Assignment3 Bonus Part Environment Setup

1. Environment Setting

- VirtualBox
- Ubuntu-18.04

  These are the platforms to run Bochs IA-32 Emulation.

- gcc-3.4 & g++3.4
- cpp-3.4.6 & libc++6

  Compiler and cpp for the linux-0.11 source code

- Bochs IA-32 Emulation Platform

Bochs is a highly portable open source IA-32 (x86) PC emulator written in C++, that runs on most popular platforms. It includes emulation of the Intel x86 CPU, common I/O devices, and a custom BIOS. Bochs can be compiled to emulate many different x86 CPUs, from early 386 to the most recent x86-64 Intel and AMD processors which may even not reached the market yet. Bochs is capable of running most Operating Systems inside the emulation including Linux, DOS or Microsoft Windows. You could get more information about this from this link: https://bochs.sourceforge.io/

In this assignment, Bochs would be the x86 PC emulator to run the linux-0.11 source code. After the whole installation and start Bochs x86, you would see the linux-0.11 system could be loaded as red frame in Figure1:

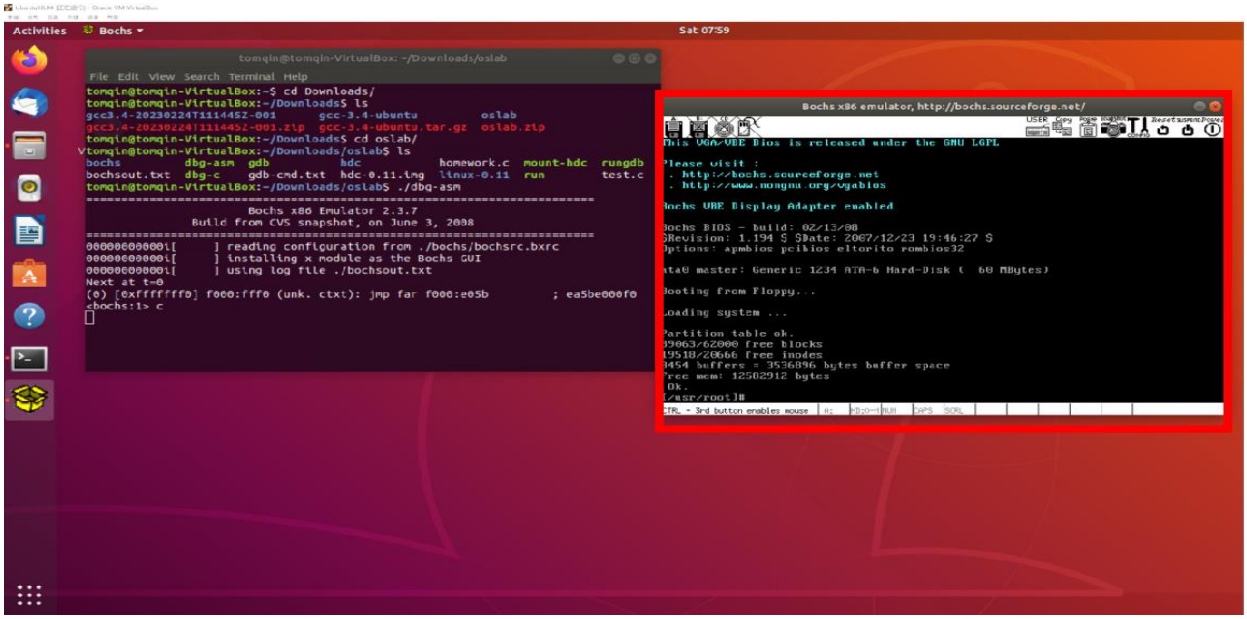

Figure 1

- Linux-0.11 source code (oslab code)

  The source code for the earlier linux system

2. Install the Ubuntu-18.04 on the VirtualBox

1) Download the VirtualBox from the following link:

   https://www.virtualbox.org/wiki/Downloads

   You could choose windows host and mac host according to your os in your computer

2) Download the Ubuntu18.04 iso

   https://releases.ubuntu.com/18.04/

   Choose ubuntu-18.04.6-desktop-amd64.iso and download

3) Install the VirtualBox on your PC

4) Load the Ubuntu18.04 iso into the VirtualBox

   First step: Click 'New' button on the top and then load the Ubuntu18.04 iso. The Name is filled with 'Ubuntu18.04'. The Iso Image is selected as where you save Ubuntu18.04 iso.

   A reminding is that you need to click 'Skip Unattended Installation'. Or you would find some troubles when you configure the network and find the terminal. (Seen as Figure 2)

   Second step: Configure the Memory Capacity, cpu number and hard disk size. (Seen as Figure 3 and Figure 4)

   Third step: Then we could finish the loading step of Ubuntu18.04 iso into the Virtual Box.

