Xilinx Development Environment Installation Tutorial

ALINX

Version Record

Version	Date	Release By	Description
Rev1.0	2022-04-30	Rachel Zhou	First Release

The ALINX official Documents are in Chinese, and the English version was translated by **Shanghai Tianhui** Trading Company. They has **not been officially Review by ALINX** and are for reference only. If there are any errors, please send email feedback to support@aithtech.com for correction.

Amazon Store: https://www.amazon.com/alinx

Aliexpress Store:

https://alinxfpga.aliexpress.com/store/911112202?spm=a2g0o.detail.1000007.1.704e2bedqLBW90

Ebay Store: https://www.ebay.com/str/alinxfpga

Customer Service Information

Skype: rachelhust@163.com

Wechat: rachelhust

Email: support@aithtech.com and rachehust@163.com

Content

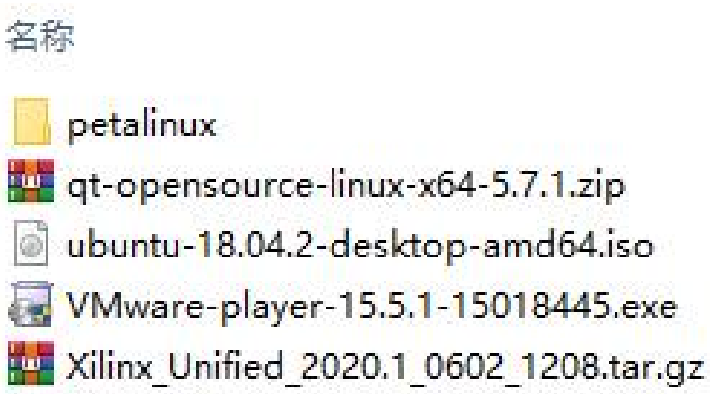
Version Record 2
Content 3
Preparation and Precautions 6
Windows Chapter 6
Part 1: Vitis Installation 7
Part 1.1: Vitis Software Version 7
Part 1.2: Vitis software installation under Windows 8
Part 1.3: Re-download the driver to install the driver 16
Linux Chapter 18
Part 2: Install virtual machine and Ubuntu system 18
Part 2.1: Virtual machine software installation 18
Part 2.2: Ubuntu installation 20
Part 2.2.1: install the system 20
Part 2.2.2: Modify the Software Source Server 29
Part 2.2.3: Set bash to default sh 31
Part 2.2.4: Set screen lock time 32
Part 2.3: Q&A 32
Part 2.3.1: Virtual machine requires virtualization support 32
Part 3: Install Linux version of Vitis software on Ubuntu 34
Part 3.1: Install Vitis for Linux 34
Part 3.2: Permission settings 39
Part 3.3: Add license 39
Part 3.4: Install the downloader driver 41
Part 3.5: Test Vivado 42
Part 3.6: Q&A 44

Part 3.6.1: Prompt is occupied when downloading Linux downloader 44
Part 3.6.2: Cross compiler suitable for ZYNQ 46
Part 3.6.3: Cross compiler suitable for zynqMP 46
Part 4: Petalinux tool installation 47
Part 4.1: Petalinux Introduction 47
Part 4.2: Install the necessary libraries 47
Part 4.3: Install Petalinux 52
Part 5: NFS service software installation 56
Part 5.1: Install the NFS service 56
Part 5.2: Testing NFS 58
Part 5.3: Q&A 58
Part 5.3.1: NFS cannot be mounted 58
Part 6: QT Creator development environment 61
Part 6.1: QT Creator installation 61
Part 6.2: hello world in Qt Creator 69
Part 6.2.1: Create project 69
Part 6.2: Project compilation 71
Part 7: Linux Common Commands 73
Part 7.1: File and folder operations 73
Part 7.1.1: Creat a new folder 73
Part 7.1.2:Create a folder for multiple levels of directories 73
Part 7.1.3: Change current directory 73
Part 7.1.4: Copy file 73
Part 7.1.5:File Rename 73
Part 7.1.6:Delete files and folders 73
Part 7.2: Document List 74

Part 7.3: Compression and Decompression 74
Part 7.3.1: Create a Compressed File Package 74
Part 7.3.2: Extract a “tar.gz” File 74
Part 7.4: System Management 74

Preparation and Precautions

The software package and the tutorial are downloaded separately. If you have not downloaded the software package, please download the software package. The name of the Vivado and Vitis installation package is **Xilinx_Unified_2020.1_0602_1208.tar.gz**. Windows and Linux are unified installation packages, and WinRar is used to decompress under Windows.



Windows Chapter

The software environment introduced in this chapter is applicable to:

1. Use Vivado for FPGA development under Windows
2. Use Vitis to develop zynq and zynqMP bare metal software under Windows

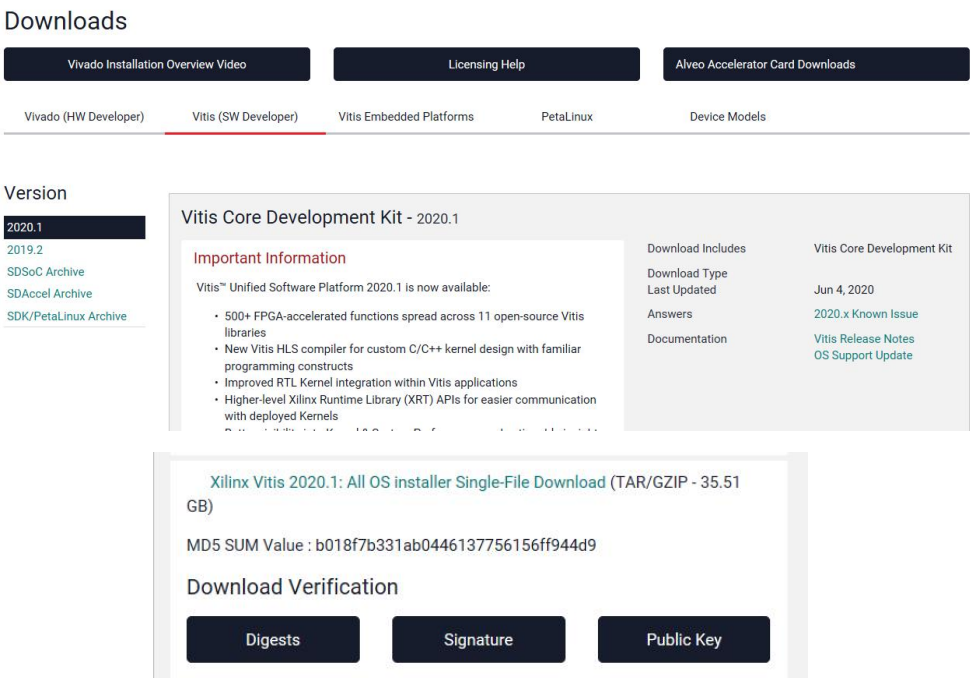
Part 1: Vitis Installation

Part 1.1: Vitis Software Version

The software version of Vitis is constantly being upgraded. So far, the latest software version is 2020.1. Because all the routines and tutorials of the ZYNQ development board are completed in the development environment of Vitis2020.1. In order to avoid some unexplainable problems caused by the software version, I still hope that everyone will keep pace with us during the learning process. The user needs to install the software of Vitis 2020.1 before using it. Because the Vivado software is relatively large, we did not provide CD installation files, only download links. In addition, users can also download from Xilinx's official website. To download from the official website, you need to register for the relevant account.

Xilinx official download address of Vivado software:

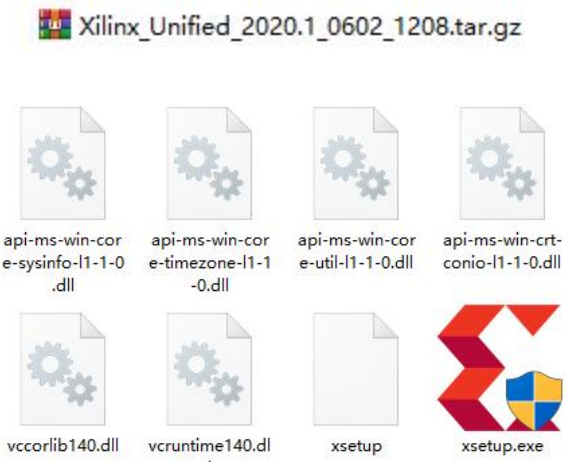
https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vitis.html



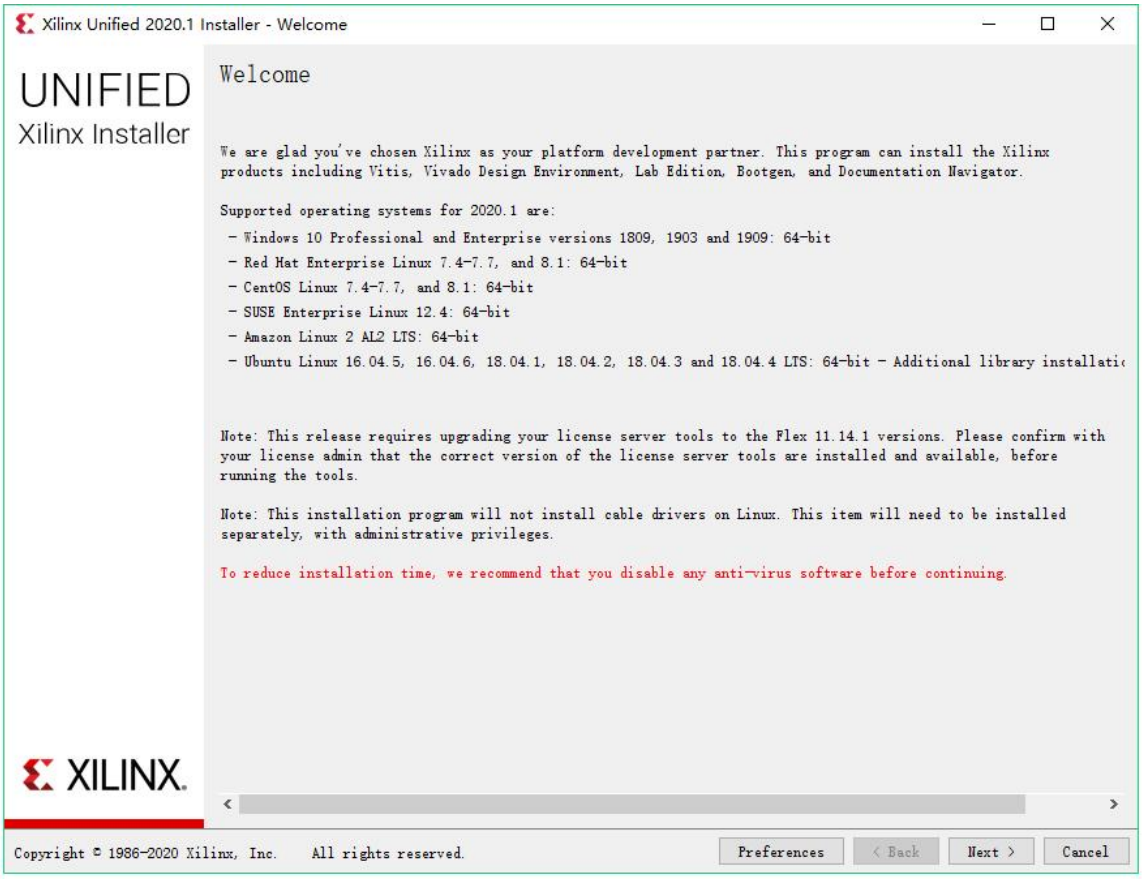
Vitis provides a Linux version and a Windows version, as well as a two-in-one version. We use the two-in-one version here, which can meet both Windows development and Linux development. Vitis requires the operating system to be 64-bit. Note: The Vitis installation package includes the Vivado software.

Part 1.2: Vitis software installation under Windows

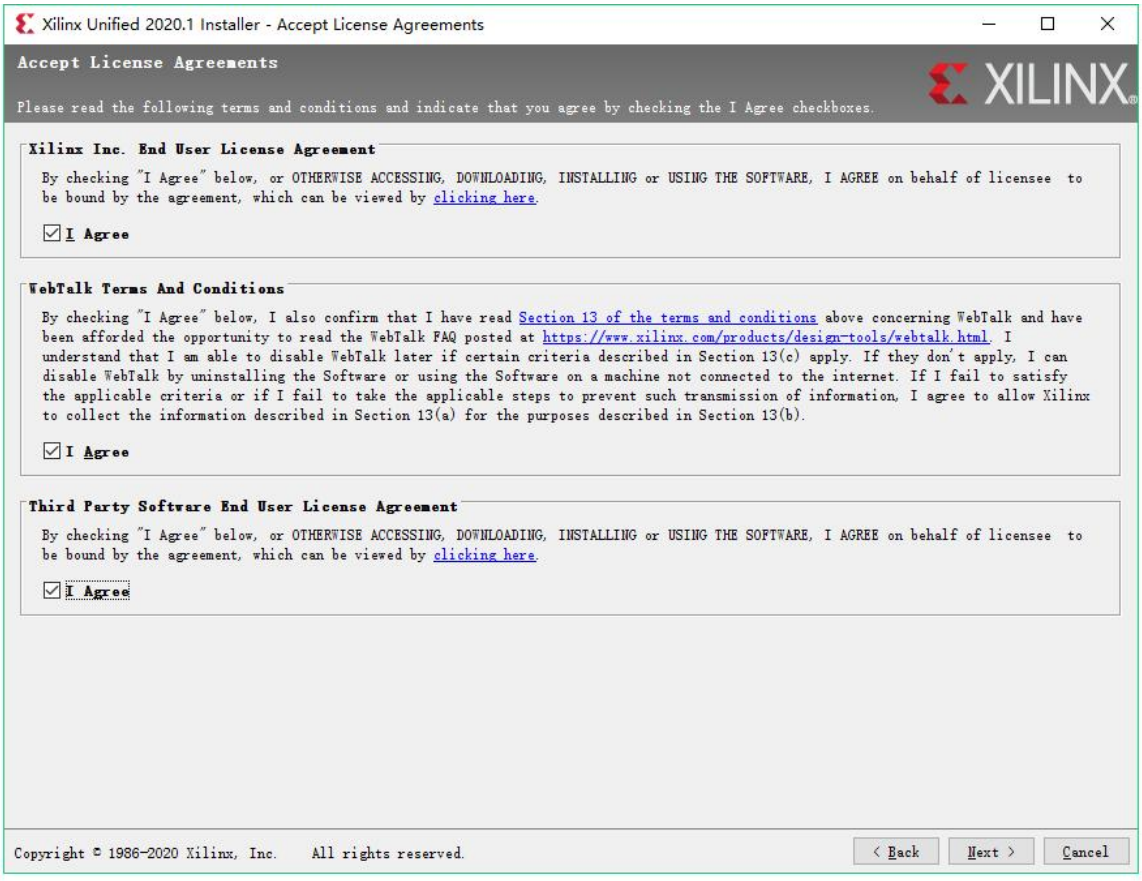
1) Download and decompress the Vivado software compression package, and click xsetup.exe directly to enter the installation, but for better installation, please close the anti-virus software, various computer housekeepers, and the computer user name should not have Chinese characters or spaces



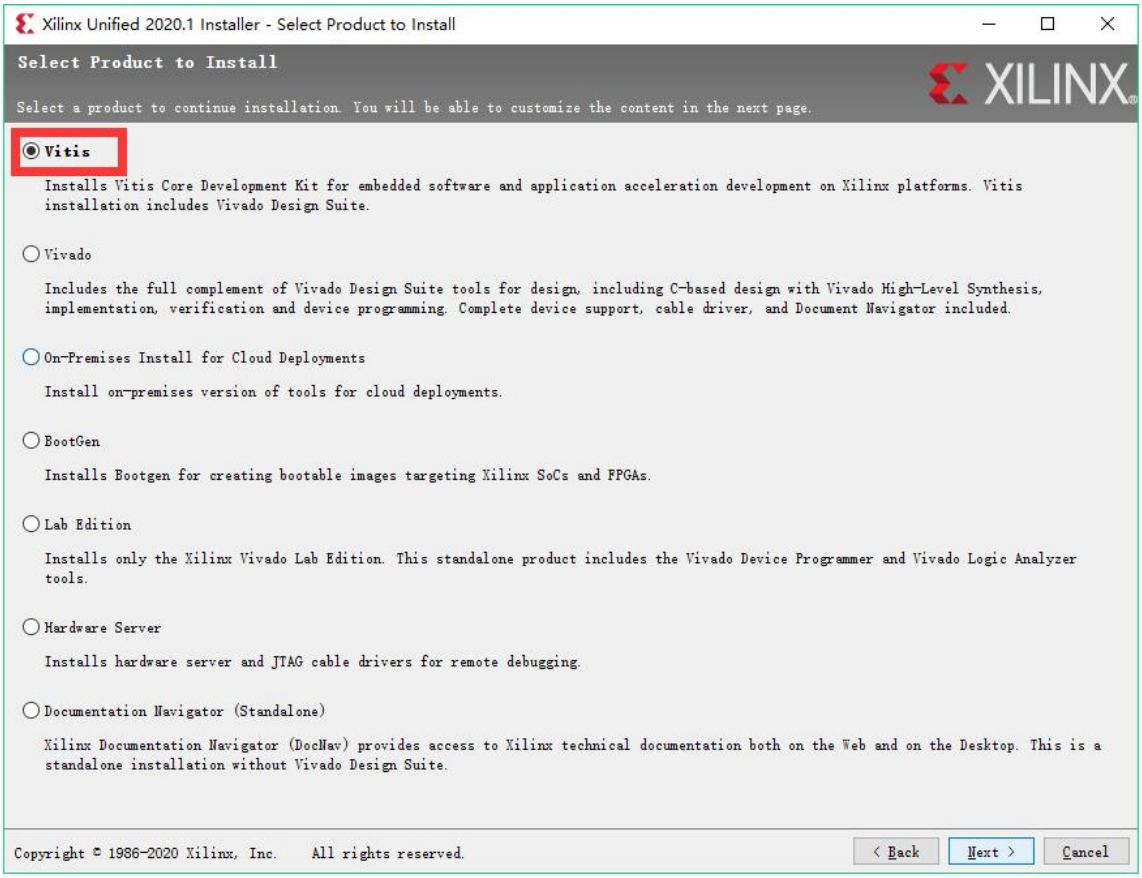
2) If prompted to update the version, ignore the update and click "Continue"
3) Click "next" to install, you can see Vivado's requirements for the system



