosinstall

rosinstall is a frequently used command-line tool in ROS that is distributed separately. It enables you to download many source trees easily for ROS packages with one command.

To install this tool on Ubuntu, run:

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

3.7 Create a ROS Workspace

Let's create a catkin workspace:

```
$ mkdir -p ~/catkin_ws/src

$ cd ~/catkin_ws/src

$ catkin_init_workspace
```

Even though the workspace is empty, you can still "build" the workspace.
(There are no packages in the 'src' folder. There's just a single CMakeLists.txt link)

```
$ cd ~/catkin_ws/

$ catkin_make
```

The catkin_make command is a convenience tool for working with catkin workspaces. Before continuing, source your new setup.*sh file:

```
$ source devel/setup.bash
```

You have to source this setup.*sh file when you want to use this workspaces you made above. For convenience, you can make the ROS environment variables automatically added to your bash session every time a new shell is launched: (**Recommended**)

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

4. "Hello World" example

4.1 creating a catkin Package

This tutorial will demonstrate how to use the catkin_create_pkg script to create a new catkin package, and what you can do with it after it has been created.

First change to the source space directory of the catkin workspace you created in the Creating a Workspace for catkin tutorial:

```
# You should have created this in the Creating a Workspace Tutorial

$ cd ~/catkin_ws/src
```

Now use the catkin_create_pkg script to create a new package called 'beginner_tutorials' which depends on std_msgs, roscpp, and rospy:

```
$ catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
```

This will create a beginner_tutorials folder which contains a package.xml and a CMakeLists.txt, which have been partially filled out with the information you gave catkin_create_pkg.

catkin_create_pkg requires that you give it a package_name and optionally a list of dependencies on which that package depends:

```
# This is an example, do not try to run this

# catkin_create_pkg <package_name> [depend1] [depend2] [depend3]
```

4.2 Writing Node

```
cd ~/catkin_ws/src/beginner_tutorials/src
```

This directory will contain any source files for our beginner_tutorials package.

Create the src/hello.cpp file within the beginner_tutorials package and paste the following inside it:

```
#include "ros/ros.h"

#include <stdio.h>


int main(int argc, char** argv)

{

        ros::init(argc,argv,"hello");
        ros::NodeHandle n;
        printf("hello!! World!!₩n");
        ros::spinOnce();

        return 0;

}
```

4.3 Building your nodes

You used catkin_create_pkg in a previous tutorial which created a package.xml and a CMakeLists.txt file for you. simply add these few lines to the bottom of your CMakeLists.txt:

```
add_executable(hello src/hello.cpp)

target_link_libraries(hello ${catkin_LIBRARIES})
```

This line will create an executable, hello, which by default will go into package directory of your devel space, located by default at ~/catkin_ws/devel/lib/<package name>.

Now run catkin_make:

```
# In your catkin workspace

$ cd ~/catkin_ws

$ catkin_make
```

4.4 Run “Hello World”

roscore is the first thing you should run when using ROS.

Please run:

```
$ roscore
```

rosrun allows you to use the package name to directly run a node within a package (without having to know the package path). Usage:

```
$ rosrun [package_name] [node_name]
```

So now we can run the hello in the beginner_tutorials package.

Then, in a **new terminal**:

```
$ rosrun beginner_tutorials hello
```

5. Submission Format

Submit your screenshots of execution of roscore and hello node separately using eTL. Figure 5 and Figure 6 are examples of the submission format.

Due to: 2017.09.20. 23:59 KST

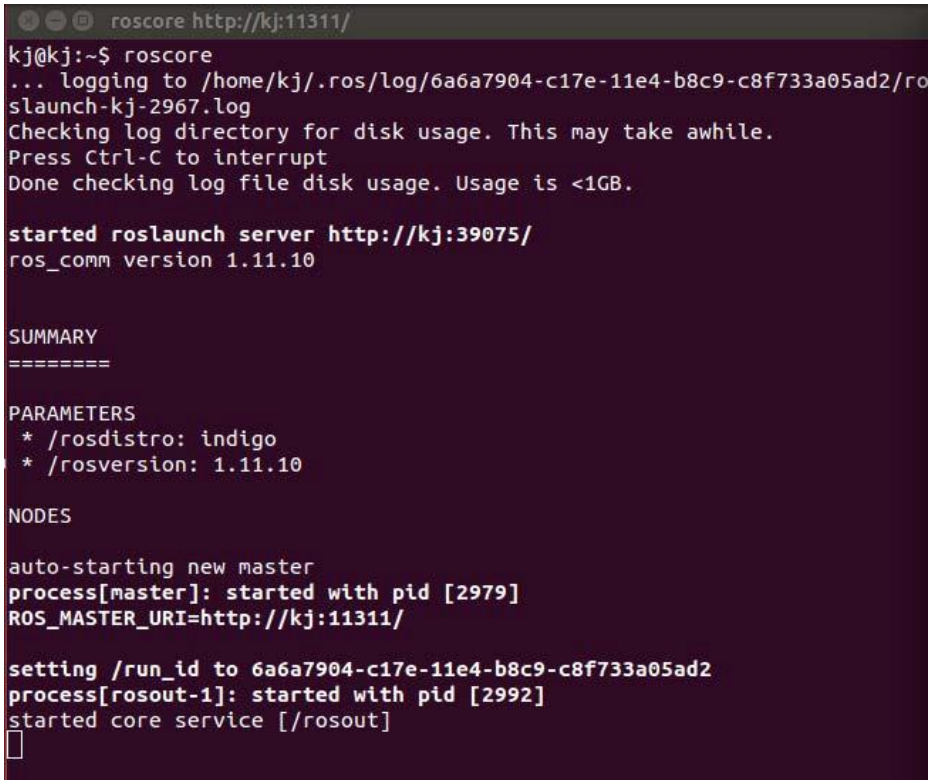

Figure 5 Submission Format (1)

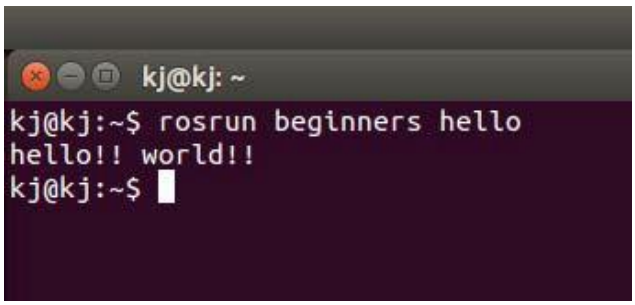

Figure 6 Submission Format (2)