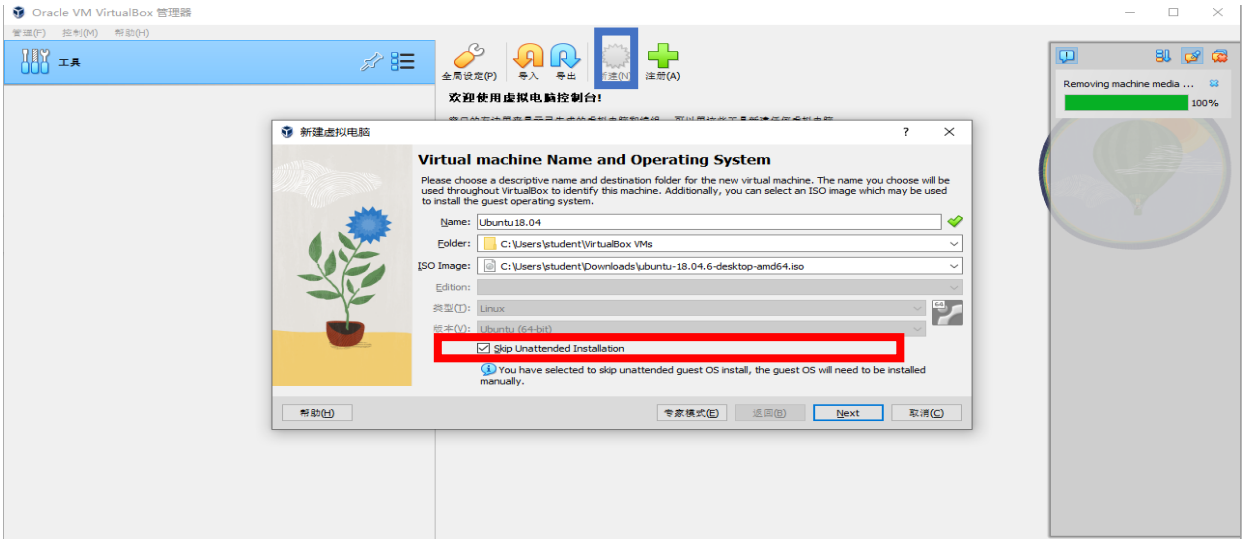

Figure 2

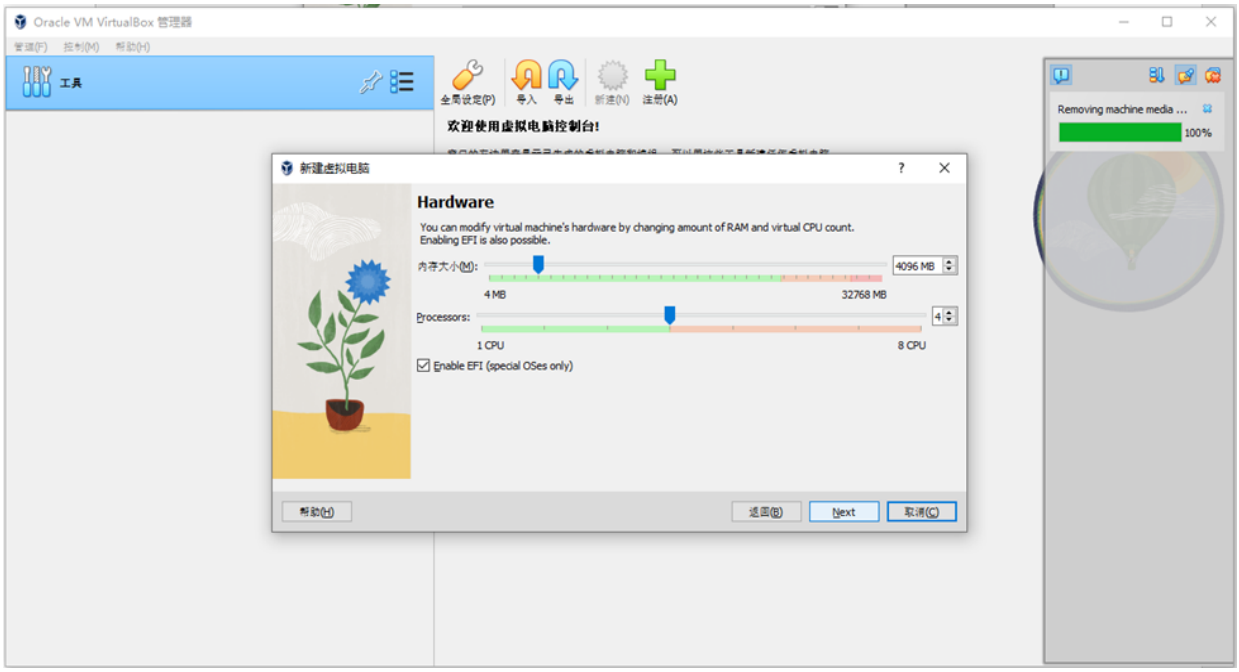

Figure 3

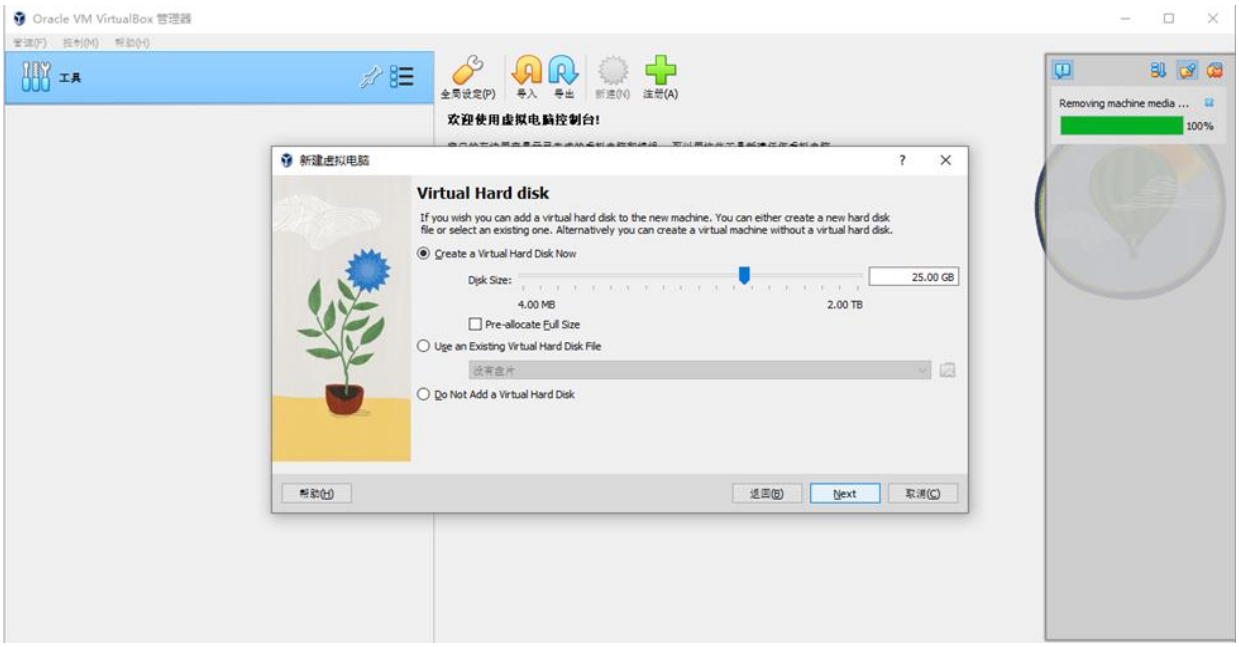

Figure 4

5) Install the Ubuntu18.04

First step: Click 'Start' button on the top, and then the virtual machine would power up. After the system starts up, we would find the 'Install Ubuntu 18.04.6 LTS' on the left up corner of the desktop. Click it. (Seen as Figure 5)

Second step: In the language and Keyboard layout setting, you should choose **'English' and 'English(US)'.** (seen as Figure 6&7)

Third step: In the 'Updates and Other software' and 'Installation Type', you should choose 'Normal Installation' and 'Erase Disk and install Ubuntu', and click 'Install now'.

Fourth step: In the 'Who are you', you should enter the username and password in 'Your name' and 'choose a password'. **Please remember it well and you would need to log in with it.** Then continue and installation begins.

Fifth step: After installing and restarting your virtual system, **checking whether you could open your terminal and the connection of your network! (Figure 10)**

Check Terminal: Press 'Activity' and Input 'Terminal', and then click 'Terminal' icon. (Figure 9)

Check network connection: Open 'Firefox' browser and a website.

Sixth step: **Install Upgrade Guest Addition.** Click 'Device' and then 'Upgrade Guest Addition'. Also let the 'shared paste' as 'bidirectional'. This would let you paste things between vm and your pc. (Seen as Figure 11)