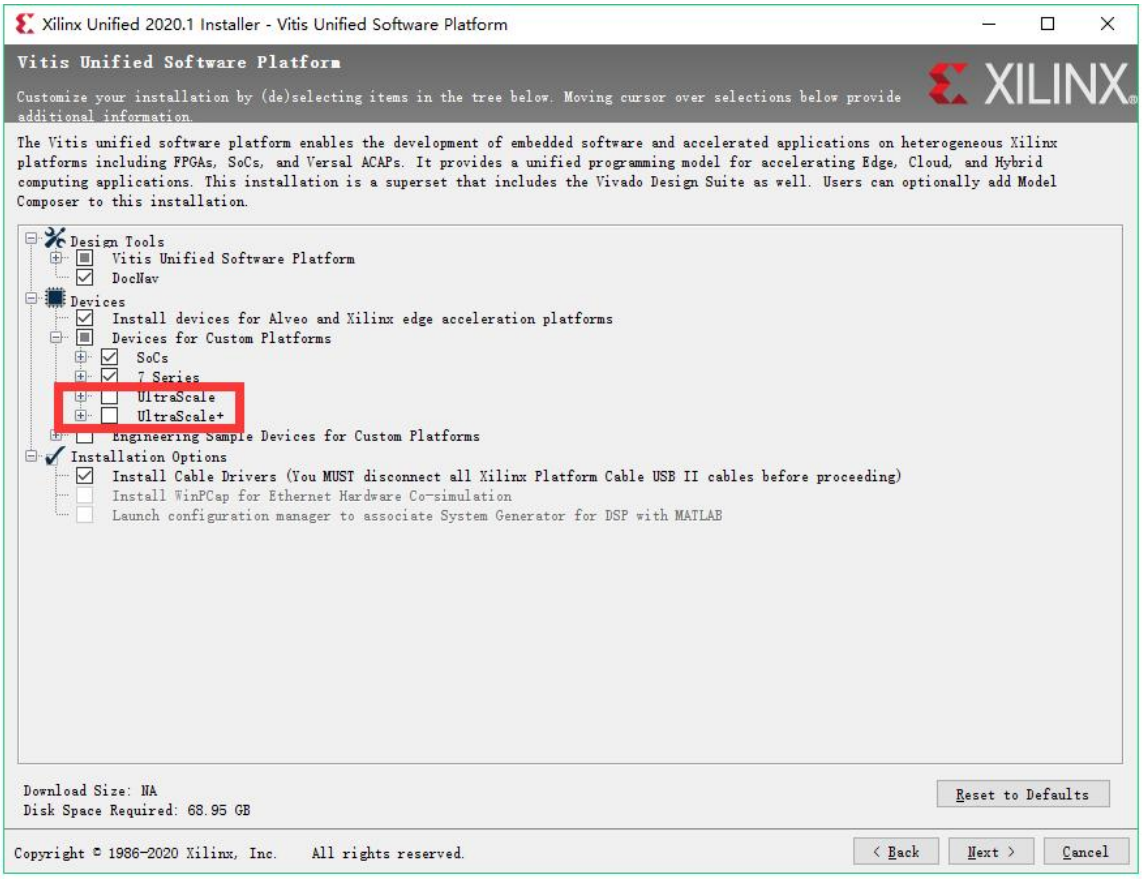
4) Click "I Agree" to accept the terms



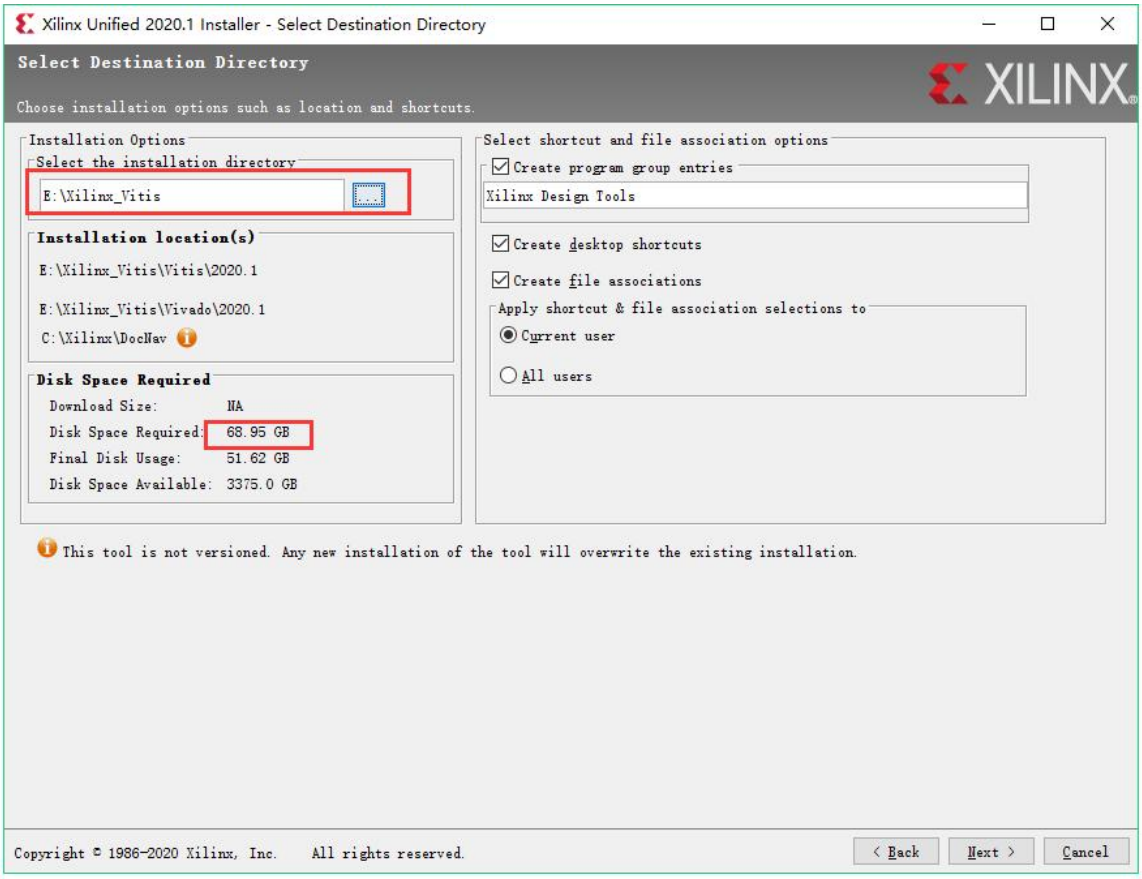
5) Choose Vitis to install



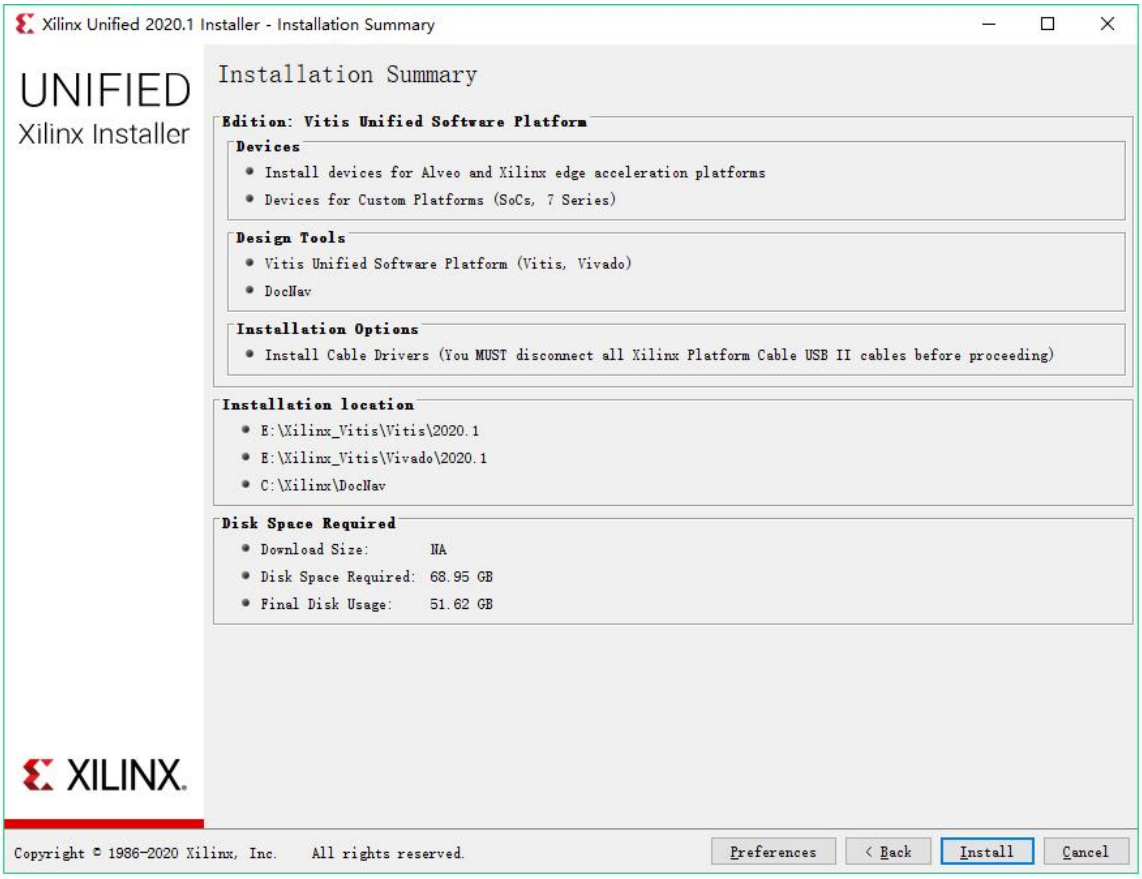
6) Choose the device library to install here. Since we don't need UltraScale and UltraScale+ chips, you can uncheck to save installation space. Keep the others as default, click "next"



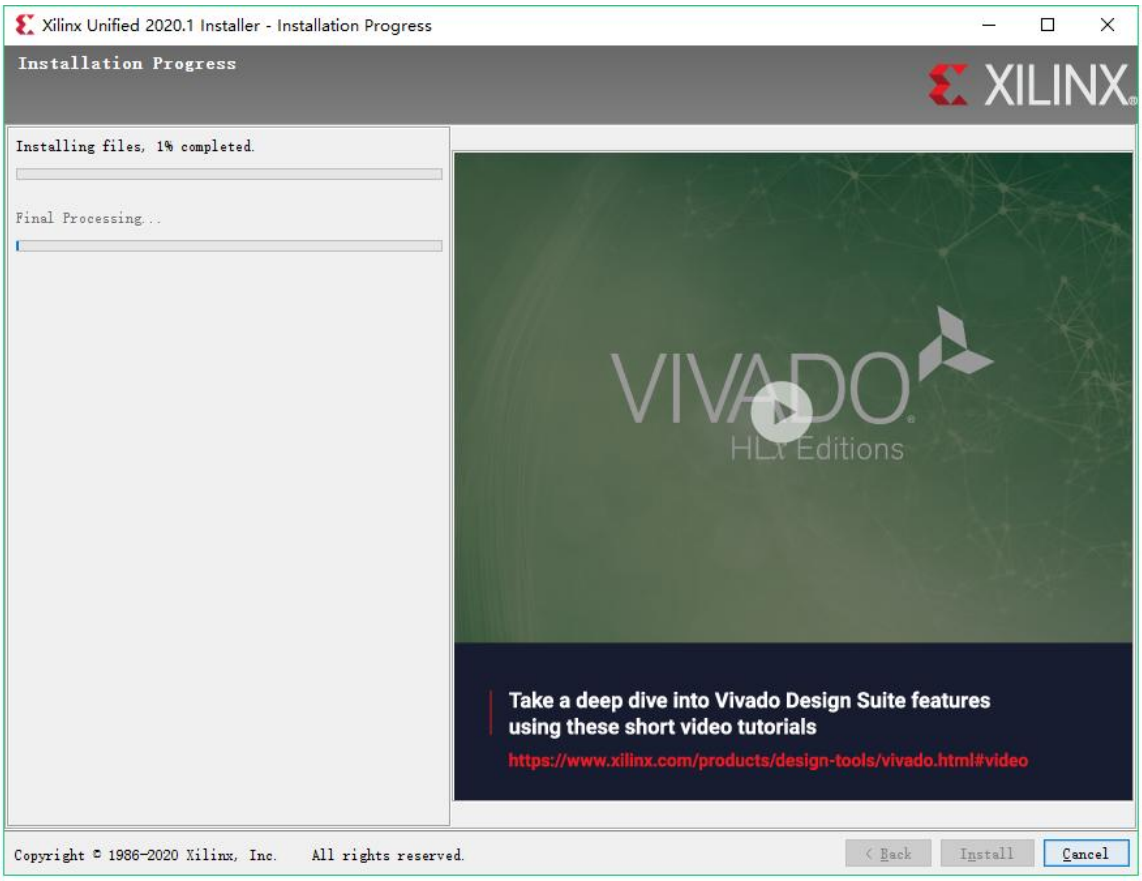
7) The installation path has not been modified here. The installation path cannot have special characters such as Chinese and spaces. At the same time, the user name of the computer should not be a Chinese name with spaces. You can see that Vivado requires at least 80G for the size of the hard drive.



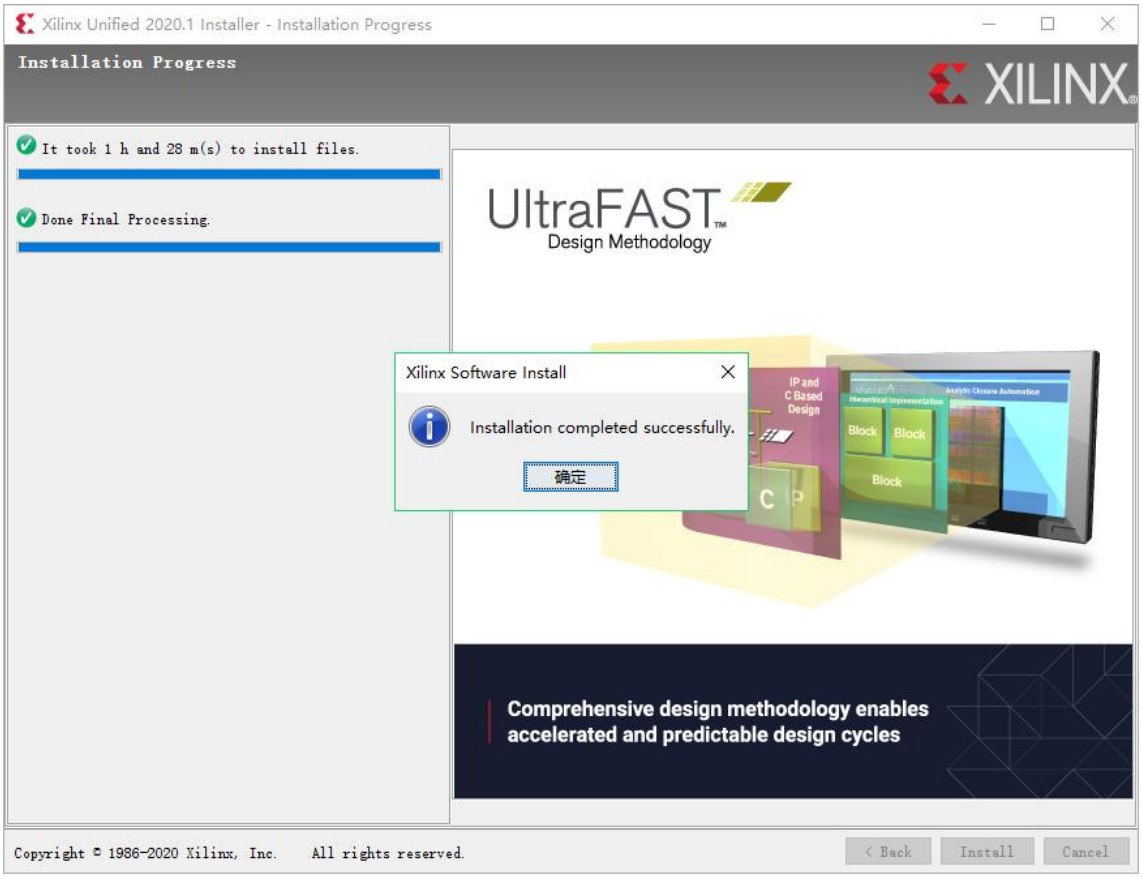
8) Click "Install"



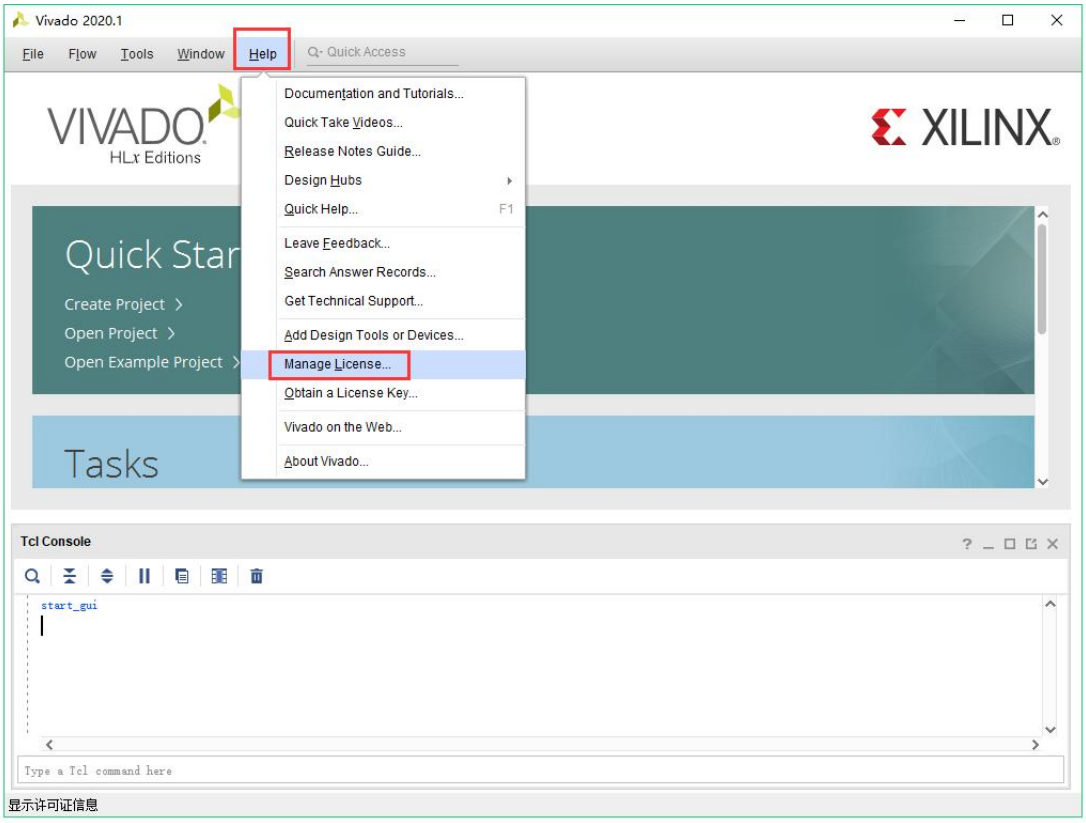
9) It takes a long time to wait for the installation. If the antivirus software and the computer housekeeper are not turned off, the installation process may be blocked, causing the software to be unusable after installation.



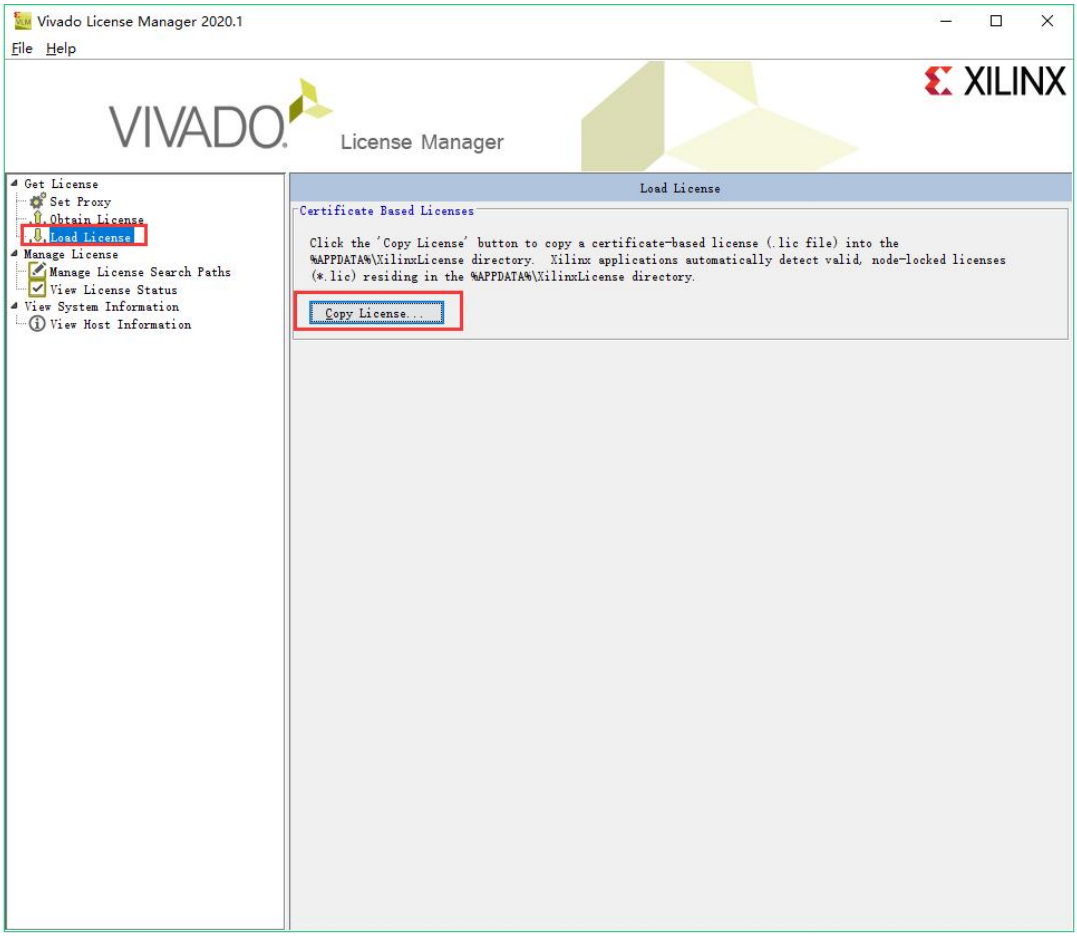
10)Prompt for successful installation



11)Install the license file, open the vivado software, select Help→Manage License



12)Click "Copy License" and select the "xilinx_ise_vivado.lic" file.



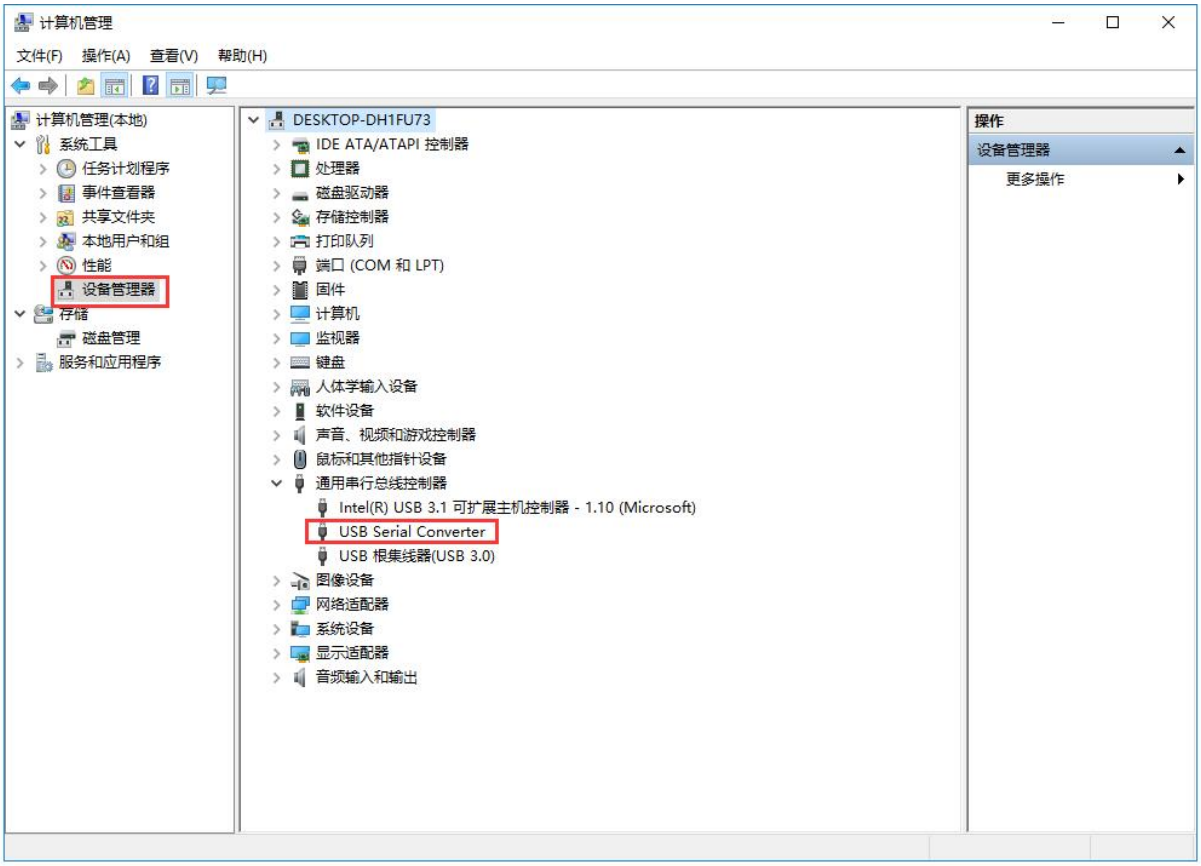
13)You can see that the installation was successful

Part 1.3: Re-download the driver to install the driver

Generally, when vivado is installed, the downloader driver will be installed.

If you need to install the downloader driver again, enter the vivado installation path "X:\XXX\Vivado\2020.1\data\xicom\cable_drivers\nt64\digilent", double-click the "install_digilent.exe" file to install, and close the vivado software before installation. If vivado cannot recognize the downloader, please try to turn off the firewall, antivirus software, and you cannot open multiple versions of vivado and ise at the same time.

After the installation is complete, connect the downloader, open the device manager, and find the USB Serial Converter in the universal serial bus controller, indicating that the installation is successful


计算机管理
文件(F) 操作(A) 查看(V) 帮助(H)
计算机管理(本地)
系统工具
任务计划程序
事件查看器
共享文件夹
本地用户和组
性能
设备管理器
存储
磁盘管理
服务和应用程序
DESKTOP-DH1FU73
IDE ATA/ATAPI 控制器
处理器
磁盘驱动器
存储控制器
打印队列
端口 (COM 和 LPT)
固件
计算机
监视器
键盘
人体学输入设备
软件设备
声音、视频和游戏控制器
鼠标和其他指针设备
通用串行总线控制器
Intel(R) USB 3.1 可扩展主机控制器 - 1.10 (Microsoft)
USB Serial Converter
USB 根集线器(USB 3.0)
图像设备
网络适配器
系统设备
显示适配器
音频输入和输出
操作
设备管理器
更多操作

Linux Chapter

The software environment introduced in this chapter is applicable to:

1. Use Vivado and Vitis under Linux
2. Build Embedded Linux
3. Develop Embedded Linux Applications

Part 2: Install virtual machine and Ubuntu system

The latter tutorial involves embedded Linux development and generally requires a Linux operating system host to compile uboot or Linux-kernel. It is the easiest way to install a virtual machine on a Windows operating system and then install the Linux operating system on the virtual machine

Part 2.1: Virtual machine software installation

The installation software version of the virtual machine we provide is “VMware-worksation-full 12.1.1”. Users can find it in the information we provide, double-click the “VMware-workstation-full-12.1.1-3770994.exe” icon to start the installation. Because the installation is relatively simple, the specific installation steps are not specifically introduced here. Users only need to click the "Next" button to install according to the default installation items. In the final interface of the installation, we need to select a license to enter a VMware12 serial number.


VMware Workstation Pro 安装
VMware Workstation Pro 安装向导已完成
单击"完成"按钮退出安装向导。
如果要立即输入许可证密钥，请按下面的"许可证"按钮。
VMWARE
WORKSTATION
PRO
12
vmware
许可证(L)
完成(F)


VMware Workstation Pro 安装
输入许可证密钥
此对话框可保存产品许可证密钥。
许可证密钥格式: XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
GA1T2-4JF1P-4819Y-GDWEZ-XYAY8
输入您的许可证密钥，或者单击"跳过"将此字段留空，以后再输入。
跳过(S)
输入(E)

Once the installation is complete, there is an icon for VMware Workstation Pro on the desktop

Part 2.2: Ubuntu installation

Part 2.2.1: install the system

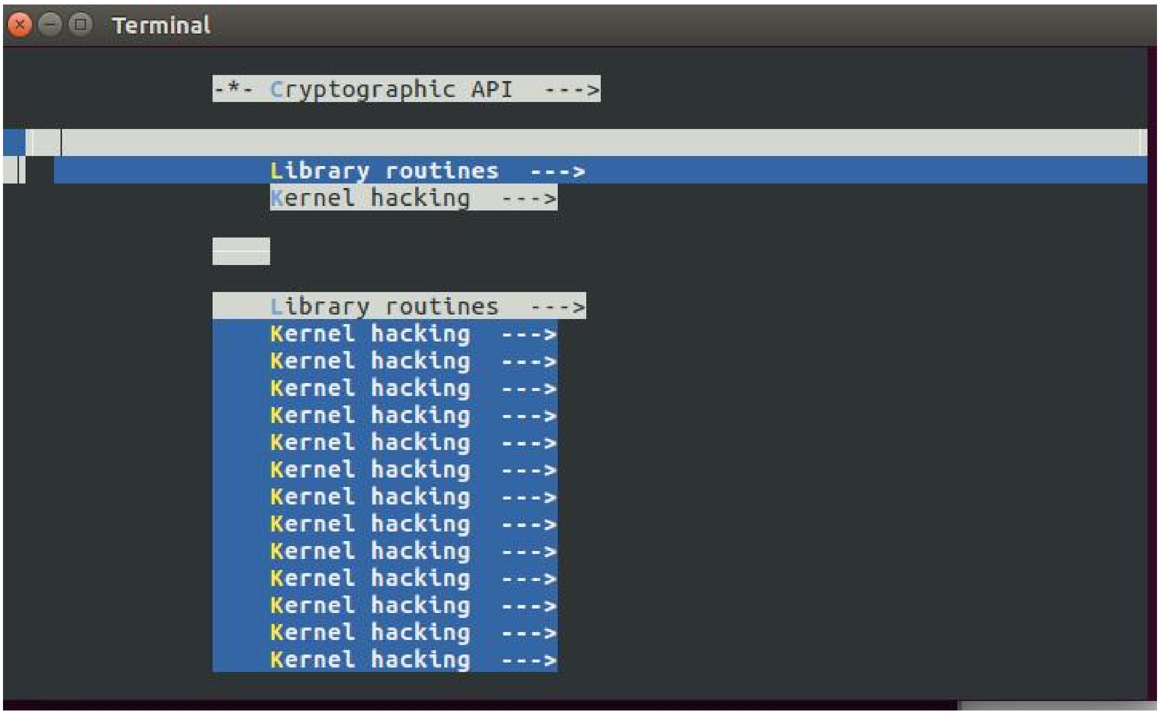
After installing the virtual machine, you must install the Linux operating system on the virtual machine. Due to the simple installation and configuration of the ubuntu Linux desktop operating system, we chose the ubuntu desktop operating system.

This tutorial uses Ubuntu 16.04.3 LTS 64-bit operating system

This tutorial uses Ubuntu 16.04.6 and Ubuntu 18.04.2 LTS 64-bit operating systems (ubuntu-16.04.6-desktop-amd64,ubuntu-18.04.2-desktop-amd64), and the last minor version number should be the same.

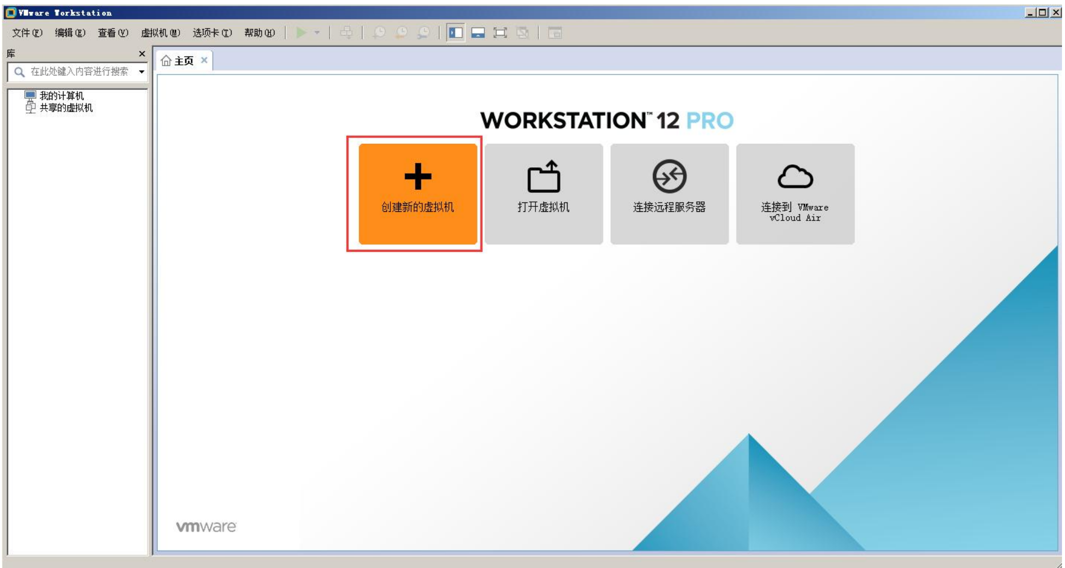
If you use other versions, there may be unpredictable errors, please keep the version consistent.

ubuntu-16.04.6-desktop-amd64 will have some minor problems that do not affect the use as shown in the figure below. When using petalinux for kernel configuration, the visual effect is relatively poor, and no solution has been found for the time being. However, 16.04.6 is still recommended, which is relatively stable.

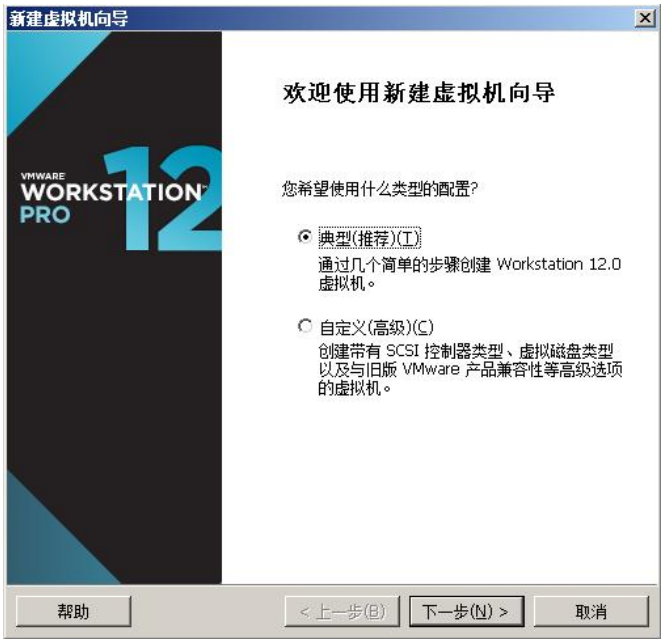


The ubuntu installation steps are as follows:

1) Double-click the icon for "VMware Workstation Pro" on your desktop, and click the Create New Virtual Machine icon on the "VMware" work interface.



2) Select "Typical" and click Next

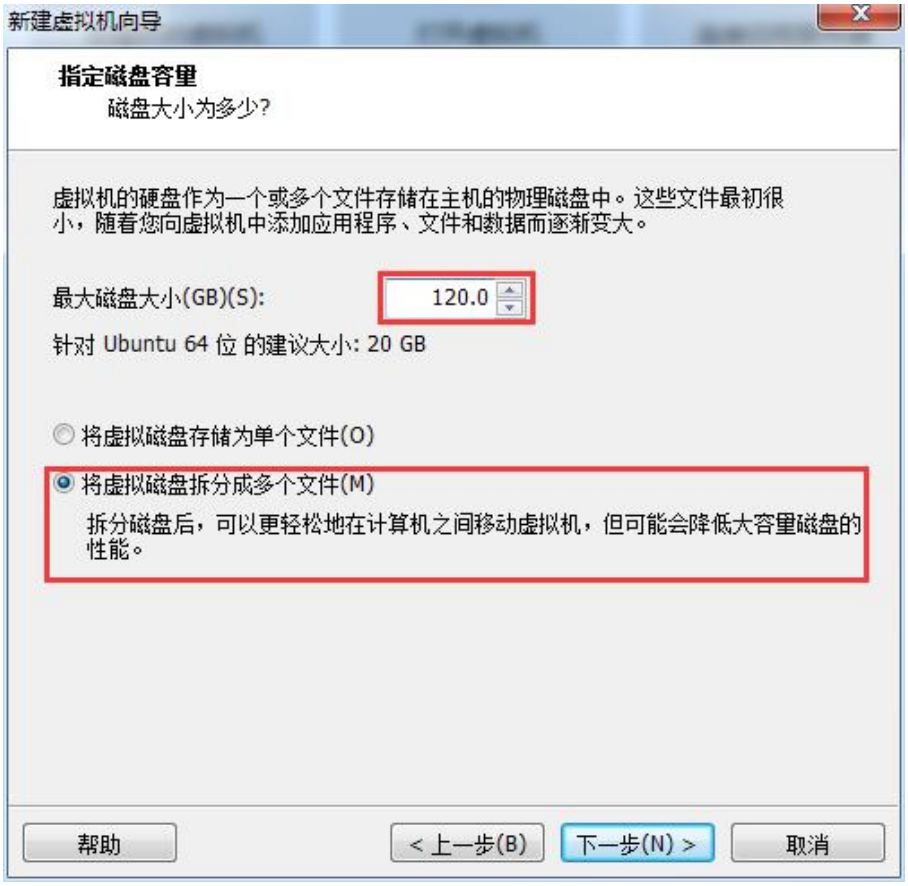


3) Select the "Installer CD Image (iso)" item and click Browse to locate the ubunt CD image file "ubuntu-16.04.3-desktop-amd64.iso"

4) Enter the full name, user and password of the virtual machine in the virtual machine wizard. The full name, username and password can be set by the user.

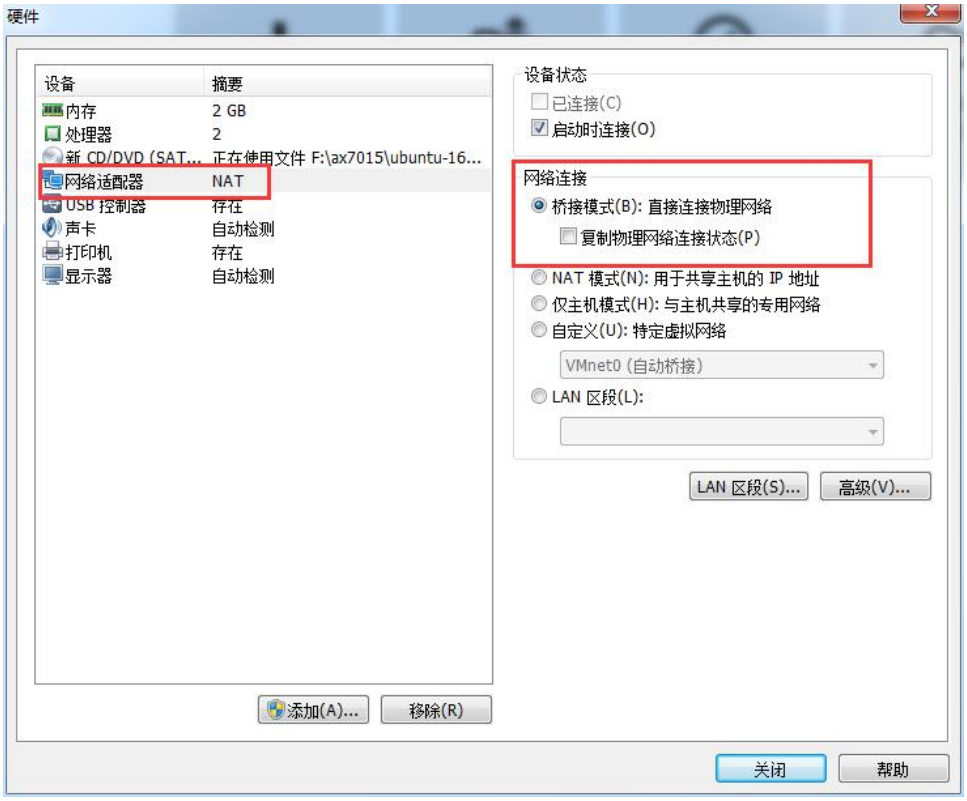
5) The virtual machine name can be modified by itself. The installation location needs to be selected to be installed on a disk with sufficient disk space.

6) To set the maximum disk size to 300G, we need to install the software in the virtual machine, where the reserved space is larger. Users can choose the appropriate space size according to their hard disk space. It is recommended to be 300G



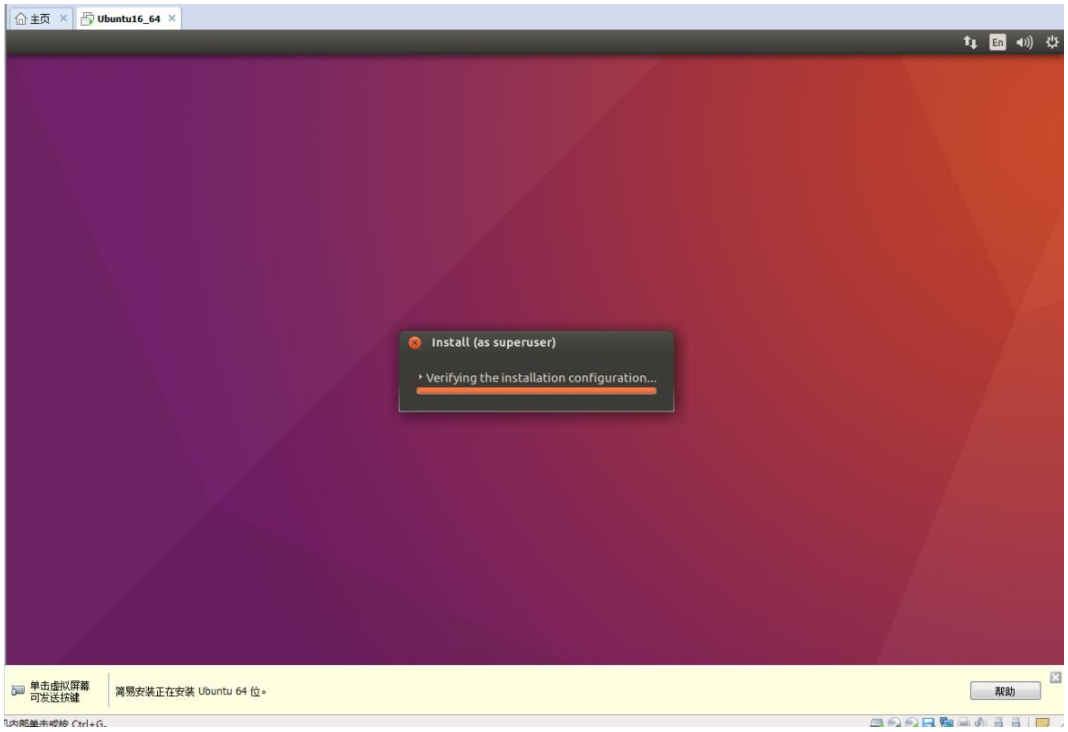
7) Choose custom hardware

8) You can modify the memory size and processor core, network adapter options, and network connection to select the bridge mode according to the modification. The bridge mode means that it is in the same network environment as the host, that is, there will be a virtual network device in the network environment, which is a virtual machine. If some network environments have network control or require account verification to access the Internet, the virtual machine cannot be connected to the Internet at this time NAT mode is a proxy mode, the virtual machine is hidden behind the host, but the external device (development board) cannot find the virtual machine, nor can it connect to the virtual machine; Therefore, you need to select the bridge mode. If the bridge mode cannot connect to the Internet, you can select the NAT mode first, and then switch to the bridge mode after using it. The memory is recommended to be set to 8G, otherwise the compilation will fail. If you want to use the Vitis acceleration function, the memory should be set to at least 16G, otherwise the compilation will not be possible.

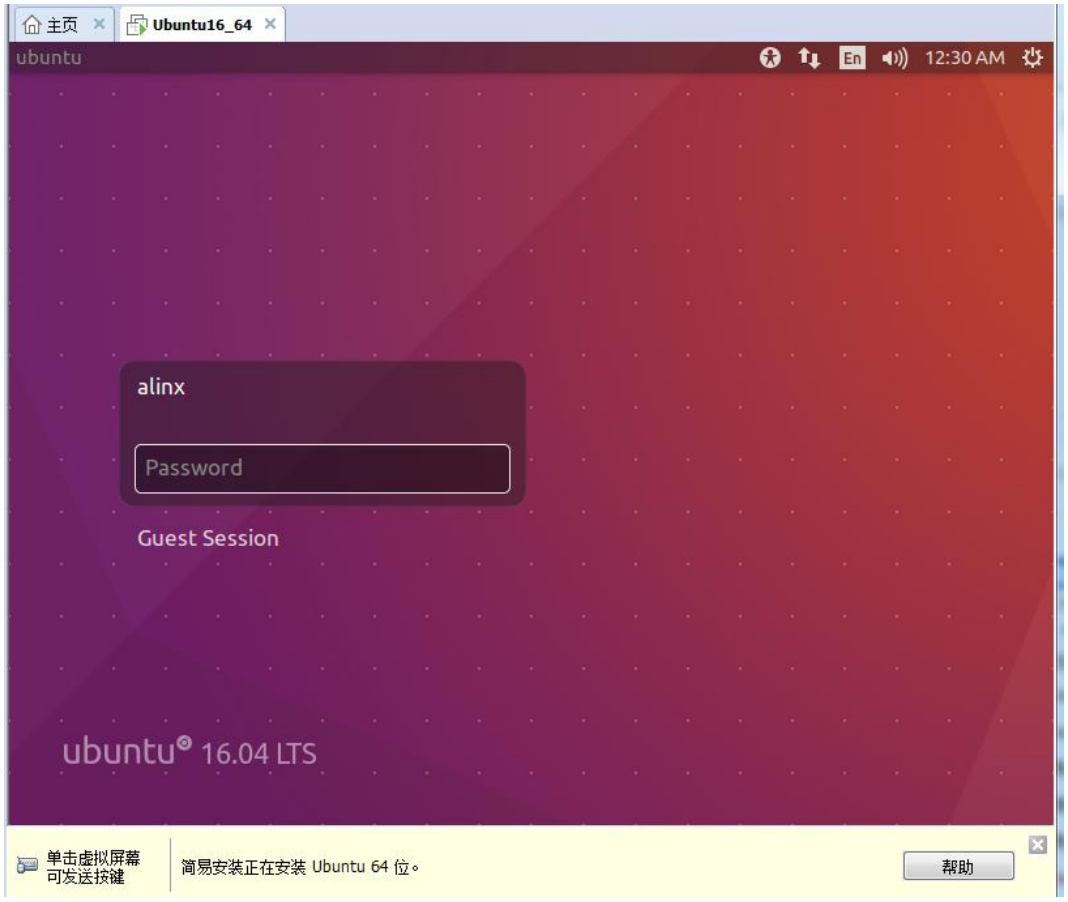


9) Click Finish to start installing Ubuntu.

10)The installation process is slow, waiting for a while

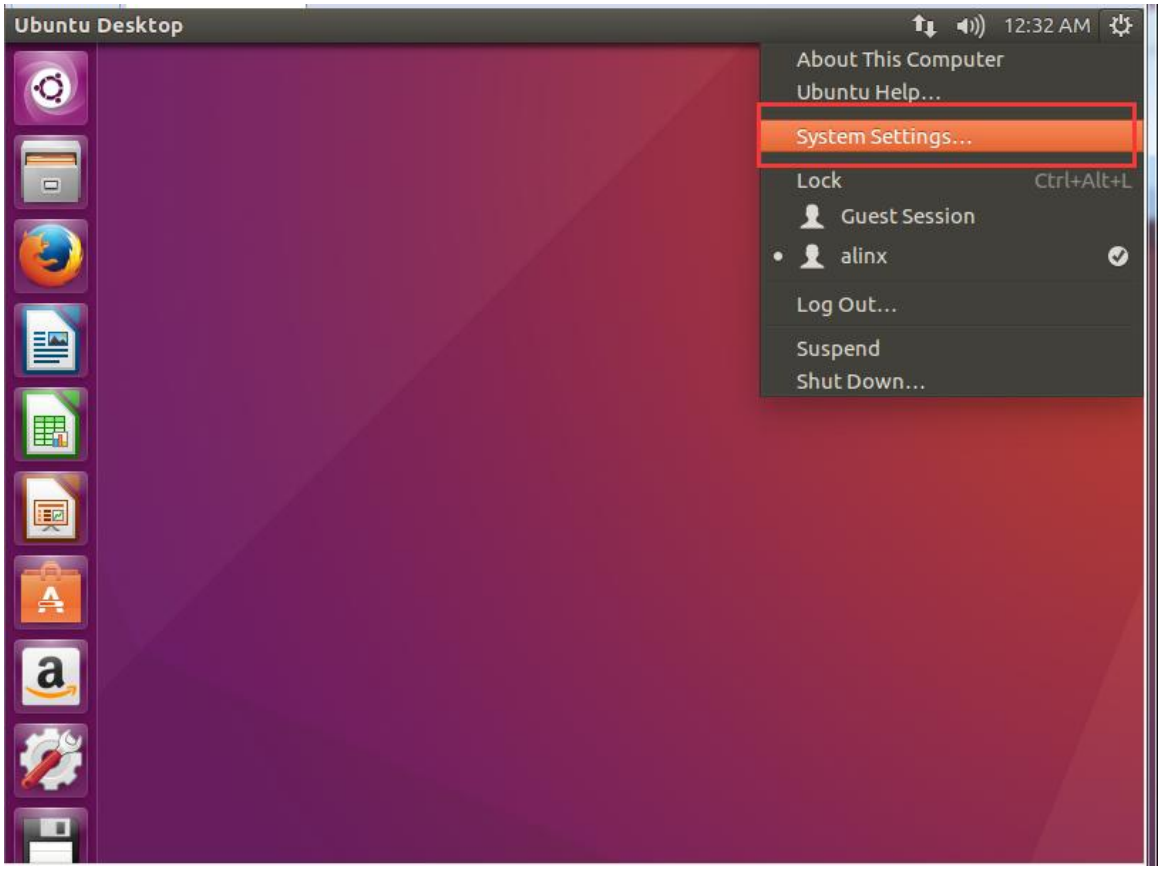


11)Enter the system after installation is complete

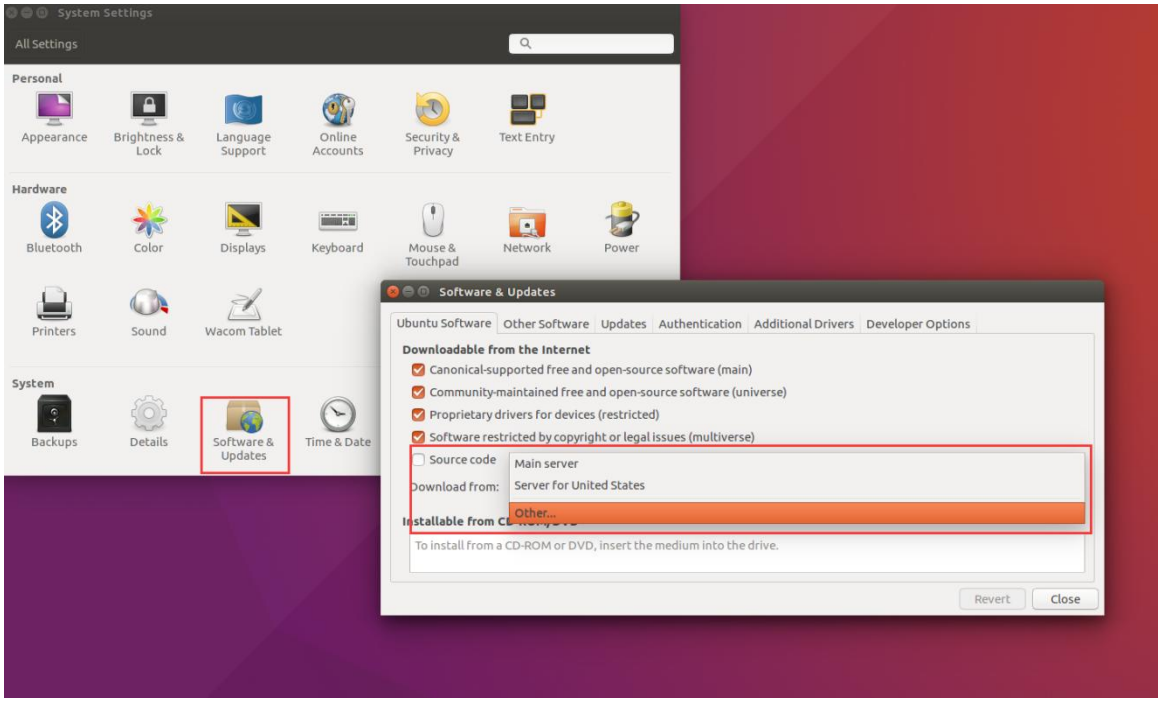


Part 2.2.2: Modify the Software Source Server

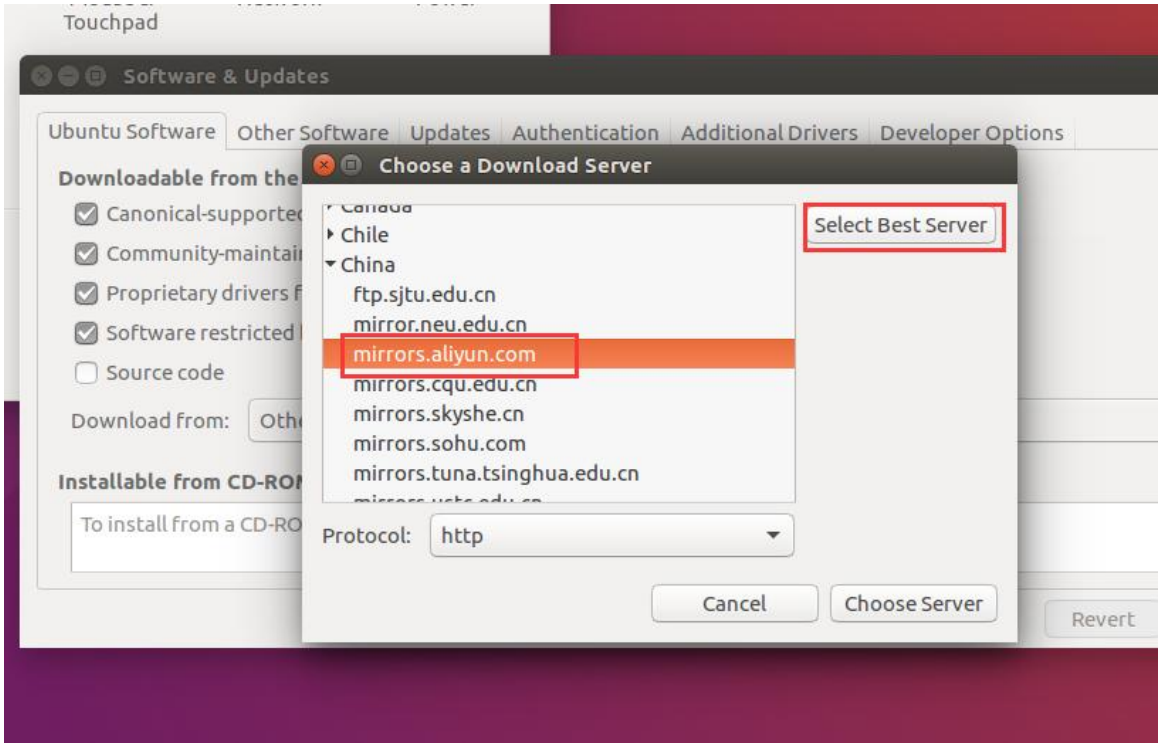
1) In order to install the software conveniently, we need to set the software source, click on the system settings.