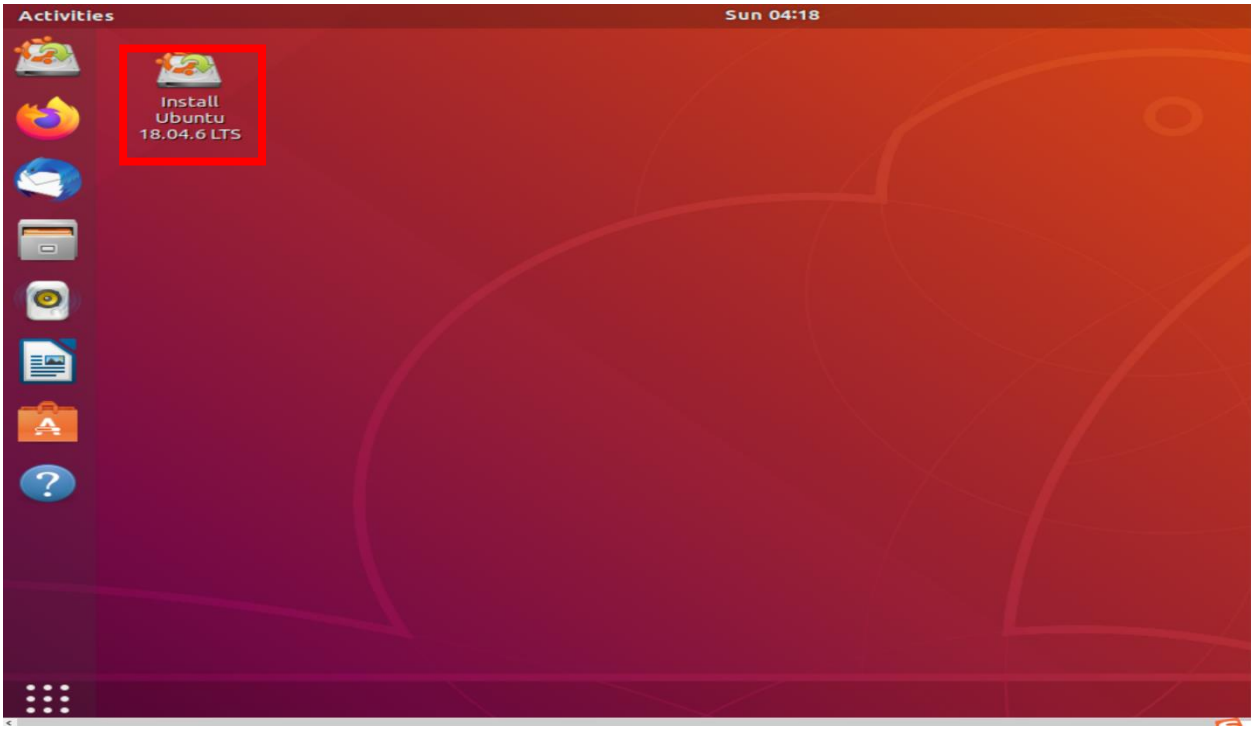

Figure 5

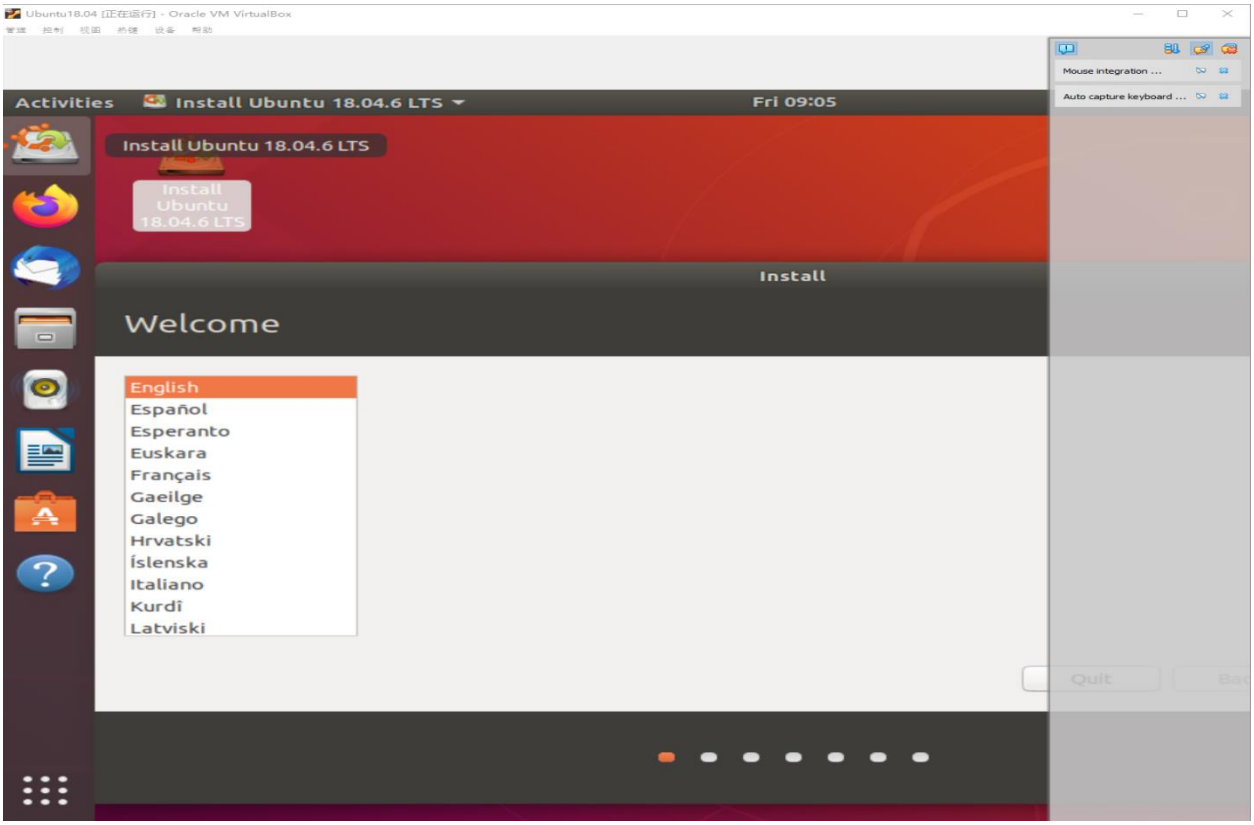

Figure 6

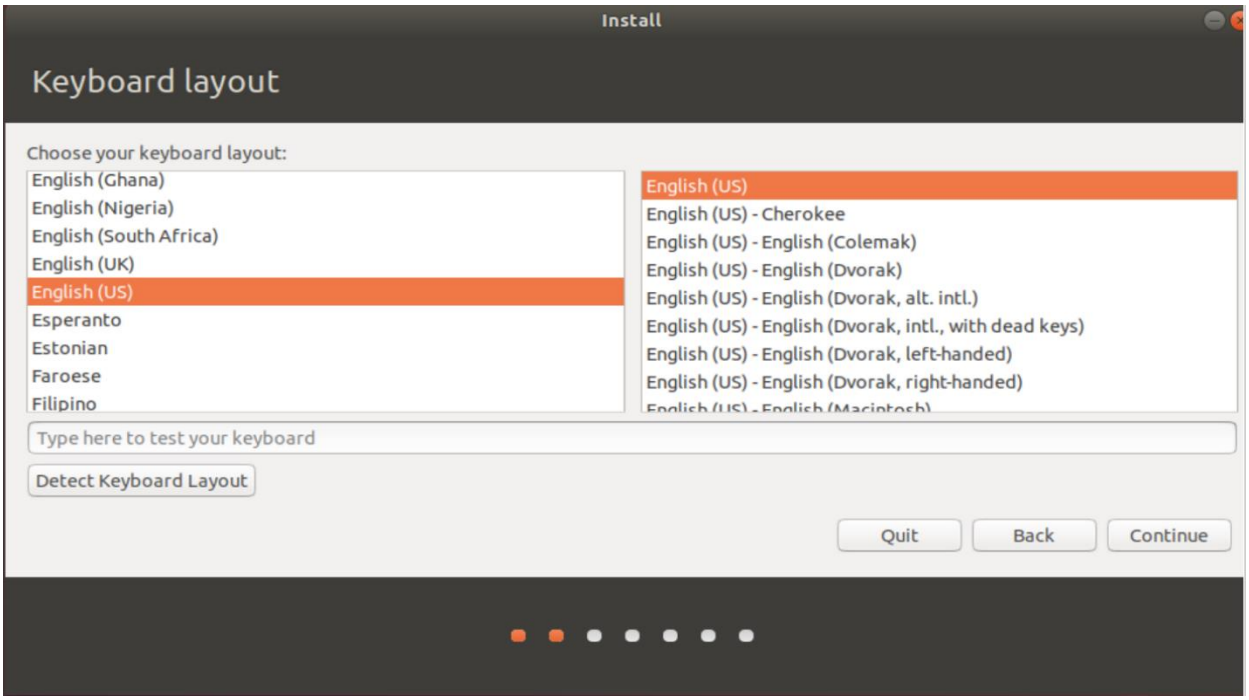

Figure 7

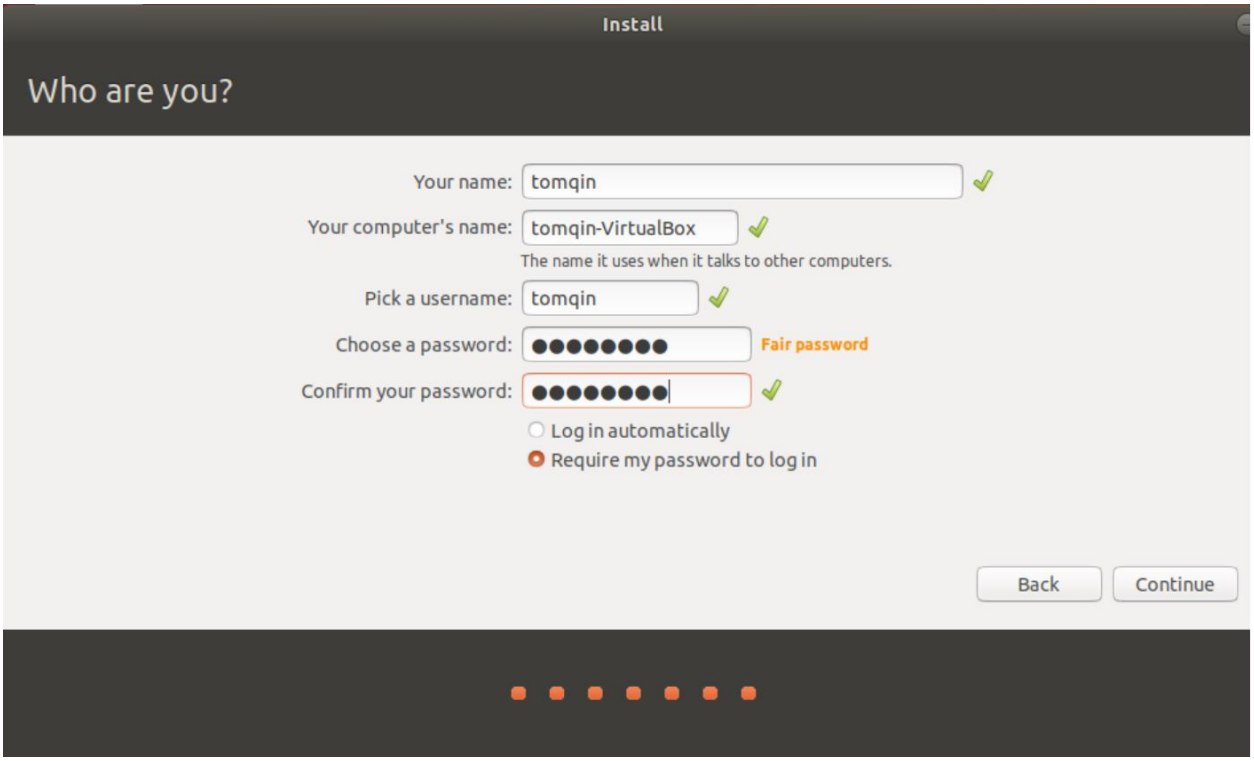

Figure 8

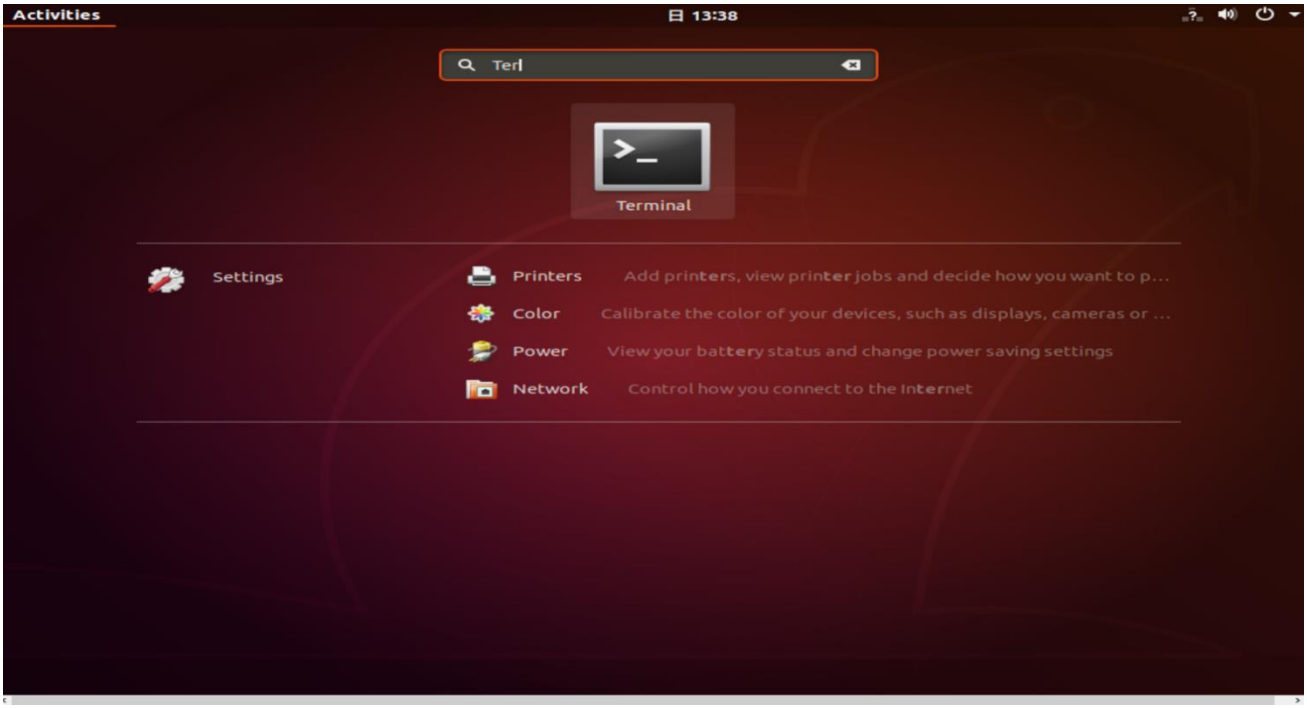

Figure 9

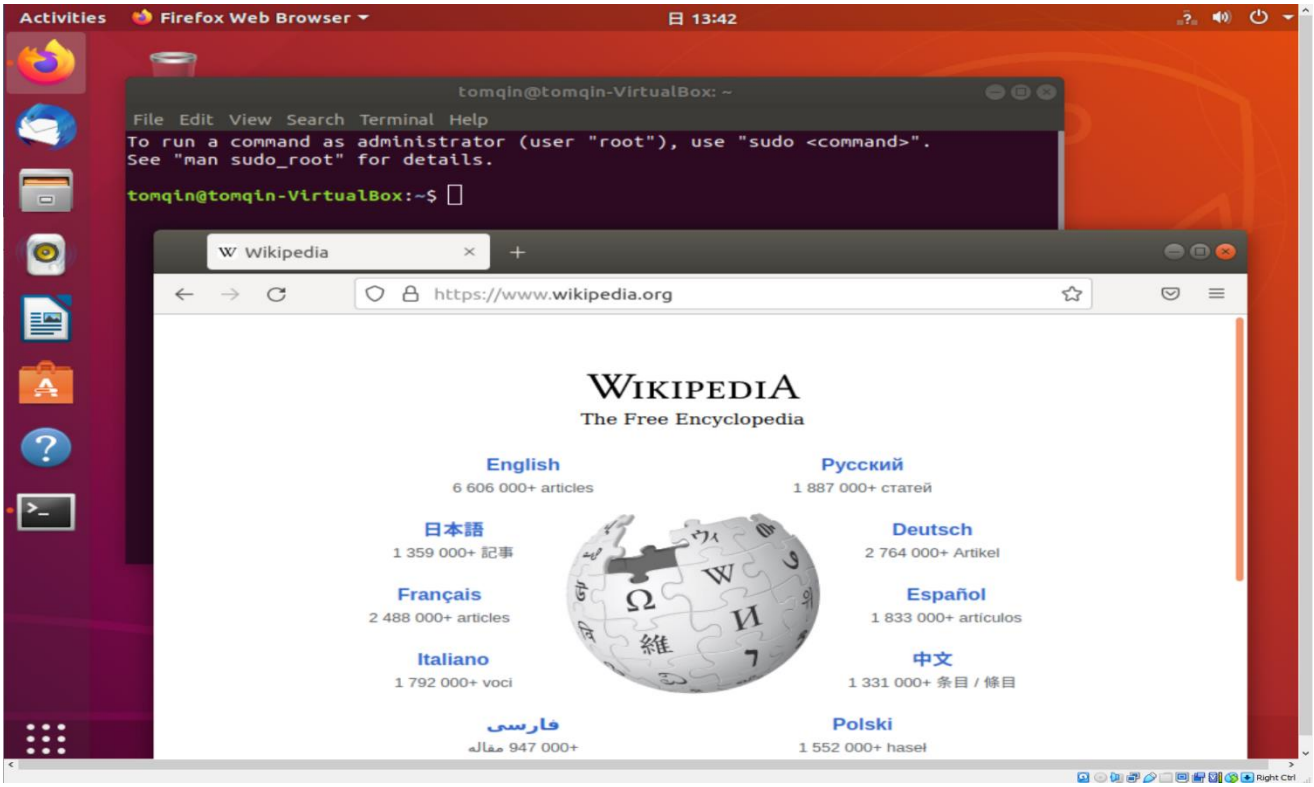

Figure 10

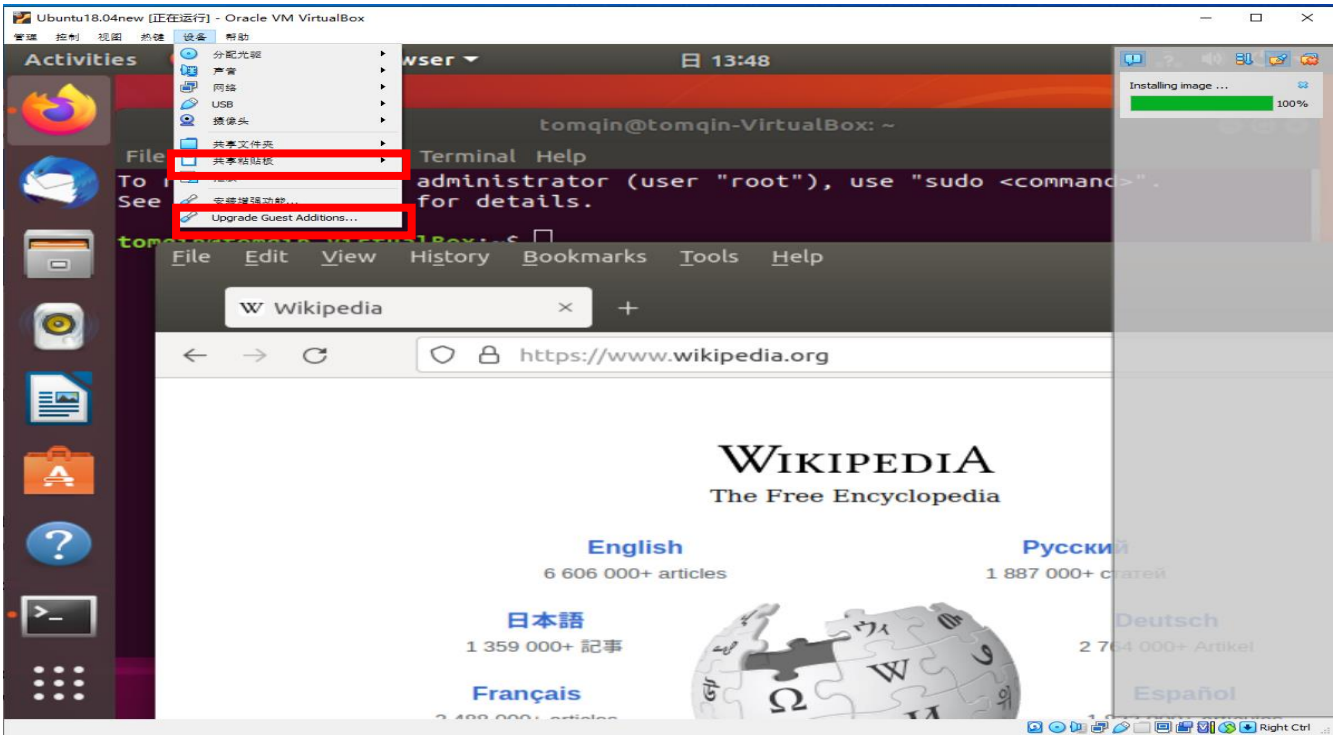

Figure 11

3. Install the gcc-3.4 and g++-3.4 compiler in the system

1) Download the compiler package on the PC

   Download link:
   https://drive.google.com/drive/folders/1C88TWhOxB5UVpD6b7ETvDTLHWj59qjN2?usp=share_link

2) Extract the files and open the terminals. Use 'cd' to arrive at the download place. We could see 5 files in the package. (Figure 12)

   Then we input the following commands in the terminal step by step (Figure 13)

   sudo dpkg --force-depends -i gcc-3.4-base_3.4.6-6ubuntu3_amd64.deb

   sudo dpkg --force-depends -i gcc-3.4_3.4.6-6ubuntu3_amd64.deb

   sudo dpkg --force-depends -i cpp-3.4_3.4.6-6ubuntu3_amd64.deb

   sudo dpkg --force-depends -i g++-3.4_3.4.6-6ubuntu3_amd64.deb

   sudo dpkg --force-depends -i libstdc++6-dev_3.4.6-6ubuntu3_amd64.deb

3) After entering these command, we would find a call saying **'libstdc++6-dev: dependency problems'** (Figure 14). Then we could solve it by entering these commands:

sudo -i

apt --fix-broken install

exit

4) We could use the following commands to check the installation status of gcc-3.4 and g++3.4 (Figure 15)

ls /usr/bin/gcc* -ll

ls /usr/bin/g++* -ll

If there is no gcc-3.4 and g++-3.4, then you should return to step 2) to install the corresponding package again.