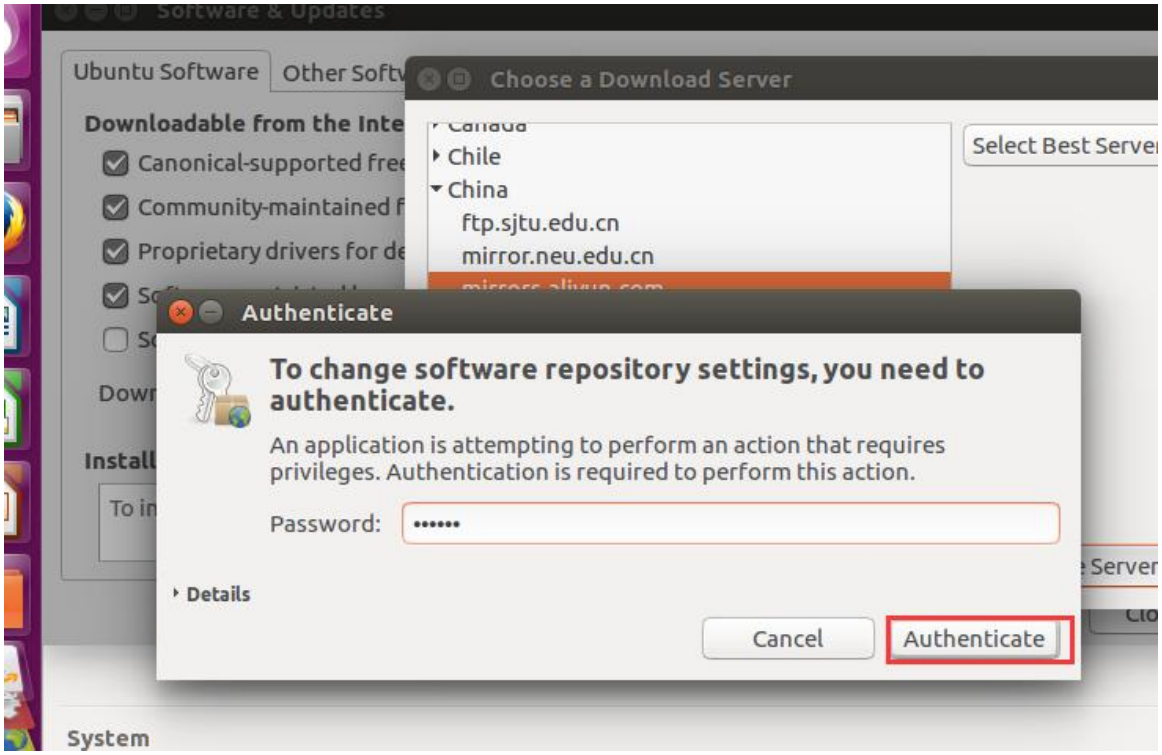
2) Select "Other..." in "Software&Updates"



3) Click on "Select Best Server" to test out the fastest server and then select "Choose Server", which is based on the fact that the virtual machine can connect to the Internet.

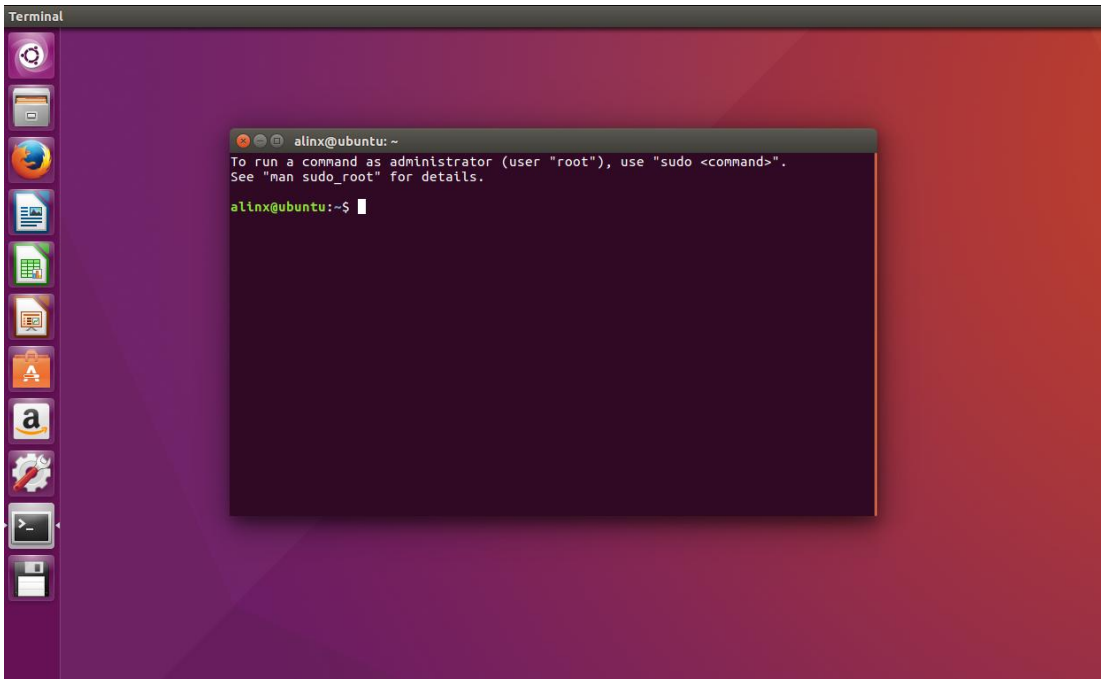


4) Enter the password to complete the software source modification.



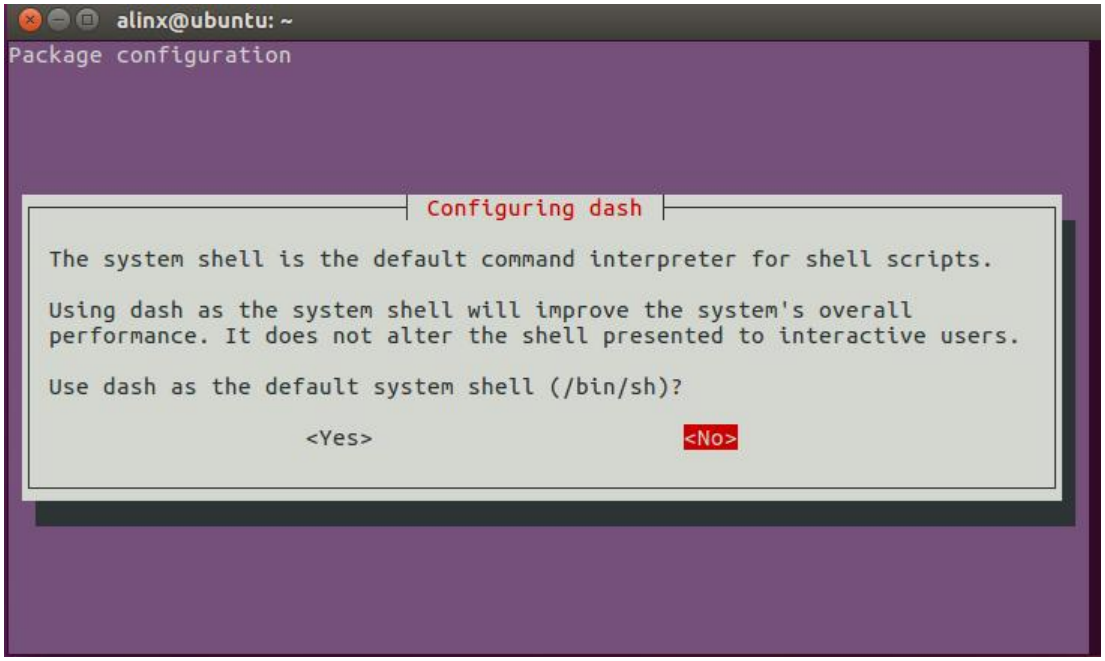
Part 2.2.3: Set bash to default sh

1) “Ctrl+Alt+T” to open the terminal



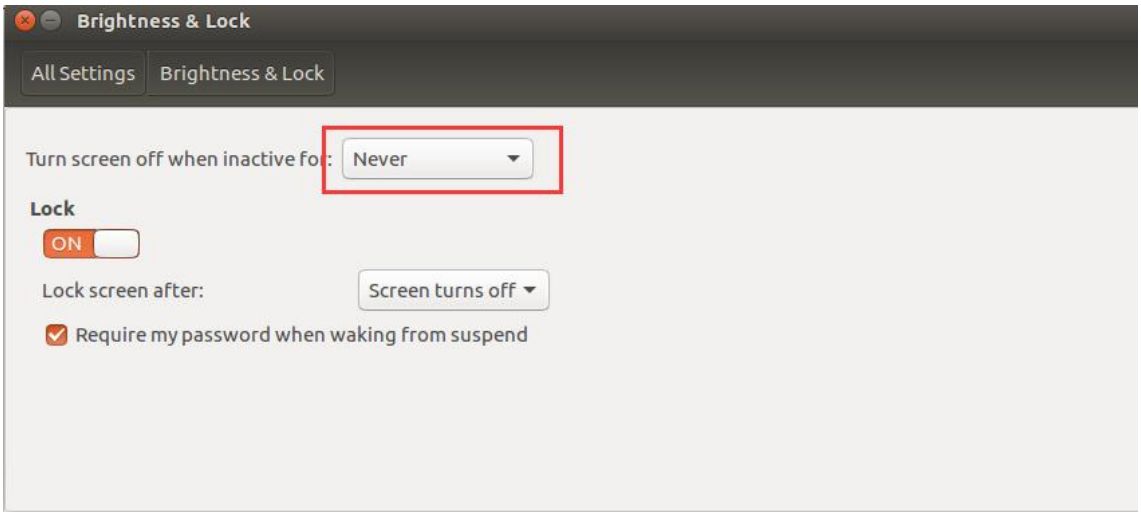
2) Enter the command, “Configuring dash” select "No", press Enter to confirm

```
sudo dpkg-reconfigure dash
```



Part 2.2.4: Set screen lock time

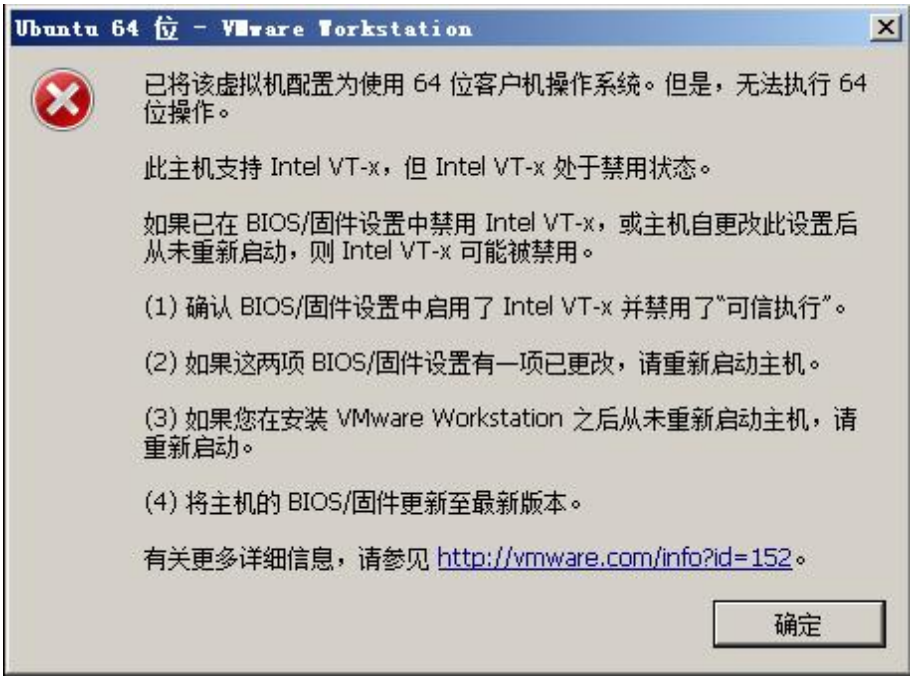
In order to copy large files to the Ubuntu system, we cancel the screen lock



Part 2.3: Q&A

Part 2.3.1: Virtual machine requires virtualization support

1) If Ubuntu is installed, the following error message box will pop up, the user needs to restart the computer and enter the BIOS for setting



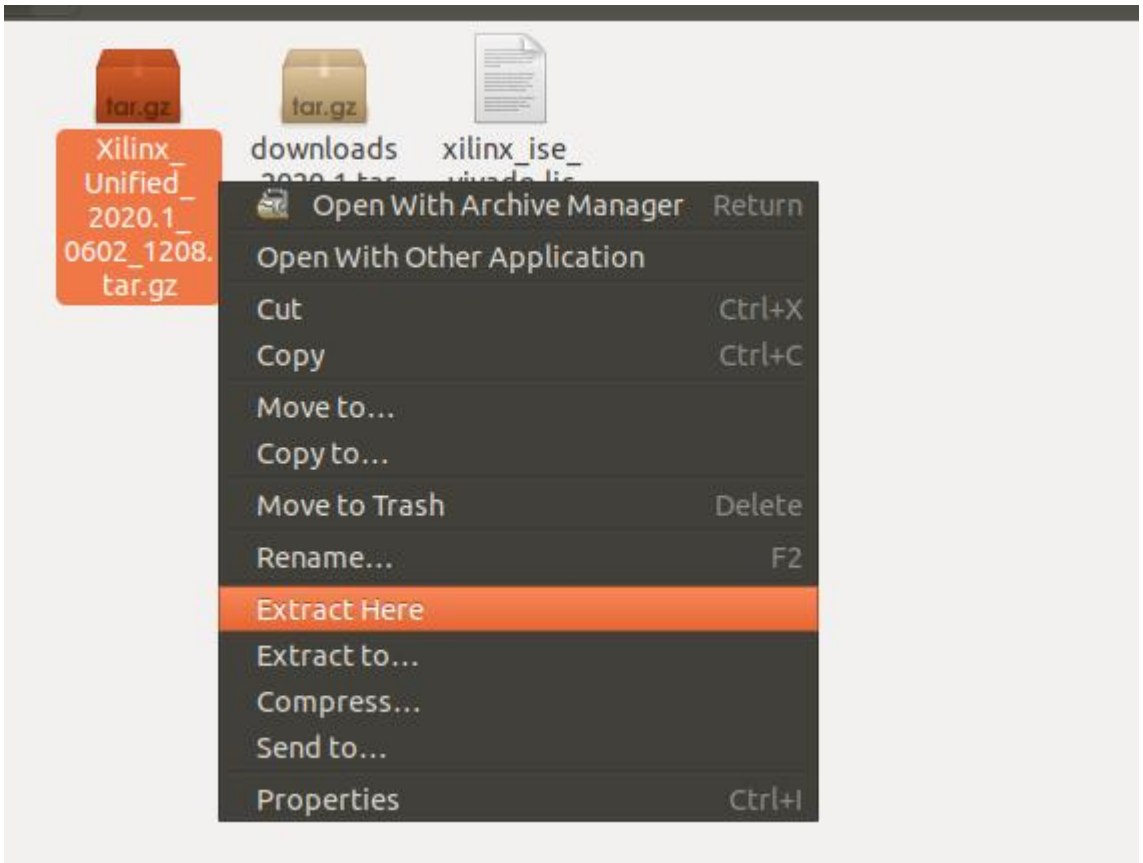
After restarting the computer, go to the BIOS, find the Intel virtualization technology and click Open. Different FPGA development boards, may have different names

Part 3: Install Linux version of Vitis software on Ubuntu

Although the Vivado software under Windows can solve most of the problems, occasionally we still use the Linux version of Vivado, especially Vitis, and we can cross-compile many applications.

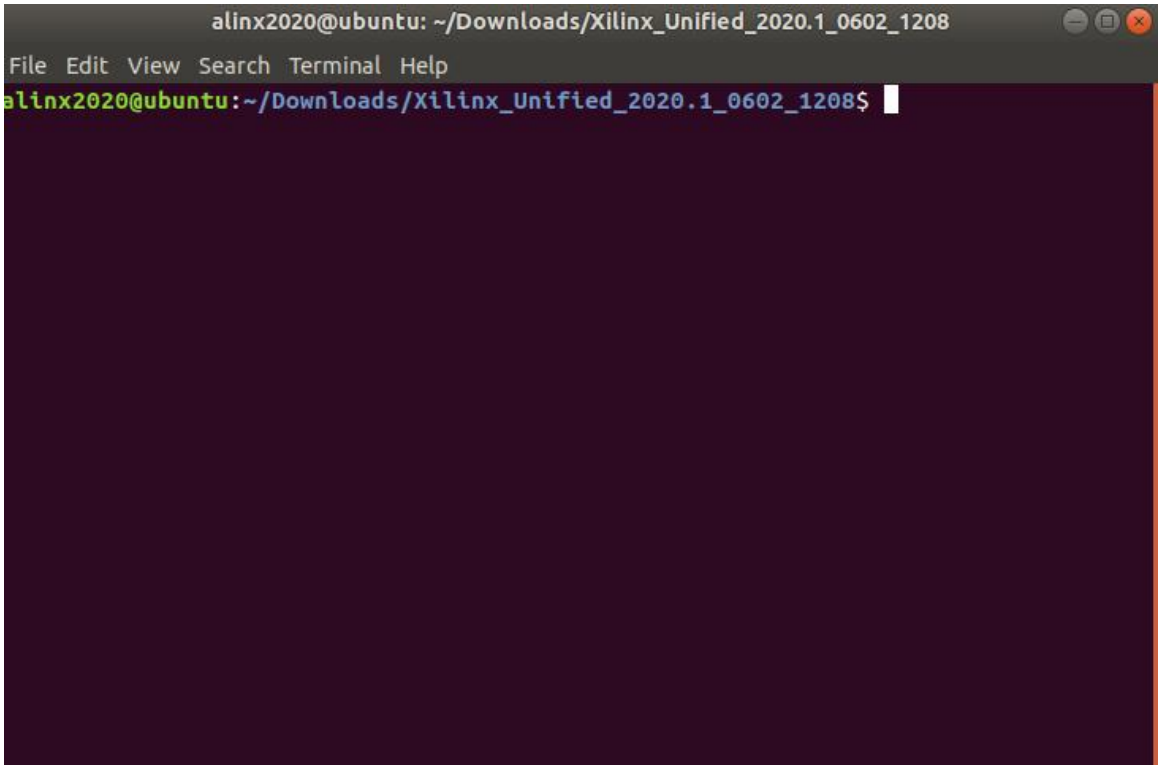
Part 3.1: Install Vitis for Linux

1) Copy the installation files to the virtual machine Ubuntu, unzip the files, and copy large files to the virtual machine. It is easy to fail. It is best to copy to the U disk or mobile hard disk first, and then open the U disk or mobile hard disk in the virtual machine to copy the required files.



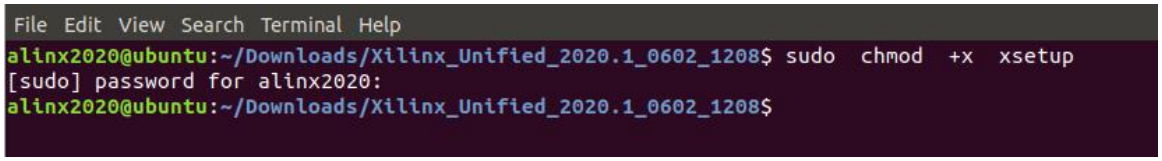
2) Use the terminal to enter the decompressed file (you can first click

on the folder with the mouse, and then right-click to open the terminal)



3) Run command

```
sudo chmod +x xsetup
```

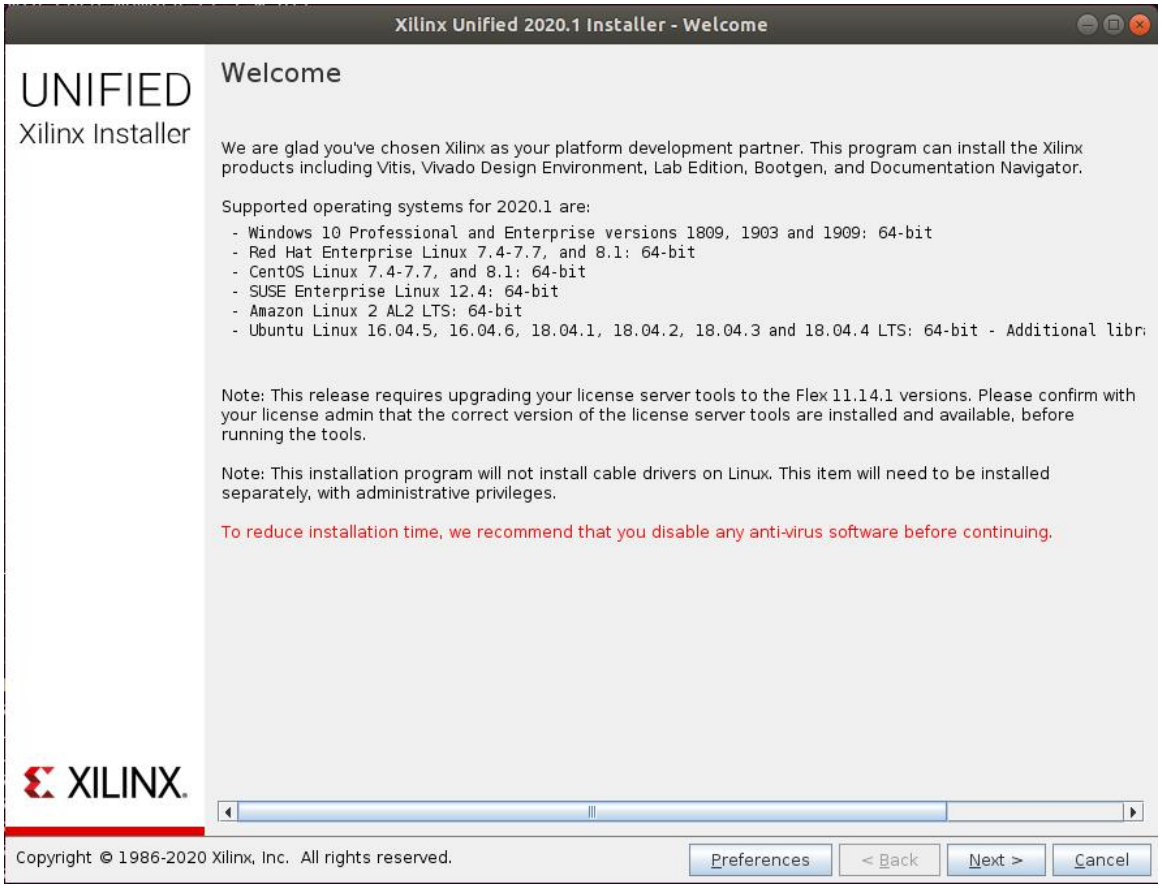


4) Run the command to start the installation, you need to pay attention, here is the sudo installation

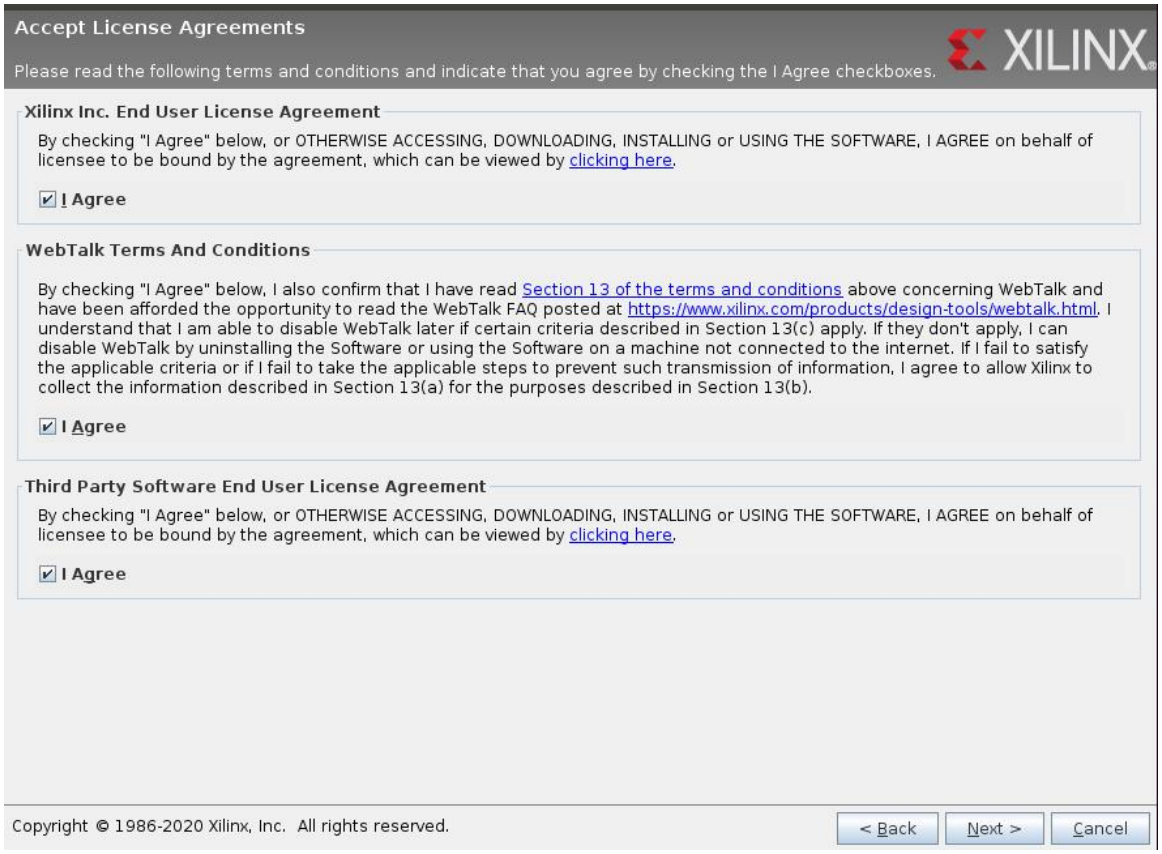
```
sudo ./xsetup
```

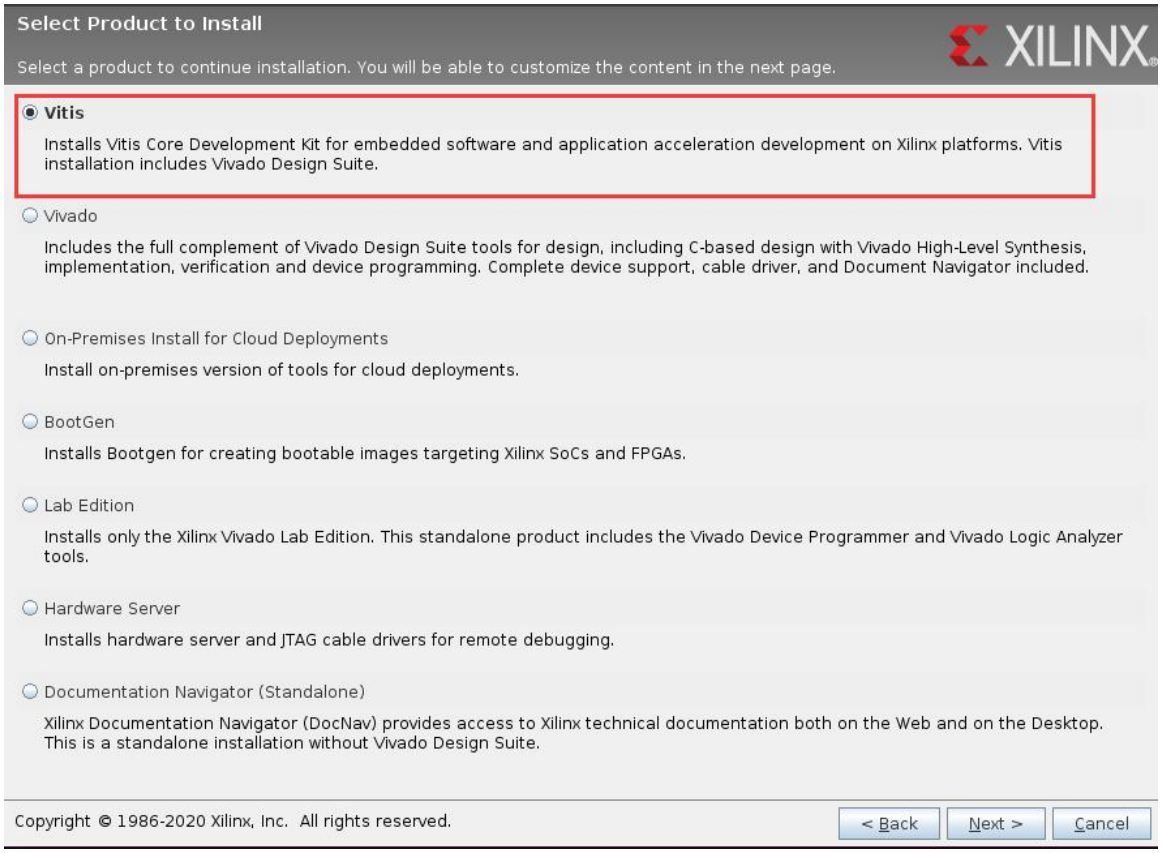
5) If these windows pop up, click "Ignore"

6) The installation process requires us to close the anti-virus software

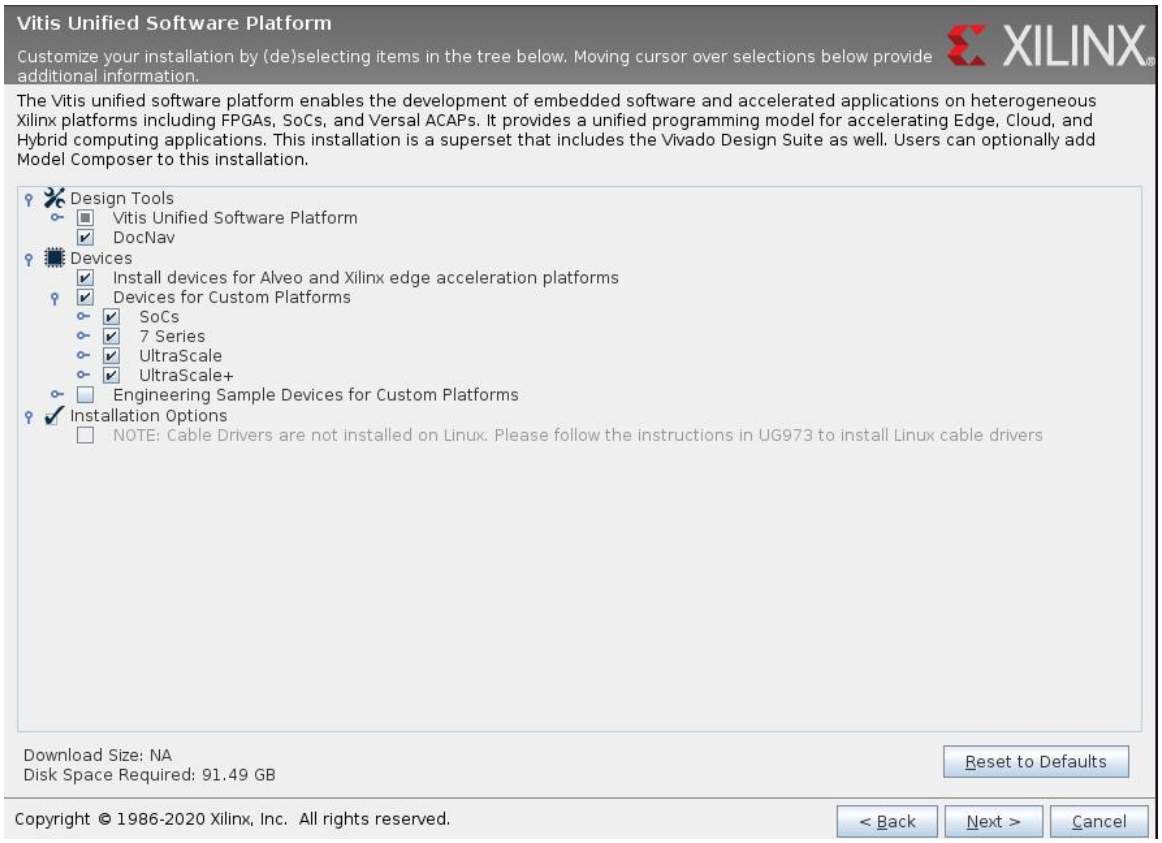


7) Agree to all terms

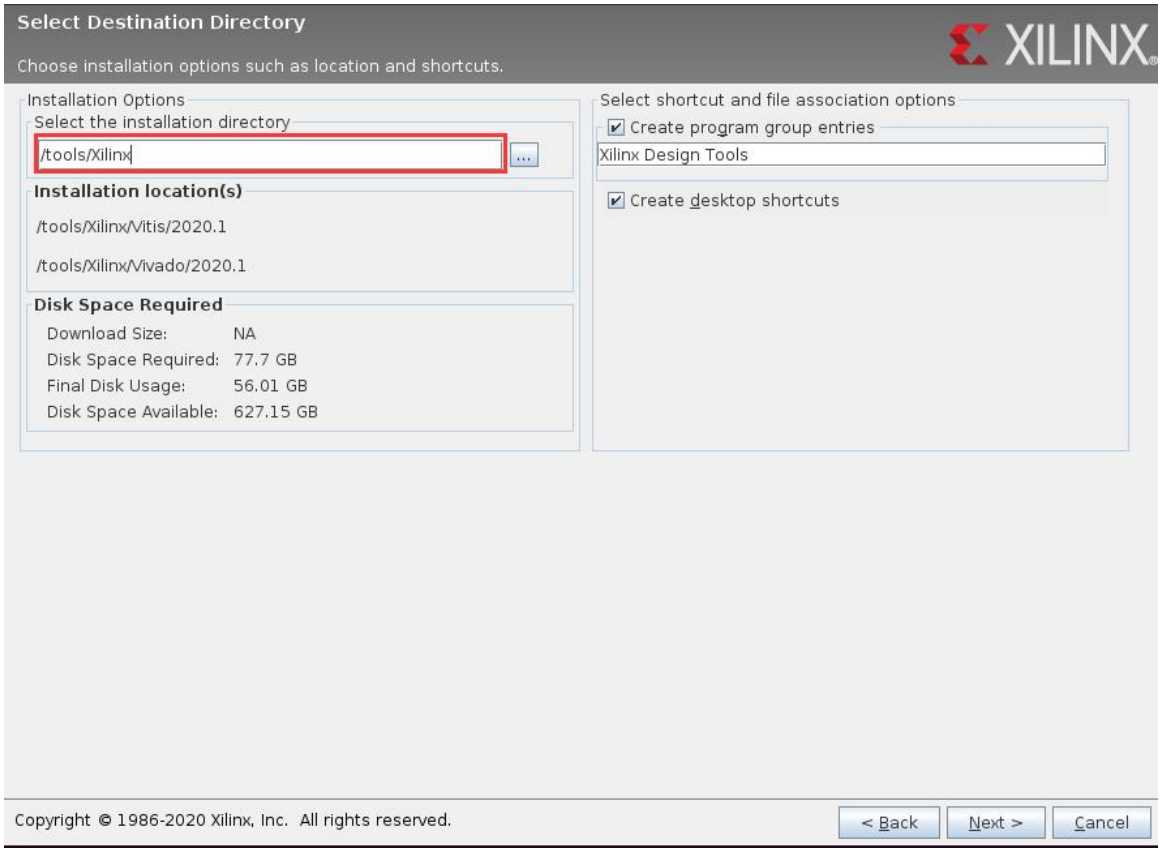




8) Keep the default configuration, select Next



9) The installation path uses the default path, click "Next" to start the installation, if prompted to create a folder, click "yes", the installation will take about 1-4 hours.



Part 3.2: Permission settings

Run command to add run permission

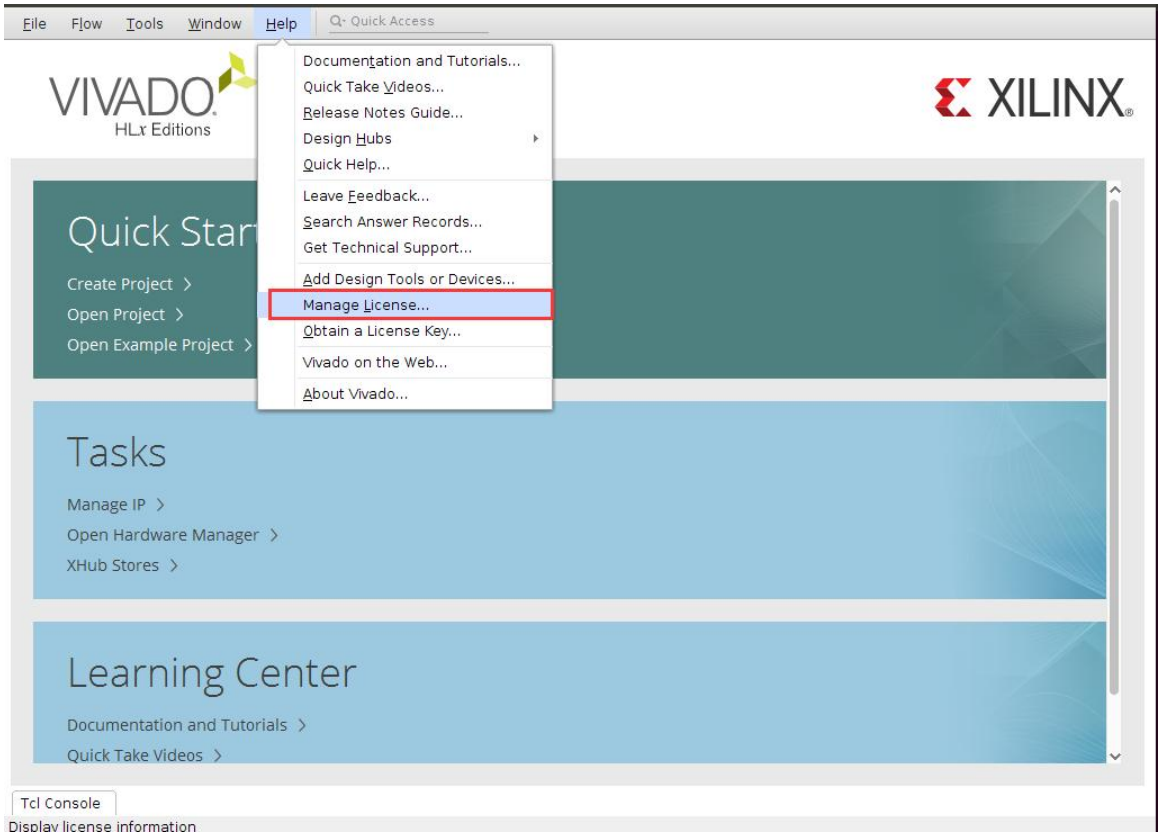
```
sudo chmod 777 -R /tools/Xilinx/
```

Part 3.3: Add license

10)Run Vivado software

```
source /tools/Xilinx/Vivado/2020.1/settings64.sh
vivado&
```

11)After the software is started, add the lice file in "Help → Manage license..."



12)Click "Load License" and then click Copy License to install the "lic" file. The lic file is common to the Windows version, and various versions are also common

Part 3.4: Install the downloader driver

Run the following command to install the downloader driver

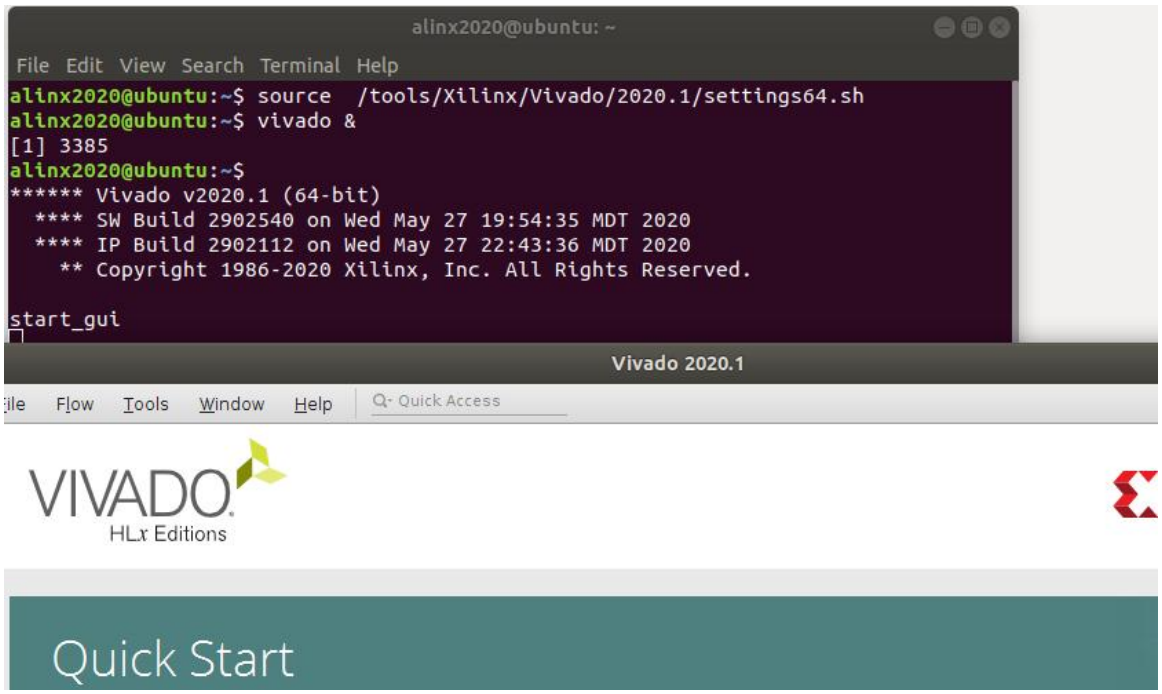
```
cd    /tools/Xilinx/Vivado/2020.1/data/xicom/cable_drivers/lin64/install_script/install_drivers
sudo ./install_drivers
```

```
alinx2020@ubuntu:~$ cd  /tools/Xilinx/Vivado/2019.2/data/xicom/cable_drivers/lin64/install_script/instal
ll_drivers/
alinx2020@ubuntu:/tools/Xilinx/Vivado/2019.2/data/xicom/cable_drivers/lin64/install_script/install_driv
ers$ sudo ./install_drivers
INFO: Installing cable drivers.
INFO: Script name = ./install_drivers
INFO: HostName = ubuntu
INFO: Current working dir = /tools/Xilinx/Vivado/2019.2/data/xicom/cable_drivers/lin64/install_script/i
nstall_drivers
INFO: Kernel version = 4.15.0-88-generic.
INFO: Arch = x86_64.
USB udev file exists and will not be updated.
--File /etc/udev/rules.d/52-xilinx-ftdi-usb.rules exists.
--File /etc/udev/rules.d/52-xilinx-ftdi-usb.rules version = 0001
--File 52-xilinx-ftdi-usb.rules exists.
--File 52-xilinx-ftdi-usb.rules version = 0001
--File 52-xilinx-ftdi-usb.rules is already updated.
--File /etc/udev/rules.d/52-xilinx-pcusb.rules exists.
--File /etc/udev/rules.d/52-xilinx-pcusb.rules version = 0002
--File 52-xilinx-pcusb.rules exists.
--File 52-xilinx-pcusb.rules version = 0002
--File 52-xilinx-pcusb.rules is already updated.

INFO: Digilent Return code = 0
INFO: Xilinx Return code = 0
INFO: Xilinx FTDI Return code = 0
INFO: Return code = 0
INFO: Driver installation successful.
CRITICAL WARNING: Cable(s) on the system must be unplugged then plugged back in order for the driver sc
ripts to update the cables.
```