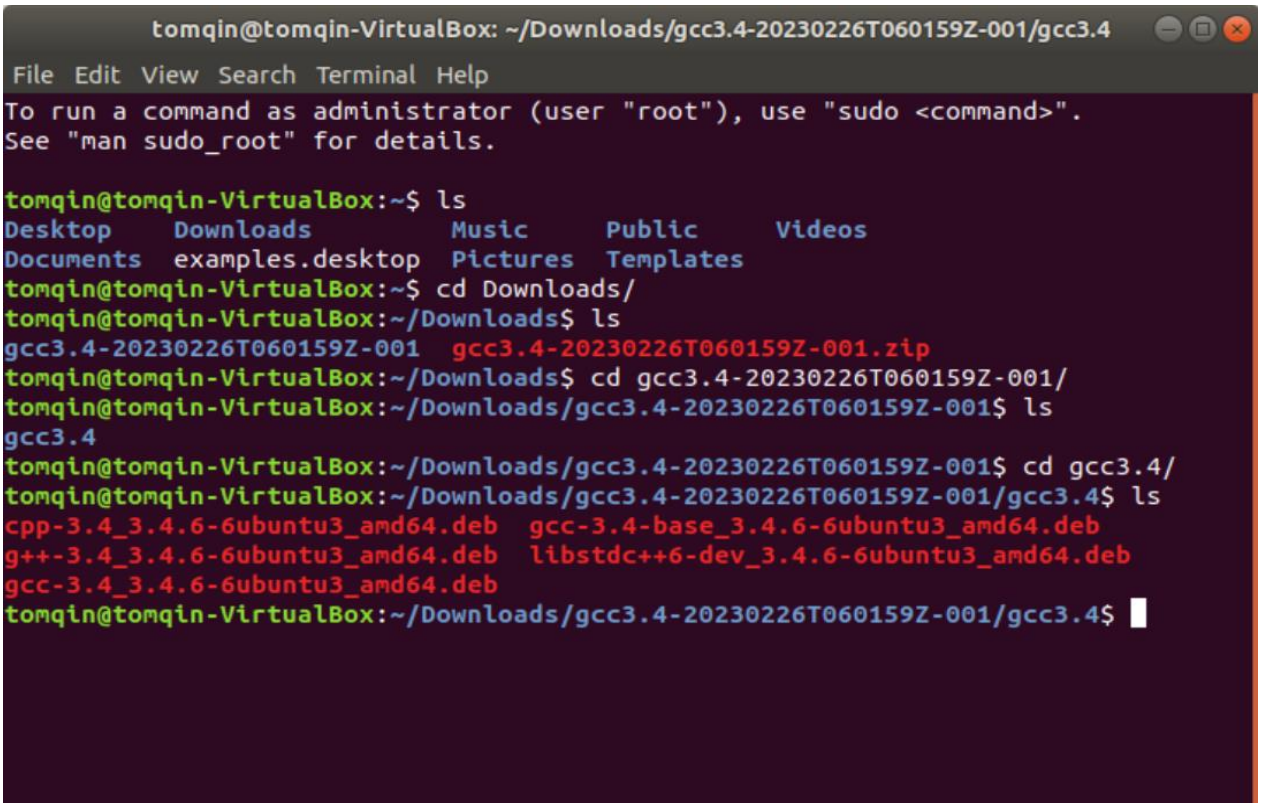

Figure 12

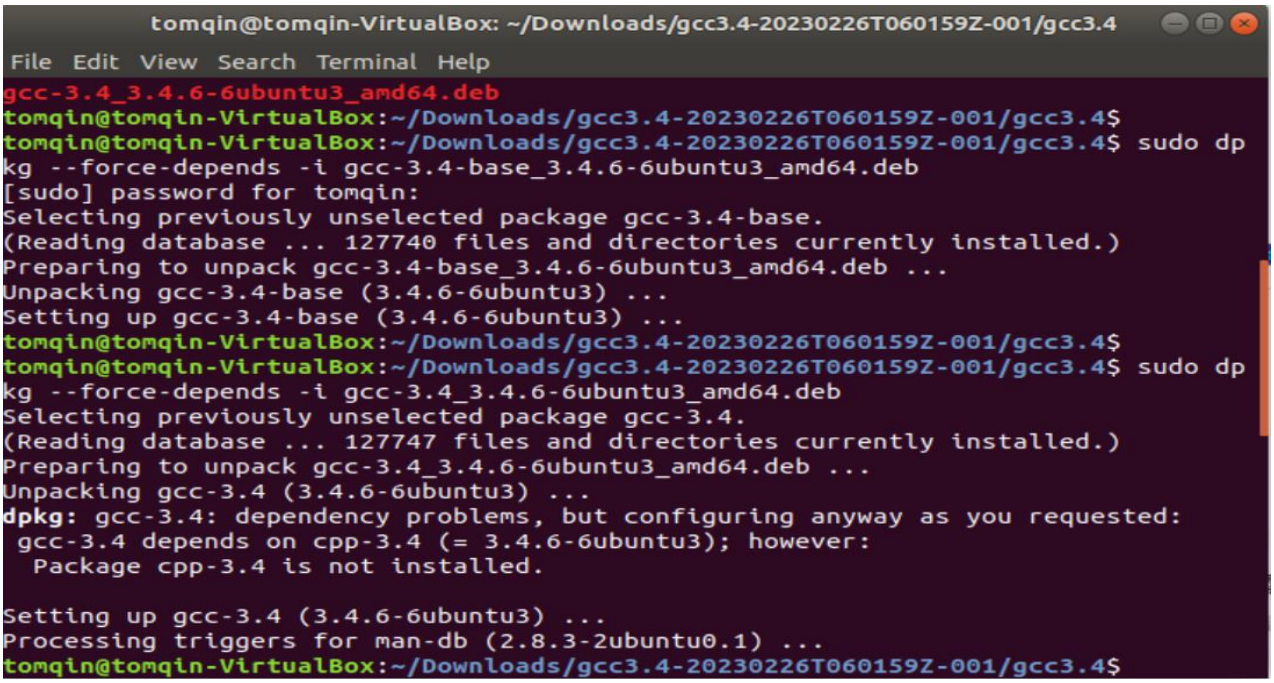

Figure 13

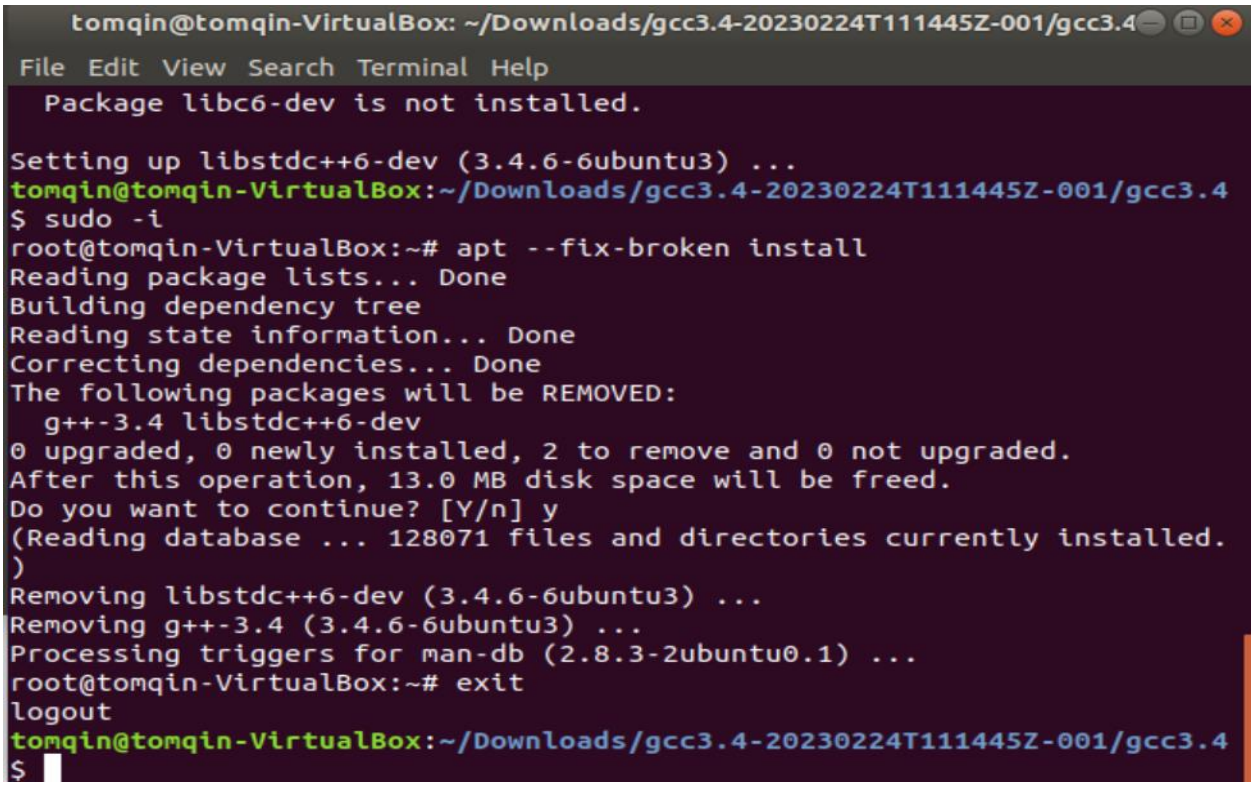

Figure 14

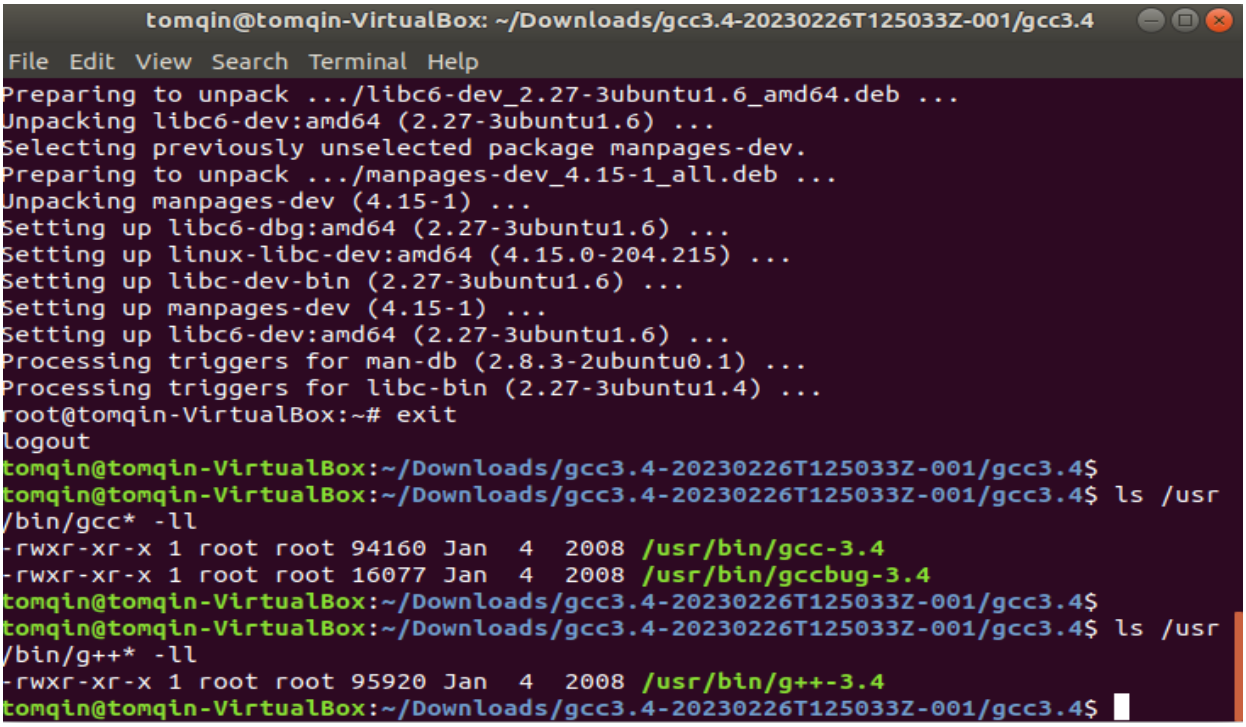

Figure 15

4. Install Bochs and Switch compiler version

1) Install some dependent packages firstly using the following commands (Figure 16):

sudo apt-get install bin86

sudo apt-get install gcc-multilib

sudo apt-get install build-essential

2) Install the Bochs (Figure 17)

sudo apt-get install bochs bochs-x bochs-sdl

3) Set the priority of different version of gcc and g++ using the command belows (Figure 18):

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-3.4 30

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-7 70

sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-3.4 30

sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-7 70

4) Switch the gcc and g++ version (Figure 19) Firstly, input this command,

sudo update-alternatives --config gcc

and then type the selection number **1**.

The same is for the g++, input this command,

sudo update-alternatives --config g++

and then type the selection number **1**.