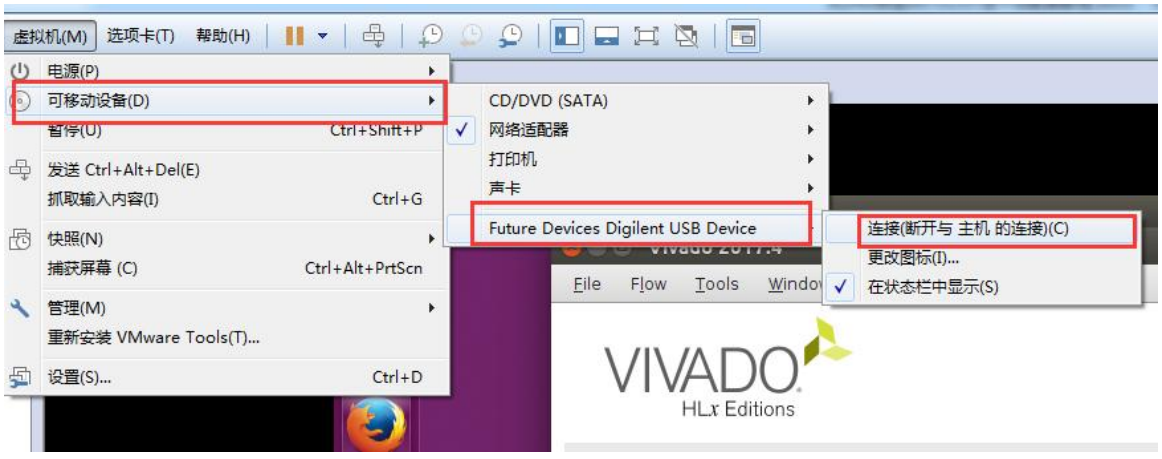
Part 3.5: Test Vivado

1) Run the following command to start Vivado

```
source    /tools/Xilinx/Vivado/2020.1/settings64.sh
vivado &
```

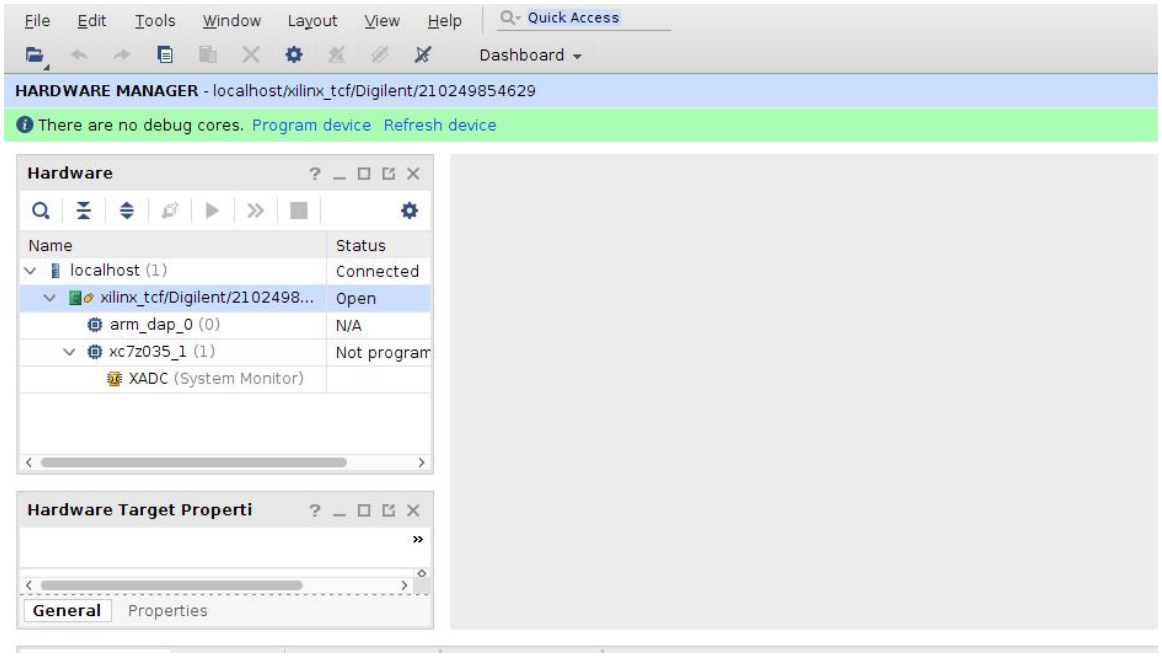


2) Connect the downloader to the virtual machine (some win10 versions will cause the virtual machine to crash, you need to use the latest virtual machine software, the software version is greater than 15.5.1)



3) Connect the development board and the downloader, use the

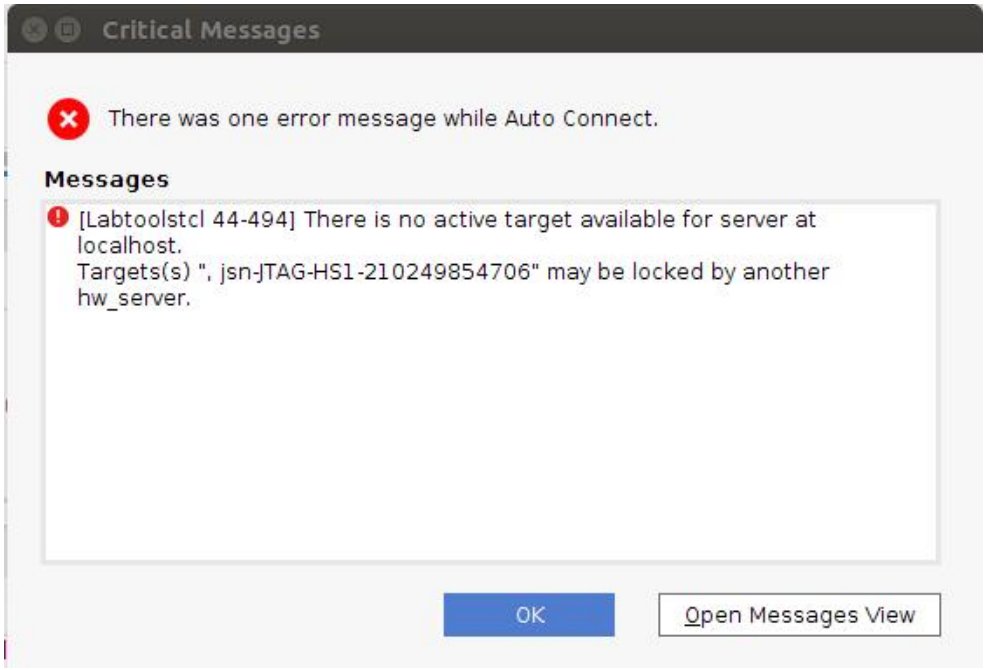
"Open Hardware Manager" test, the chip can be found under normal circumstances, indicating that the Vivado and the downloader driver are successfully installed



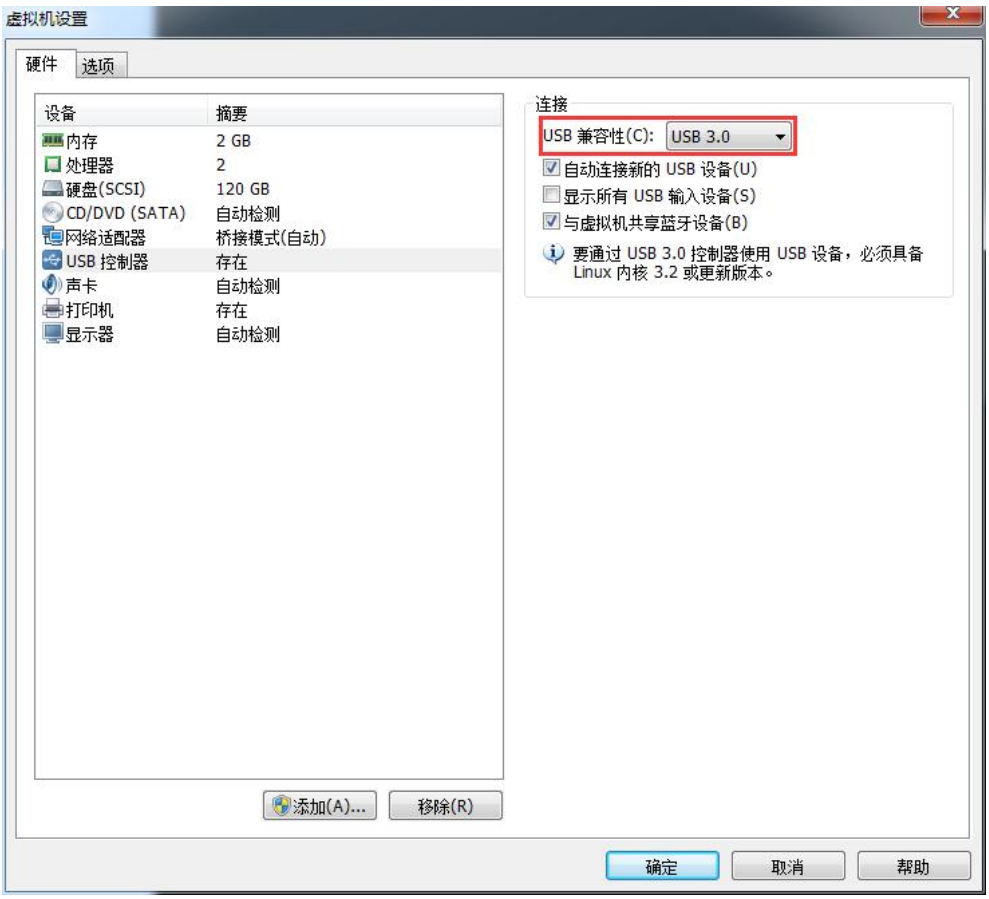
Part 3.6: Q&A

Part 3.6.1: Prompt is occupied when downloading Linux downloader

1) The downloader can be found when testing the hardware, but there is an error



2) Some motherboards need to set the USB compatibility, turn off the Ubuntu of the virtual machine, set the USB compatibility to "USB 3.0", try again, if you still cannot use the downloader, you can only use the Windows version to download



Part 3.6.2: Cross compiler suitable for ZYNQ

The cross compiler arm-linux-gnueabihf-gcc is included in the installation of the Vitis software.

```
source    /tools/Xilinx/Vivado/2020.1/settings64.sh
arm-linux-gnueabihf-gcc    -v
```

```
alinx2020@ubuntu:~$ arm-linux-gnueabihf-gcc  -v
Using built-in specs.
COLLECT_GCC=/tools/Xilinx/Vitis/2020.1/gnu/aarch32/lin/gcc-arm-linux-gnueabi/bin/../x86_64
-petalinux-linux/usr/bin/arm-xilinx-linux-gnueabi/arm-xilinx-linux-gnueabi-gcc.real
COLLECT_LTO_WRAPPER=/tools/Xilinx/Vitis/2020.1/gnu/aarch32/lin/gcc-arm-linux-gnueabi/x86_6
4-petalinux-linux/usr/bin/arm-xilinx-linux-gnueabi/../../libexec/arm-xilinx-linux-gnueabi/
gcc/arm-xilinx-linux-gnueabi/9.2.0/lto-wrapper
Target: arm-xilinx-linux-gnueabi
Configured with: ../../../../../../work-shared/gcc-9.2.0-r0/gcc-9.2.0/configure --build=x8
6_64-linux --host=x86_64-petalinux-linux --target=arm-xilinx-linux-gnueabi --prefix=/opt/p
etalinux/2020.1/sysroots/x86_64-petalinux-linux/usr --exec_prefix=/opt/petalinux/2020.1/sy
sroots/x86_64-petalinux-linux/usr --bindir=/opt/petalinux/2020.1/sysroots/x86_64-petalinux
-linux/usr/bin/arm-xilinx-linux-gnueabi --sbindir=/opt/petalinux/2020.1/sysroots/x86_64-pe
talinux-linux/usr/bin/arm-xilinx-linux-gnueabi --libexecdir=/opt/petalinux/2020.1/sysroots
/x86_64-petalinux-linux/usr/libexec/arm-xilinx-linux-gnueabi --datadir=/opt/petalinux/2020
.1/sysroots/x86_64-petalinux-linux/usr/share --sysconfdir=/opt/petalinux/2020.1/sysroots/x
86_64-petalinux-linux/etc --sharedstatedir=/opt/petalinux/2020.1/sysroots/x86_64-petalinux
-linux/com --localstatedir=/opt/petalinux/2020.1/sysroots/x86_64-petalinux-linux/var --lib
dir=/opt/petalinux/2020.1/sysroots/x86_64-petalinux-linux/usr/lib/arm-xilinx-linux-gnueabi
 --includedir=/opt/petalinux/2020.1/sysroots/x86_64-petalinux-linux/usr/include --oldinclu
dedir=/opt/petalinux/2020.1/sysroots/x86_64-petalinux-linux/usr/include --infodir=/opt/pet
```

Part 3.6.3: Cross compiler suitable for zynqMP

```
source    /tools/Xilinx/Vivado/2020.1/settings64.sh
aarch64-linux-gnu-gcc -v
```

```
File  Edit  View  Search  Terminal  Help
th-build-sysroot=/scratch/mhatle/git/internal/rel-v2020.1/build/linux-tc/tmp/work/x86_64-n
ativesdk-petalinux-linux/gcc-cross-canadian-arm/9.2.0-r0/recipe-sysroot --enable-poison-sy
stem-directories --disable-static --enable-nls --with-glibc-version=2.28 --enable-initfini
-array
Thread model: posix
gcc version 9.2.0 (GCC)
alinx2020@ubuntu:~$ aarch64-linux-gnu-gcc -v
Using built-in specs.
COLLECT_GCC=/tools/Xilinx/Vitis/2020.1/gnu/aarch64/lin/aarch64-linux/bin/../x86_64-petalin
ux-linux/usr/bin/aarch64-xilinx-linux/aarch64-xilinx-linux-gcc.real
COLLECT_LTO_WRAPPER=/tools/Xilinx/Vitis/2020.1/gnu/aarch64/lin/aarch64-linux/x86_64-petali
nux-linux/usr/bin/aarch64-xilinx-linux/../../libexec/aarch64-xilinx-linux/gcc/aarch64-xili
nx-linux/9.2.0/lto-wrapper
Target: aarch64-xilinx-linux
Configured with: ../../../../../../work-shared/gcc-9.2.0-r0/gcc-9.2.0/configure --build=x8
6_64-linux --host=x86_64-petalinux-linux --target=aarch64-xilinx-linux --prefix=/opt/petal
inux/2020.1/sysroots/x86_64-petalinux-linux/usr --exec_prefix=/opt/petalinux/2020.1/sysroo
ts/x86_64-petalinux-linux/usr --bindir=/opt/petalinux/2020.1/sysroots/x86_64-petalinux-lin
ux/usr/bin/aarch64-xilinx-linux --sbindir=/opt/petalinux/2020.1/sysroots/x86_64-petalinux-
linux/usr/bin/aarch64-xilinx-linux --libexecdir=/opt/petalinux/2020.1/sysroots/x86_64-peta
linux-linux/usr/libexec/aarch64-xilinx-linux --datadir=/opt/petalinux/2020.1/sysroots/x86_
64-petalinux-linux/usr/share --sysconfdir=/opt/petalinux/2020.1/sysroots/x86_64-petalinux-
linux/etc --sharedstatedir=/opt/petalinux/2020.1/sysroots/x86_64-petalinux-linux/com --loc
alstatedir=/opt/petalinux/2020.1/sysroots/x86_64-petalinux-linux/var --libdir=/opt/petalin
ux/2020.1/sysroots/x86_64-petalinux-linux/usr/lib/aarch64-xilinx-linux --includedir=/opt/p
etalinux/2020.1/sysroots/x86_64-petalinux-linux/usr/include --oldincludedir=/opt/petalinux
```

Part 4: Petalinux tool installation

Part 4.1: Petalinux Introduction

Petalinux is not a special Linux kernel, but a set of development environment configuration tools to reduce the workload of uboot, kernel, and root file system configuration. You can automatically configure related software from Vivado's exported hardware information.

To emphasize, petalinux has strict requirements on the system version and settings. If you use other versions of the operating system, if you encounter any problems during use, please solve it by Google

Part 4.2: Install the necessary libraries

There is a detailed installation method in Xilinx official document UG1144. If the installation fails according to this tutorial, please refer to Xilinx's ug 1144 document for installation.

https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug1144 petalinux tools reference guide.pdf

Setting Up Your Environment

Installation Steps

Installation Requirements

The PetaLinux tools installation requirements are:

- Minimum workstation requirements:
  - 8 GB RAM (recommended minimum for Xilinx® tools)
  - 2 GHz CPU clock or equivalent (minimum of eight cores)
  - 100 GB free HDD space
  - Supported OS:
    - Red Hat Enterprise Workstation/Server 7.4, 7.5, 7.6, 7.7 (64-bit)
    - CentOS Workstation/Server 7.4, 7.5, 7.6, 7.7 (64-bit)
    - Ubuntu Linux Workstation/Server 16.04.5, 16.04.6, 18.04.1, 18.04.2, 18.04.3, 18.04.4 (64-bit)
- You need to have root access to install the required packages mentioned in the following table. The PetaLinux tools need to be installed as a non-root user.
- PetaLinux requires a number of standard development tools and libraries to be installed on your Linux host workstation. Install the libraries and tools listed in the following table on the host Linux.
- PetaLinux tools require that your host system `/bin/sh` is 'bash'. If you are using Ubuntu distribution and your `/bin/sh` is 'dash', consult your system administrator to change your default system shell `/bin/sh` with the `sudo dpkg-reconfigure dash` command.

1) Update software list

```
sudo apt-get update
```

Being updated, as shown below

```
alinx@ubuntu: ~
File  Edit  View  Search  Terminal  Help
alinx@ubuntu:~$ sudo apt-get update
[sudo] password for alinx:
Get:1 http://mirrors.aliyun.com/ubuntu xenial InRelease [247 kB]
Get:2 http://mirrors.aliyun.com/ubuntu xenial-updates InRelease [109 kB]
Get:3 http://mirrors.aliyun.com/ubuntu xenial-backports InRelease [107 kB]
Get:4 http://mirrors.aliyun.com/ubuntu xenial-security InRelease [109 kB]
Get:5 http://mirrors.aliyun.com/ubuntu xenial/main amd64 Packages [1,201 kB]
Get:6 http://mirrors.aliyun.com/ubuntu xenial/main i386 Packages [1,196 kB]
Get:7 http://mirrors.aliyun.com/ubuntu xenial/main Translation-en [568 kB]
```

After the update is successful, as shown below

```
Get:67 http://mirrors.aliyun.com/ubuntu xenial-security/universe Translatio
Get:68 http://mirrors.aliyun.com/ubuntu xenial-security/universe amd64 DEP-
Get:69 http://mirrors.aliyun.com/ubuntu xenial-security/universe DEP-11 64x
Get:70 http://mirrors.aliyun.com/ubuntu xenial-security/multiverse amd64 Pa
Get:71 http://mirrors.aliyun.com/ubuntu xenial-security/multiverse i386 Pac
Get:72 http://mirrors.aliyun.com/ubuntu xenial-security/multiverse Translat
Get:73 http://mirrors.aliyun.com/ubuntu xenial-security/multiverse amd64 DE
Get:74 http://mirrors.aliyun.com/ubuntu xenial-security/multiverse DEP-11 6
Fetched 43.7 MB in 6s (6,888 kB/s)
Reading package lists... Done
alinx@ubuntu:~$
```

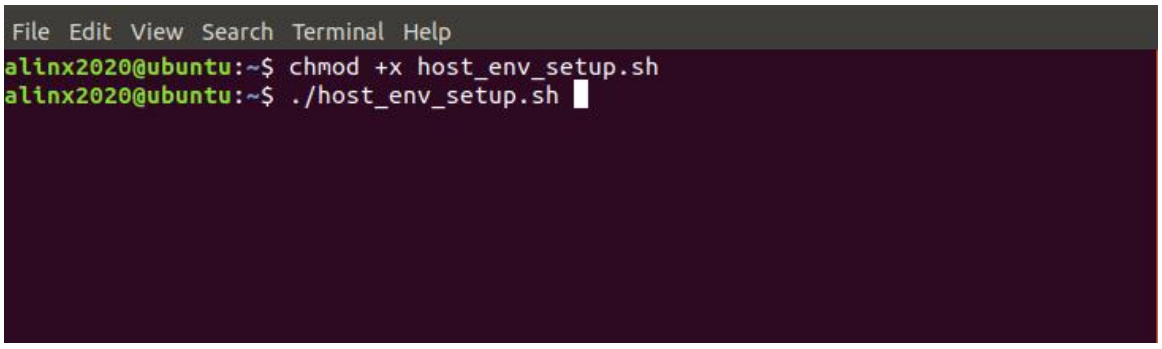
2) Run the following SHELL (host_env_setup.sh) to install the library

```
#!/bin/bash
set -x
script_dir=$(cd $(dirname ${BASH_SOURCE[0]}) && pwd)

# This script sets up a Ubuntu host to be able to create the image by
# installing all of the necessary files. It assumes a host with
# passwordless sudo

# Install a bunch of packages we need

read -d '' PACKAGES <<EOT
bc
libtool-bin
gperf
bison
flex
texi2html
texinfo
help2man
gawk
libtool
build-essential
automake
libncurses5-dev
libglib2.0-dev
libfdt-dev
```

```
device-tree-compiler
qemu-user-static
binfmt-support
multistrap
git
lib32z1
lib32ncurses5
lib32stdc++6
libssl-dev
kpartx
dosfstools
nfs-common
zerofree
u-boot-tools
rpm2cpio
curl
docker-ce
docker-ce-cli
containerd.io
libsdl1.2-dev
libpixman-1-dev
libc6-dev
chrpath
socat
zlib1g-dev
zlib1g:i386
unzip
rsync
python3-pip
python-minimal
gcc-multilib
xterm
net-tools
EOT
set -e
```

```
sudo apt-get update

if [[ $(lsb_release -rs) == "16.04" ]]; then
        echo "Install packages on Ubuntu 16.04..."
        sudo apt purge -y libgnutls-dev
elif [[ $(lsb_release -rs) == "18.04" ]]; then
        echo "Install packages on Ubuntu 18.04..."
else
        echo "Error: current OS not supported."
        exit 1
fi

# Setup docker and containerd using repository before installing them
sudo apt-get install -y \
                apt-transport-https \
                ca-certificates \
                curl \
                gnupg-agent \
                software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository \
                "deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable"
sudo apt-get update

sudo dpkg --add-architecture i386
sudo apt-get update
sudo add-apt-repository -y ppa:ubuntu-toolchain-r/ppa
sudo apt-get update
sudo apt-get install -y $PACKAGES

# Double-check docker can run with or without sudo
sudo service docker start
sudo chmod 777 /var/run/docker.sock
docker run hello-world
sudo docker run hello-world
```

```
# Create gmake symlink to keep SDK happy
cd /usr/bin
if ! which gmake
then
   sudo ln -s make gmake
fi



# update setuptools
sudo pip3 install --upgrade "setuptools>=24.2.0"



echo "Now install Vivado, SDK, and Petalinux."
echo "Re-login to    ensure the enviroment is properly set up."
```



Part 4.3: Install Petalinux

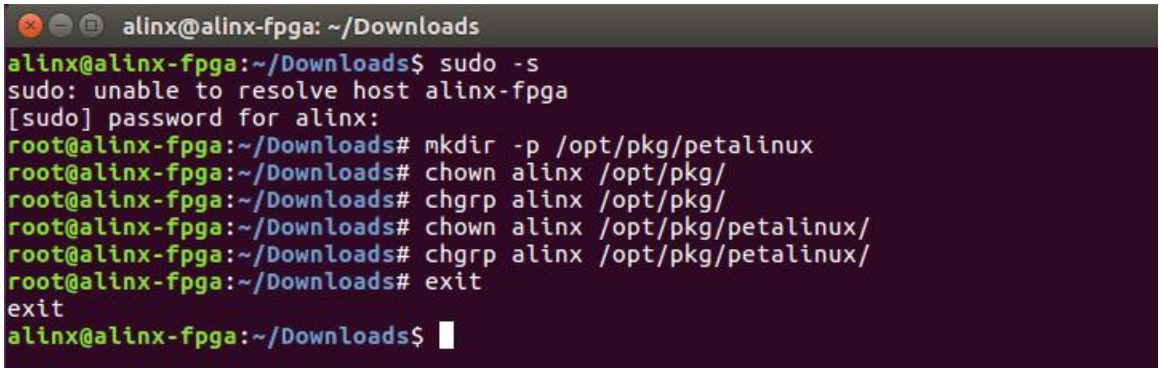
1) Copy the file petalinux-v2020.1-final-installer.run to the ubuntu system
2) Enter the directory where the file is located

```
cd   ~/Downloads
```

3) Run the following command to prepare for installation, <your_user_name> is your username

```
sudo   -s
mkdir   -p   /opt/pkg/petalinux
chgrp   <your_user_name>     /opt/pkg/petalinux
```

```
chown   <your_user_name>    /opt/pkg/petalinux
exit
```



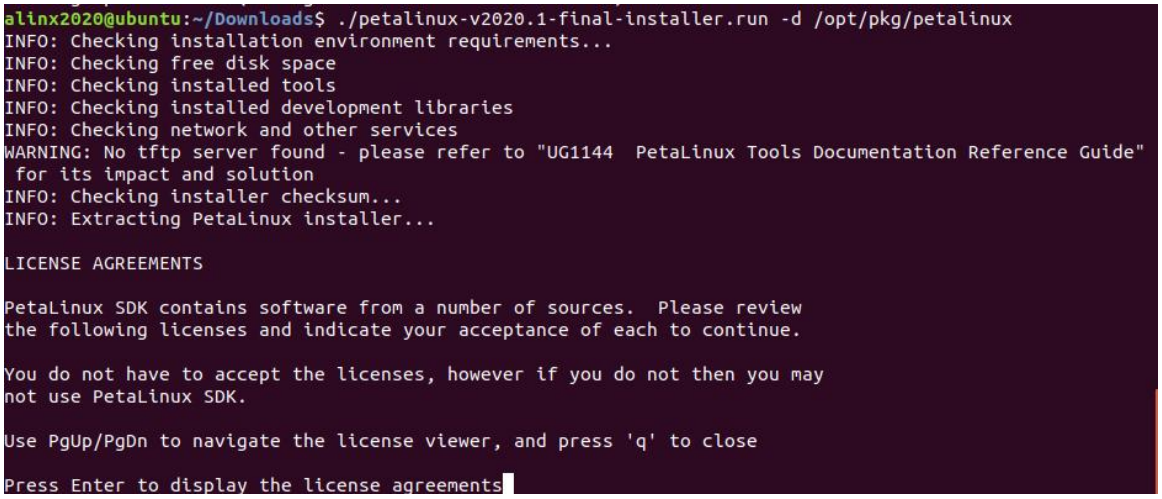
4) Add executable permissions to the file petalinux-v2020.1-final-installer.run. of course, the file petalinux-v2020.1-final-installer.run must be copied to the system first

```
chmod   +x   petalinux-v2020.1-final-installer.run
```

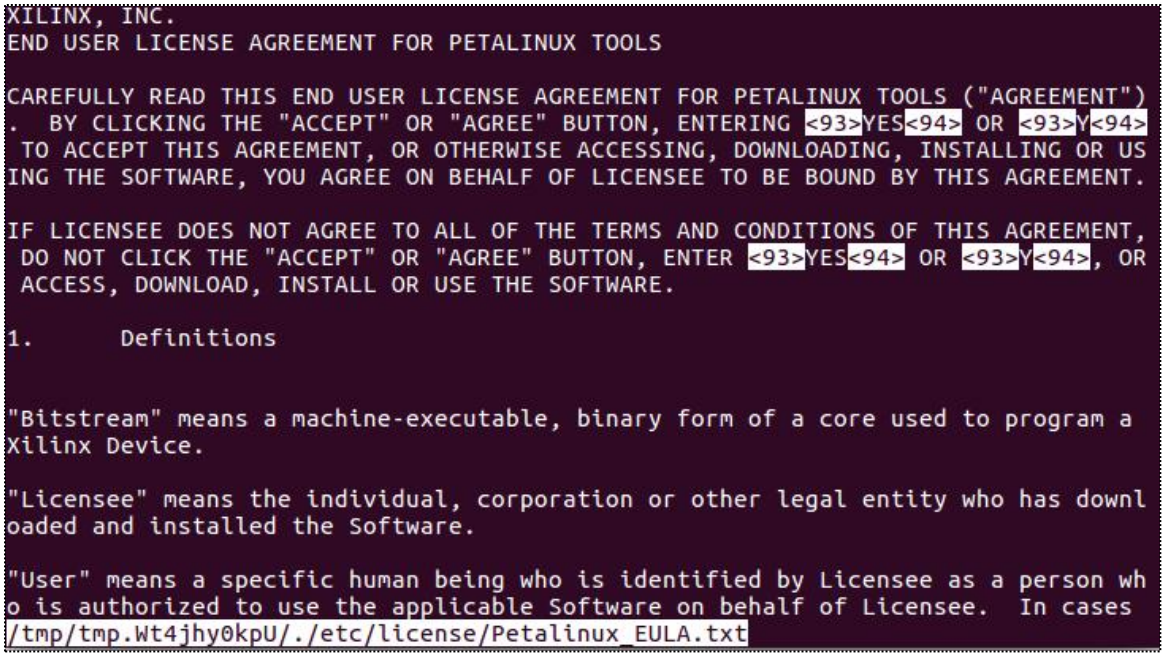
5) start installation

```
./petalinux-v2020.1-final-installer.run   -d   /opt/pkg/petalinux/
```

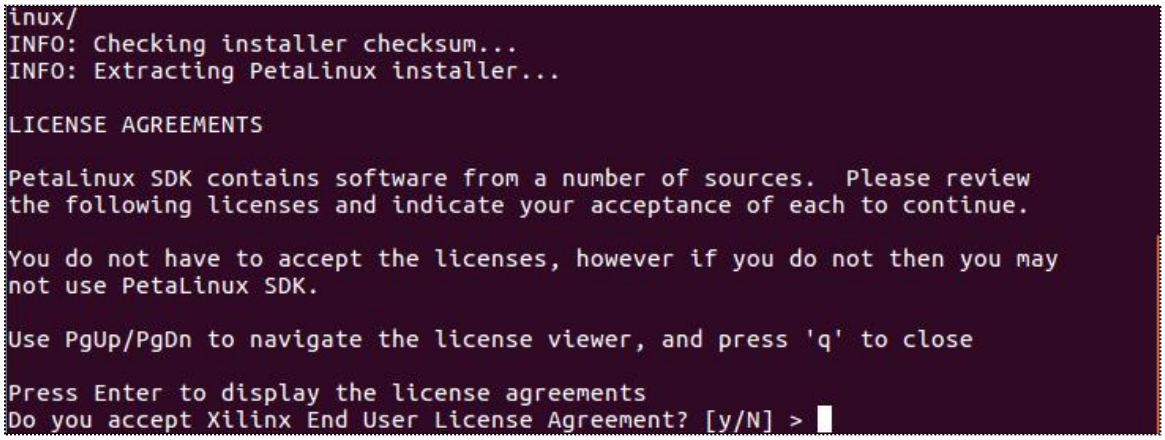
6) Press Enter to view the content of the agreement



7) Press q to exit the agreement content



8) Press y to agree to the content of the agreement



9) During the installation process, the license will pop up, press "q" to exit, and then press "y" to agree.

```
INFO: Checking installer checksum...
INFO: Extracting PetaLinux installer...

LICENSE AGREEMENTS

PetaLinux SDK contains software from a number of sources.  Please review
the following licenses and indicate your acceptance of each to continue.

You do not have to accept the licenses, however if you do not then you may
not use PetaLinux SDK.

Use PgUp/PgDn to navigate the license viewer, and press 'q' to close

Press Enter to display the license agreementsq
Do you accept Xilinx End User License Agreement? [y/N] > y
Do you accept Webtalk Terms and Conditions? [y/N] > y
Do you accept Third Party End User License Agreement? [y/N] > y
INFO: Checking installation environment requirements...
INFO: Checking free disk space
INFO: Checking installed tools
INFO: Checking installed development libraries
INFO: Checking network and other services
INFO: Installing PetaLinux...
```

Part 5: NFS service software installation

NFS (Network File System) is a technology developed by Sun Corporation and introduced in 1984 to share files between different machines and different operating systems. NFS was originally designed to be used between different systems, so its communication protocol design is independent of the host and operating system.

When using remote files, you can use the mount command to mount the file system on the remote NFS server under the local file system. Operating remote files is no different from operating local files. The shared file or directory of the NFS server is recorded in the /etc/exports file

In embedded Linux development, NFS is often used. The target system is usually used as an NFS client and the Linux host is used as an NFS server. On the target system, mount the NFS shared directory of the server to the local device through NFS, and directly run the files on the server.NFS is necessary to debug system driver modules and applications, and Linux also supports NFS root file system, which can boot the system directly from remote NFS root. This is also necessary for the cutting and integration of embedded Linux root file system.

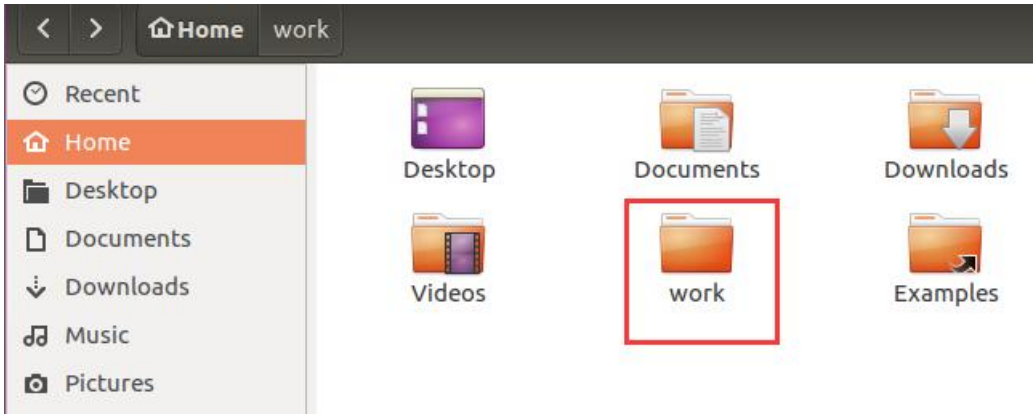
Part 5.1: Install the NFS service

1) Install the NFS server with the following command:

```
sudo apt-get install nfs-kernel-server
```

```
alinx@ubuntu: ~
alinx@ubuntu:~$ sudo apt-get install nfs-kernel-server
[sudo] password for alinx:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  keyutils libnfsidmap2 libtirpc1 nfs-common rpcbind
Suggested packages:
  open-iscsi watchdog
The following NEW packages will be installed:
  keyutils libnfsidmap2 libtirpc1 nfs-common nfs-kernel-server rpcbind
0 upgraded, 6 newly installed, 0 to remove and 325 not upgraded.
Need to get 467 kB of archives.
After this operation, 1,874 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

2) Create a new work directory as a working directory for NFS. In the future, we can put the cross-compiled program in this directory. The FPGA development board can easily share the files in this directory.



3) Use the following command to edit the "/etc/exports" file to configure the NFS service path.

```
sudo gedit /etc/exports
```

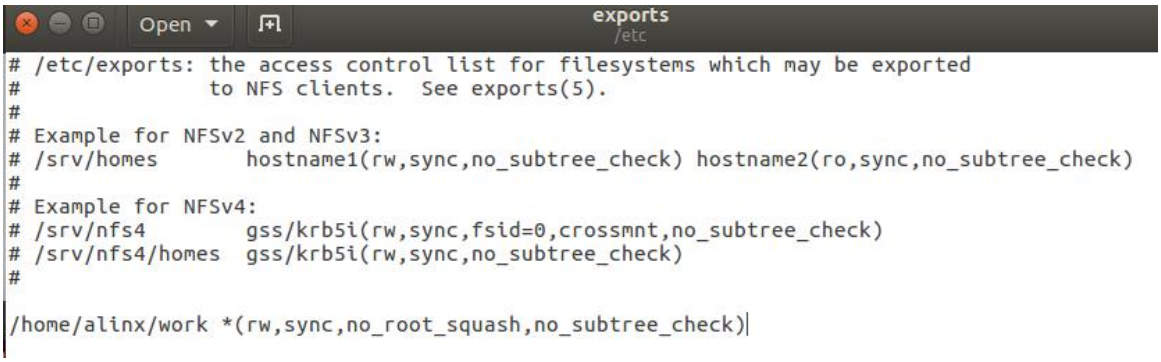
```
alinx@ubuntu: ~
alinx@ubuntu:~$ sudo  gedit /etc/exports
[sudo] password for alinx:

(gedit:60516): IBUS-WARNING **: The owner of /home/alinx/.config/ibus/bus is not
 root!
```

4) Add

/home/alinx/work *"(rw,sync,no_root_squash,no_subtree_check)

at the end to configure the "/home/alinx/work" directory to be a

working directory for NFS.



5) Execute the following command to restart the "rpcbind" service. nfs is an RPC program. Before using it, you need to map the port and set it by "rpcbind".

```
sudo    /etc/init.d/rpcbind    restart
```

6) Run the following command to restart the nfs service.

```
sudo /etc/init.d/nfs-kernel-server restart
```

Part 5.2: Testing NFS

1) Mount NFS by the following command, and mount the NFS working path in the "/mnt" directory locally. 127.0.0.1 is the local IP

```
sudo mount -t nfs 127.0.0.1:/home/alinx2020/work/ /mnt
```

2) Enter "/mnt", create a new test directory test, you can see the test folder in the "/home/alinx/work" directory.

```
cd    /mnt
mkdir test
```

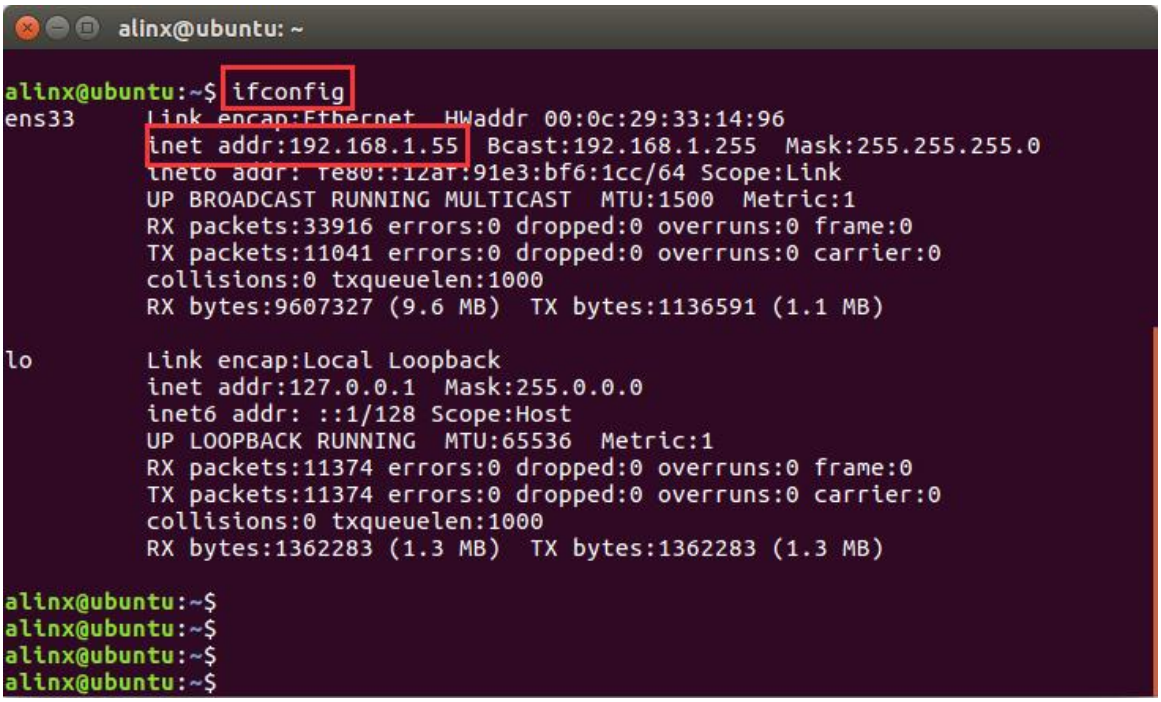
Part 5.3: Q&A

Part 5.3.1: NFS cannot be mounted

First confirm whether the virtual machine and FPGA development

board are on the same network segment?

Use the “ifconfig” command to view the virtual machine IP address. The example in the following figure is **192.168.1.55**, which belongs to the **192.168.1.x** network segment. Because there is a “DHCP” server in the development environment, the virtual machine's IP address is automatically assigned because the network environment is different.



Use the “ifconfig” command in the serial terminal to view the IP address of the development board. The example in the following figure is 192.168.1.46, which belongs to the 192.168.1.x network segment. If there is no IP, or different network segment from the FPGA development board IP, please contact the network administrator

```
root@zynq:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0a:35:00:1e:53
          inet addr:192.168.1.46  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20a:35ff:fe00:1e53/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:37 errors:0 dropped:0 overruns:0 frame:0
          TX packets:37 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4597 (4.4 KiB)  TX bytes:3782 (3.6 KiB)
          Interrupt:29 Base address:0xb000

eth1      Link encap:Ethernet  HWaddr 00:0a:35:00:03:22
          UP BROADCAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:1104 (1.0 KiB)  TX bytes:1104 (1.0 KiB)

root@zynq:~#
```

"Ping" the virtual machine in the serial terminal. The routine **ping 192.168.1.55**. This is because the virtual machine IP is 192.168.1.55. It can be pinged to mount **NFS** normally.

```
root@zynq:~# ping 192.168.1.55
PING 192.168.1.55 (192.168.1.55) 56(84) bytes of data.
64 bytes from 192.168.1.55: icmp_seq=1 ttl=64 time=0.862 ms
64 bytes from 192.168.1.55: icmp_seq=2 ttl=64 time=0.489 ms
64 bytes from 192.168.1.55: icmp_seq=3 ttl=64 time=0.779 ms
64 bytes from 192.168.1.55: icmp_seq=4 ttl=64 time=0.504 ms
64 bytes from 192.168.1.55: icmp_seq=5 ttl=64 time=0.574 ms
^C
--- 192.168.1.55 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4131ms
rtt min/avg/max/mdev = 0.489/0.641/0.862/0.153 ms
root@zynq:~#
```

Part 6: QT Creator development environment

In the previous chapter, we used gedit to edit the program and call the gcc compiler. If our application has multiple files and some library files need to be called, how to compile it? At this time, Makefile is needed. But Makefile needs to learn its grammar and various rules, and then it needs to edit and maintain the Makefile according to the changes of the program files. Thinking about it is a headache.

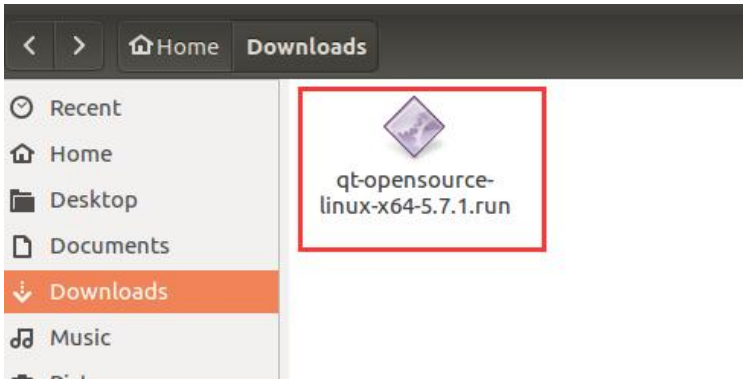
Is there a way to skip the Makefile? Unfortunately, no! But there are tools to help you automatically generate Makefile files, does it feel like another village?

Here, the tool we want to introduce is Qt Creator: an integrated development environment (IDE), its integrated tool qmake, can help us automatically generate Makefile. And integrated document compiler, including code jump, etc., can make us as convenient and fast as programming under windows

Part 6.1: QT Creator installation

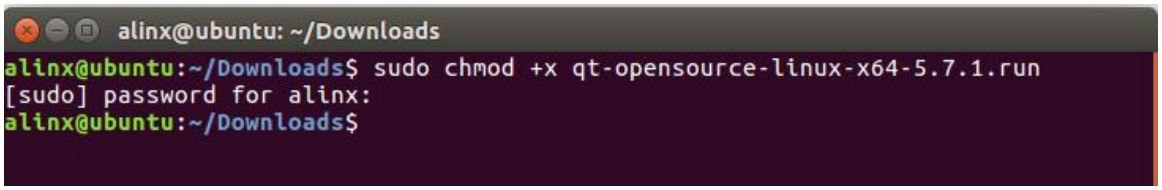
We install QT Creator in Ubuntu, the installation file name is "qt-opensource-linux-x64-5.7.1.run"

1) Copy the installation files to the Ubuntu host\



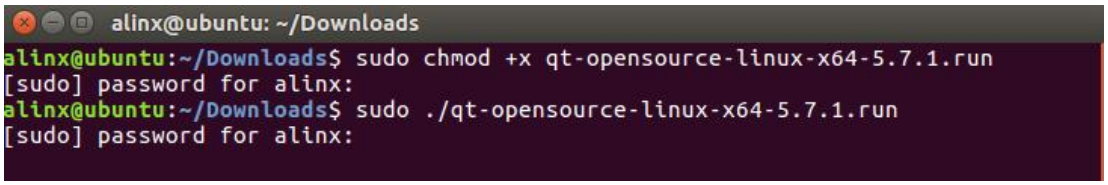
2) Add run permission

```
sudo chmod +x qt-opensource-linux-x64-5.7.1.run
```

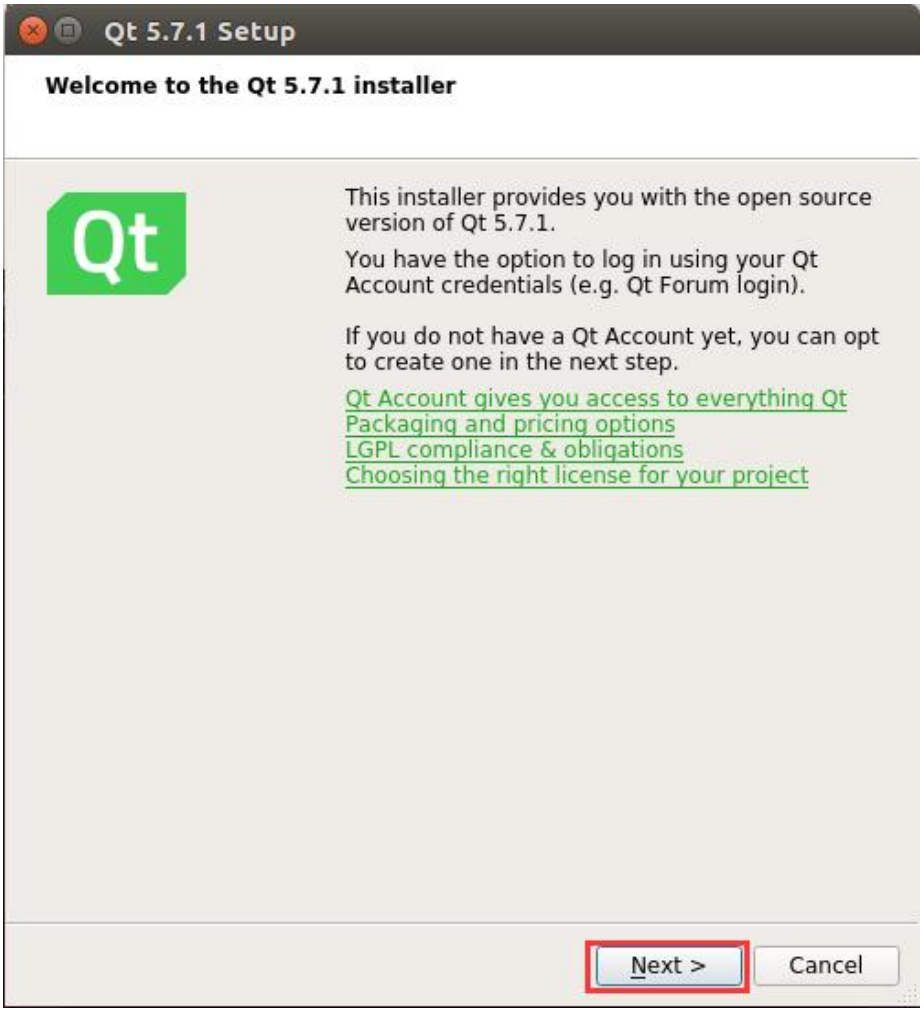


3) Run the installation software

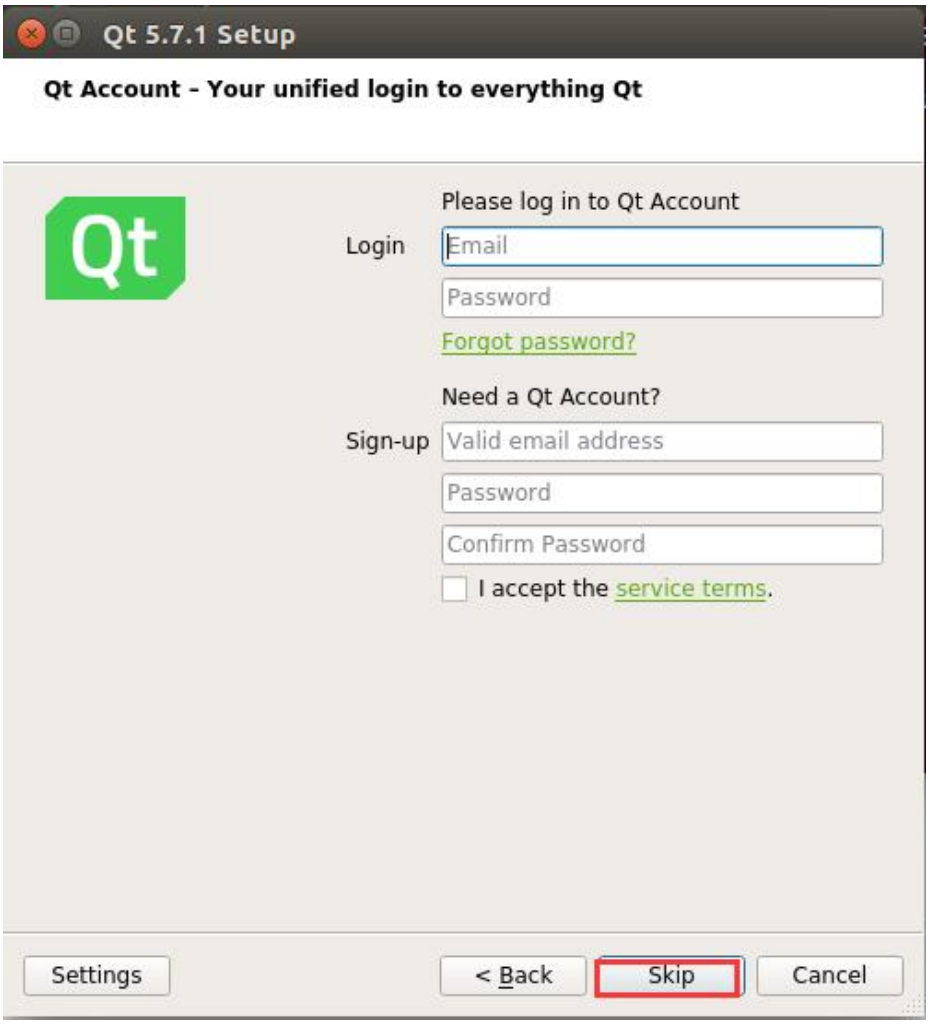
```
sudo ./qt-opensource-linux-x64-5.7.1.run
```



4) Click Next to start the installation



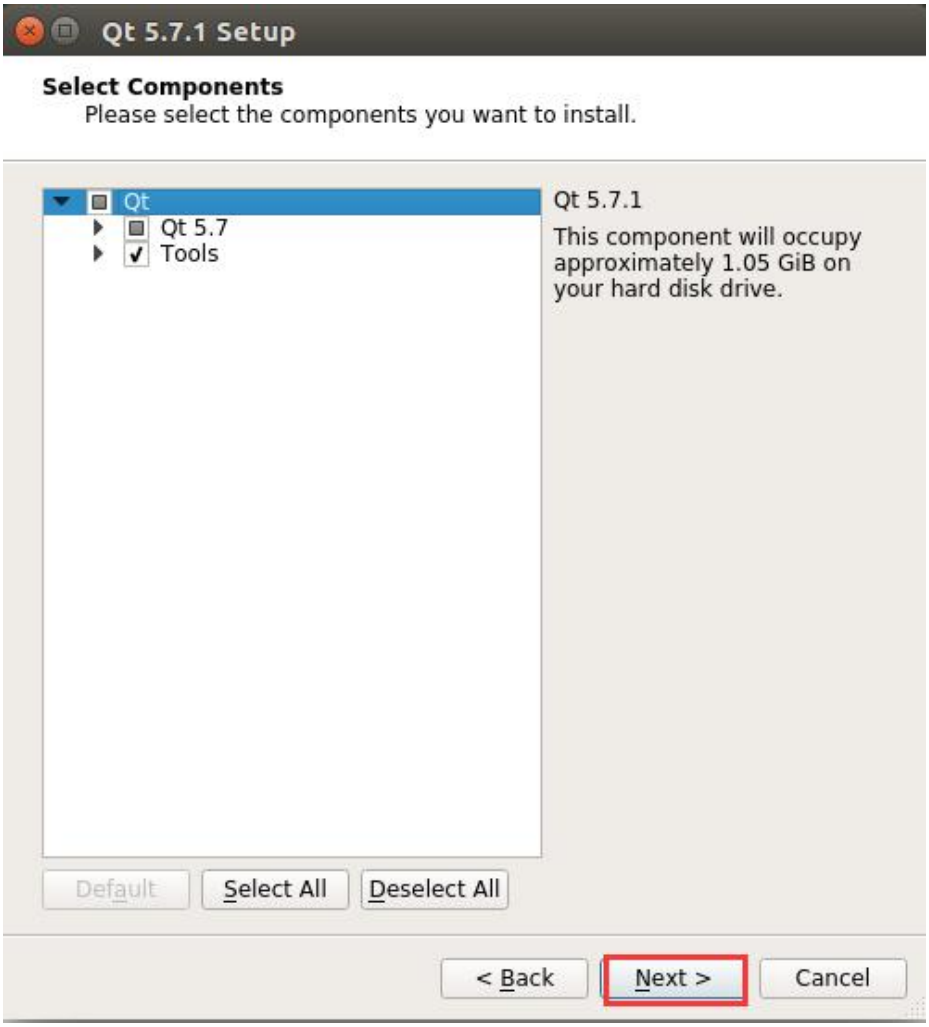
5) Click "Skip" to skip filling in account information



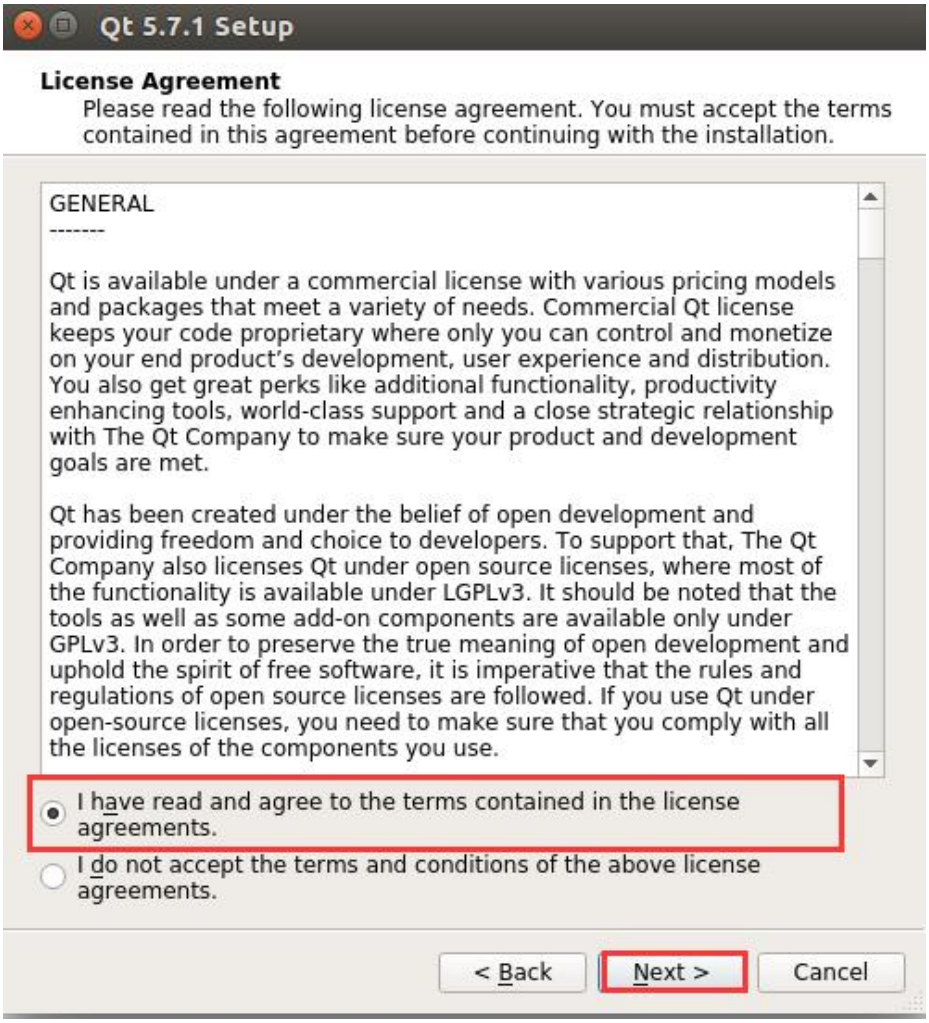
6) Click "Next" to continue the installation

7) Keep the installation path as default, click "Next" to continue the installation

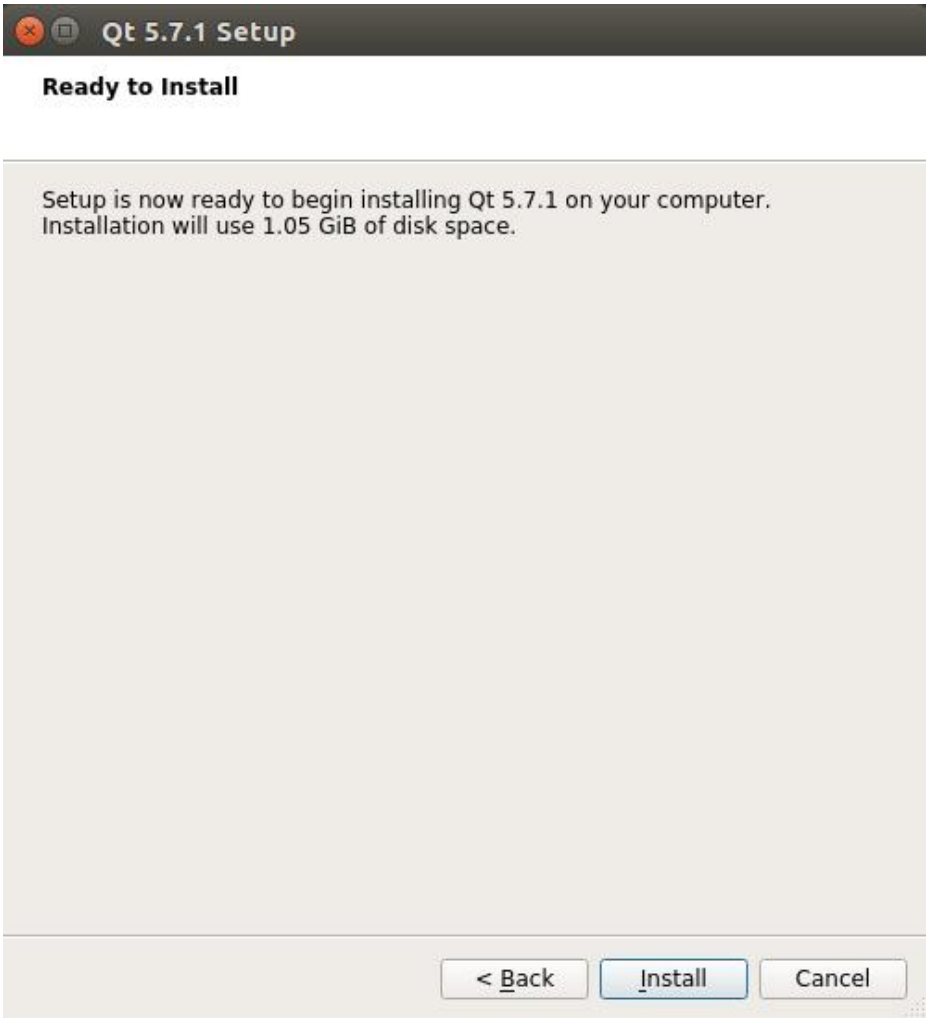
8) Keep the default installation options, click "Next" to continue the installation



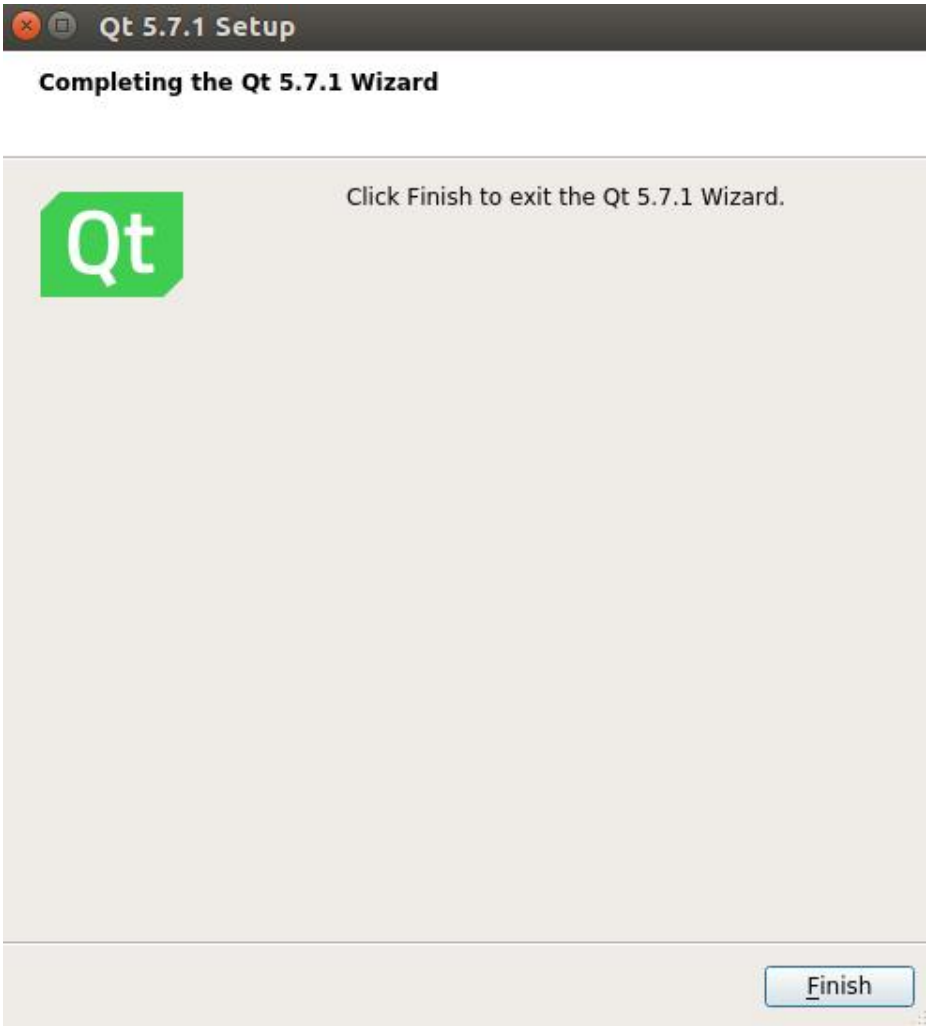
9) Choose to agree to the license



10)Click "Install" to install



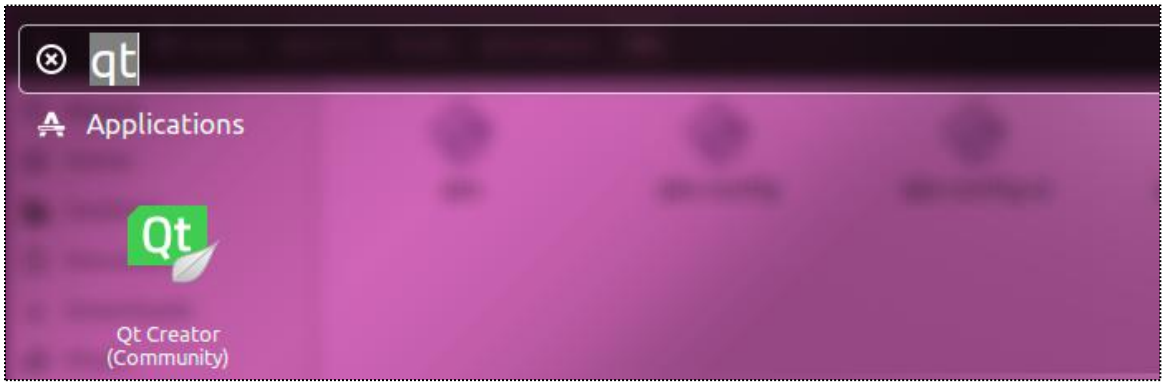
11)Click "Finish" to complete the installation



Part 6.2: hello world in Qt Creator

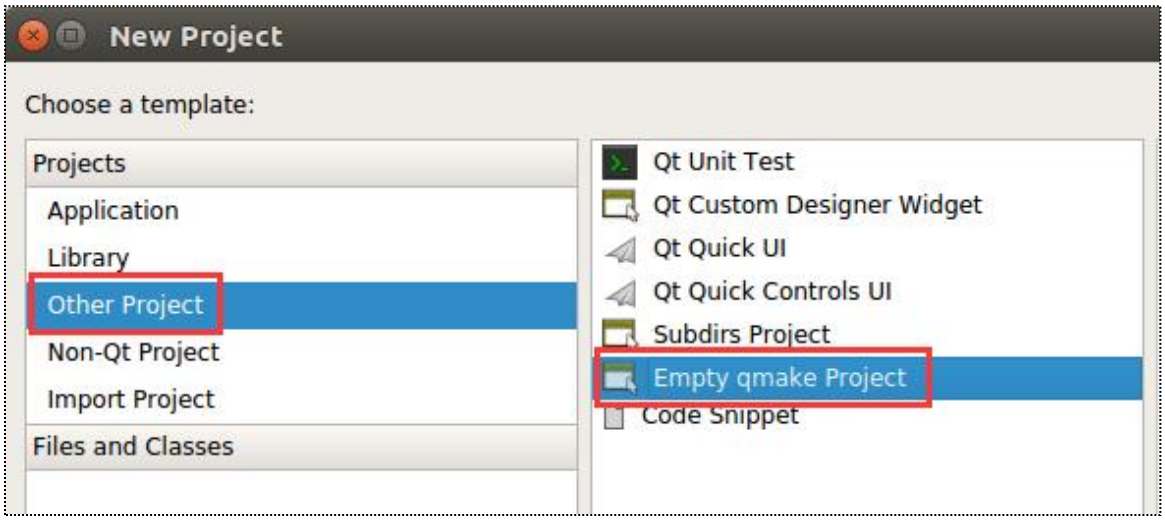
Part 6.2.1: Create project

1) Press the "win" key of the keyboard to open the application search, enter "qt", double-click the Qt Creator icon to open the Qt Creator software



2) Click the following location to open the new project wizard

3) Select the picture below and click the "Choose..." button



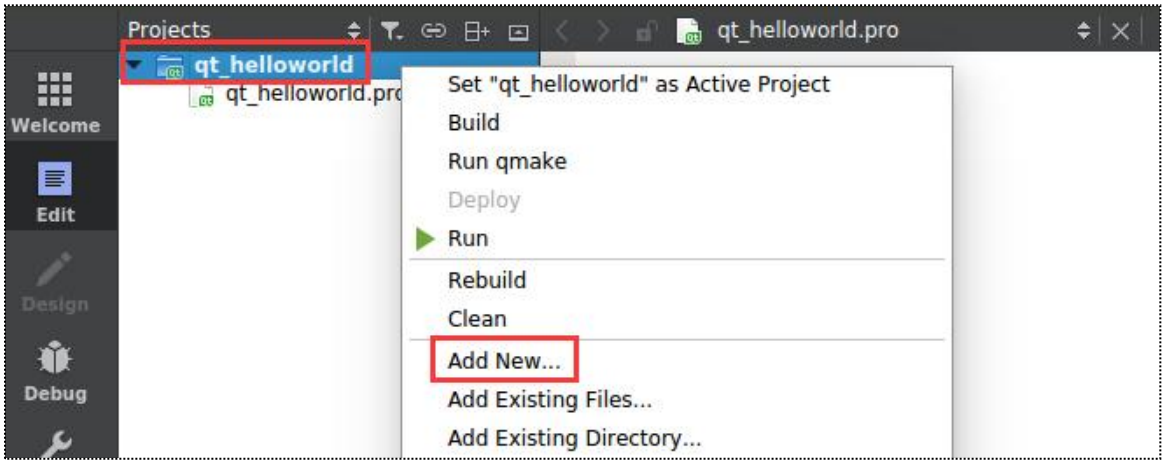
4) Set the project name and save path, click the Next button

5) Keep the current settings, click the Next button

6) Click the Finish button to end the project creation

Part 6.2: Project compilation

1) Right-click the project, and in the pop-up menu, select "Add New..."



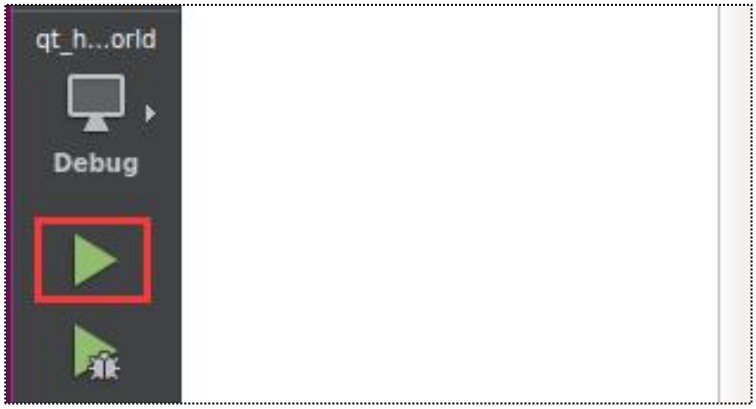
2) Choose to create a C++ source file and click the "Choose..." button

3) Set the file name to main.c, click the Next button

4) Click the Finish button to finish adding the main.c file

5) Double-click main.c, open the file, enter the **hello world** code, and press Ctrl+S to save the document

6) Click the icon below to edit and run the program



7) The results of the operation are as follows

8) The program compiled this time is running in the current host environment. As you can see, the program has run correctly.

Part 7: Linux Common Commands

Part 7.1: File and folder operations

Part 7.1.1: Creat a new folder

mkdir dir

Part 7.1.2:Create a folder for multiple levels of directories

mkdir -p dir1/dir2

Part 7.1.3: Change current directory

cd newdir: Enter the newdir directory

cd .. : Enter the upper directory

cd-: Enter the last path

cd: Enter the home directory

Part 7.1.4: Copy file

cp source_file dest_file: copy “source_file” to “dest_file”

cp file1 file2 dir: Copy “file1” and “file2” to the folder “dir”

cp -r source_dir dest_dir: Retain attribute copy, equivalent to rsync -a source_dir/ dest_dir/

ln -s linked_file link: Create a soft connection

Part 7.1.5:File Rename

mv source_file dest_file source_file rename to dest_file

Part 7.1.6:Delete files and folders

rm file1 file2: Delete file or file link

rmdir dir: Delete an empty folder

rm -rf dir: Delete a non-empty folder

Part 7.2: Document List

Ls: Most commonly used, displays all files in the current folder, does not contain hidden files

ls -l: Show more file information

ls -a: Show all files with hidden files

Part 7.3: Compression and Decompression

Part 7.3.1: Create a Compressed File Package

tar zcvf archive.tar.gz dir/

Part 7.3.2: Extract a "tar.gz" File

tar zxvf archive.tar.gz

Part 7.4: System Management

ifconfig –a: Show all available network interfaces

ping 192.168.1.1: Test network and additional host connectivity