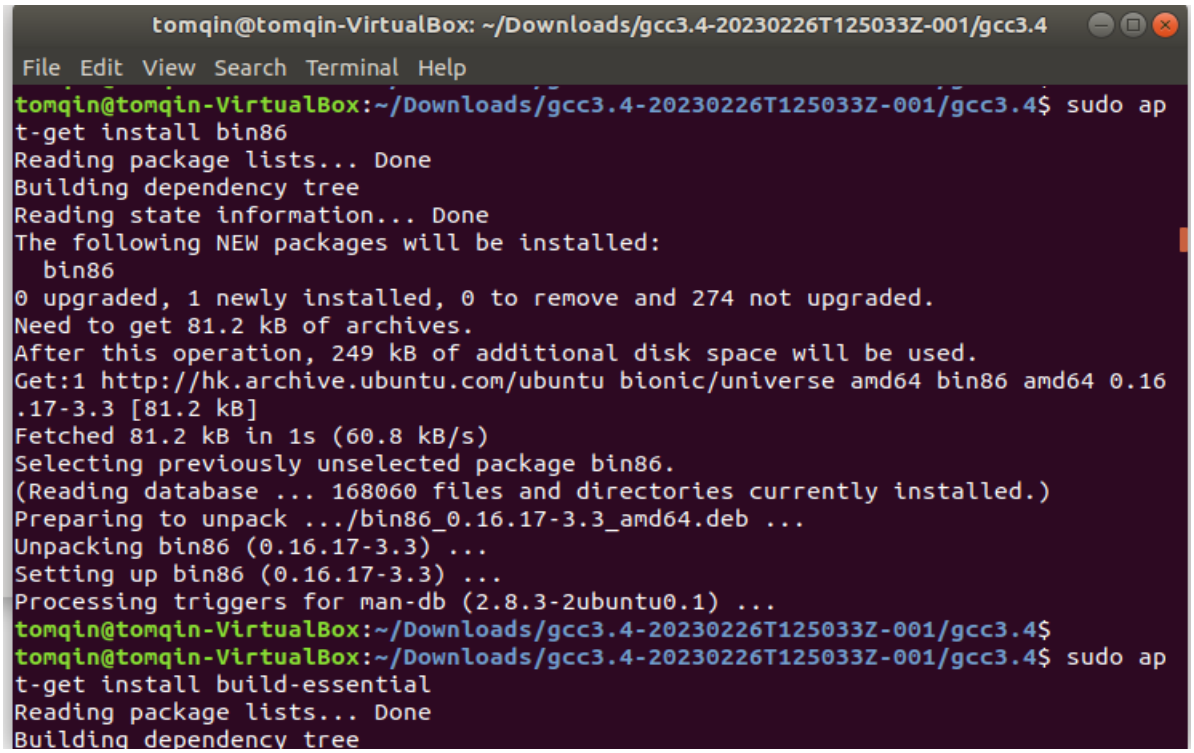

Figure 16

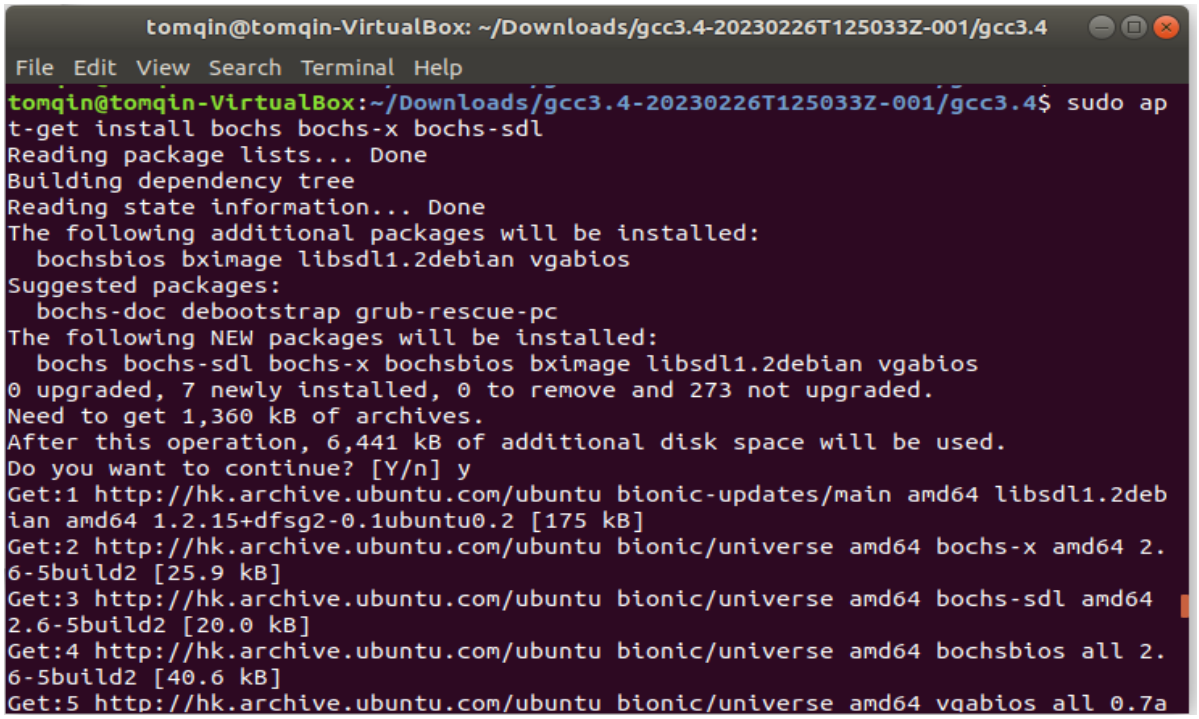

Figure 17

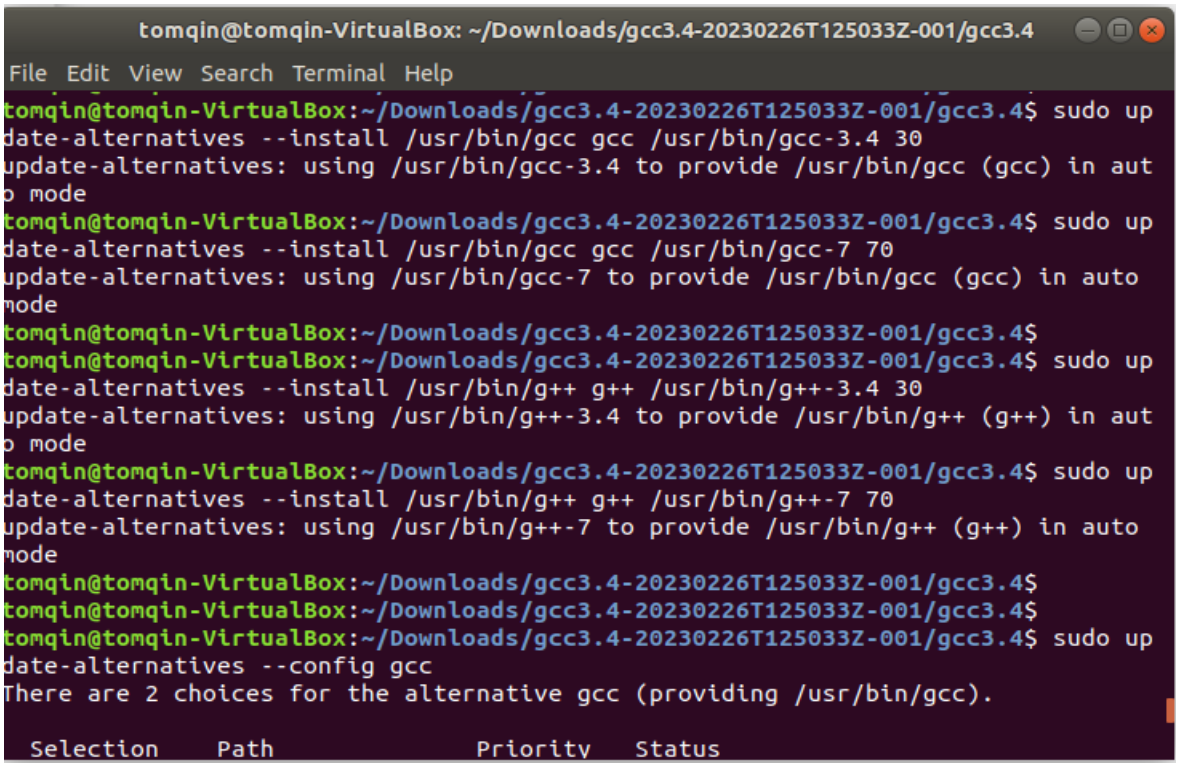

Figure 18

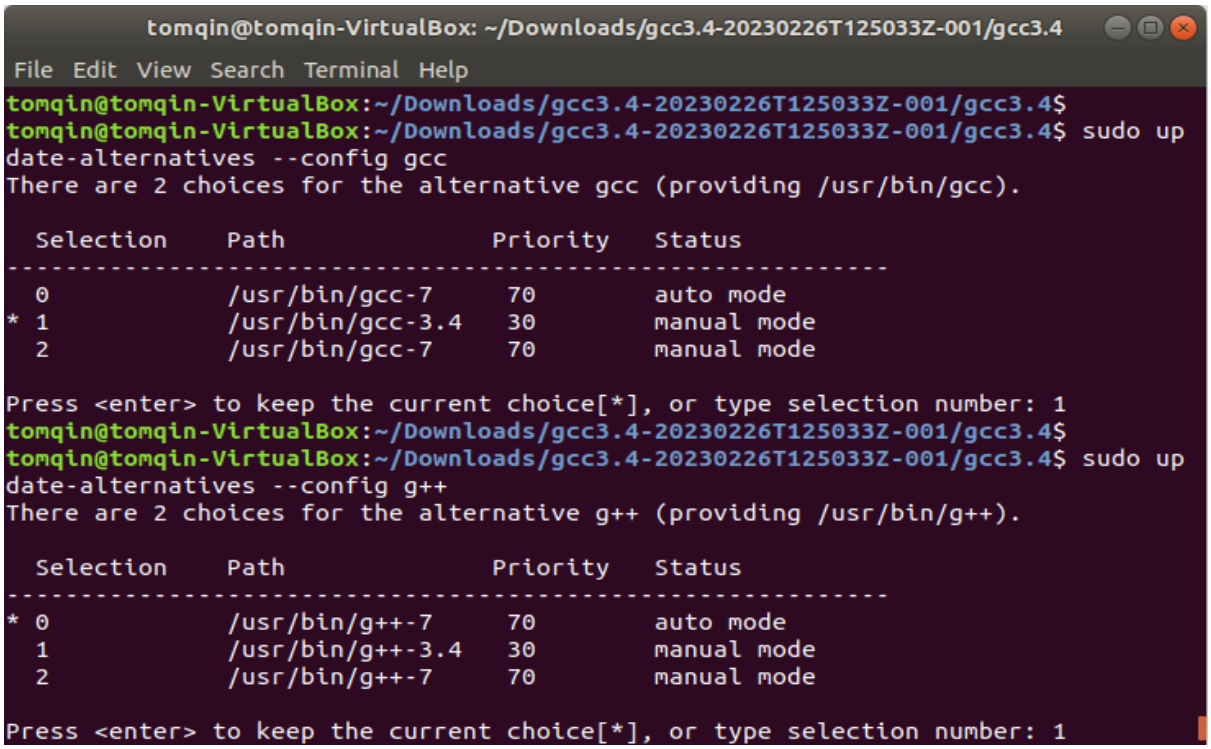

Figure 19

5. Download oslab source code and run it

1) Download oslab source code from this link:

   https://drive.google.com/file/d/16jBb8X5MgZCp0VbAY3bA8PV_BkgbVpb2/view?usp=share_link

2) Then we use 'cd' to the oslab directory, and input the command:

   ./dbg-asm

   A report from the terminal says that **'./bochs/bochs-dbg: error while loading shared libraries: libSM.so.6: cannot open shared object file: No such file or directory'**. That means we should install some package. Then we use these commands to install the needed packages (Figure 20):

   sudo apt-get install libsm6:i386

   sudo apt-get install libx11-6:i386

   sudo apt-get install libxpm4:i386

3) We input the command **./dbg-asm** again. Then the bochs x86 Emulator starts. In the bochs x86 Emulator,

   Ctrl+c means pausing the program and it would show what it executes before pausing.

   c means continuing running the program

Then we input ‘c’ here, then we would see the initialize page in the shell. Then we input ‘ls’, we could see that the example in the tutorial is ‘test.c’ file. And the homework problem 1 is the ‘homework.c’ file.

Then we use the following command to compile the ‘test.c’ file:

gcc -o test test.c

Then use ‘./test’ to run the excutable file, we could see that the program starts running.

You could see the Figure 21 to have a better understanding of this step 3).

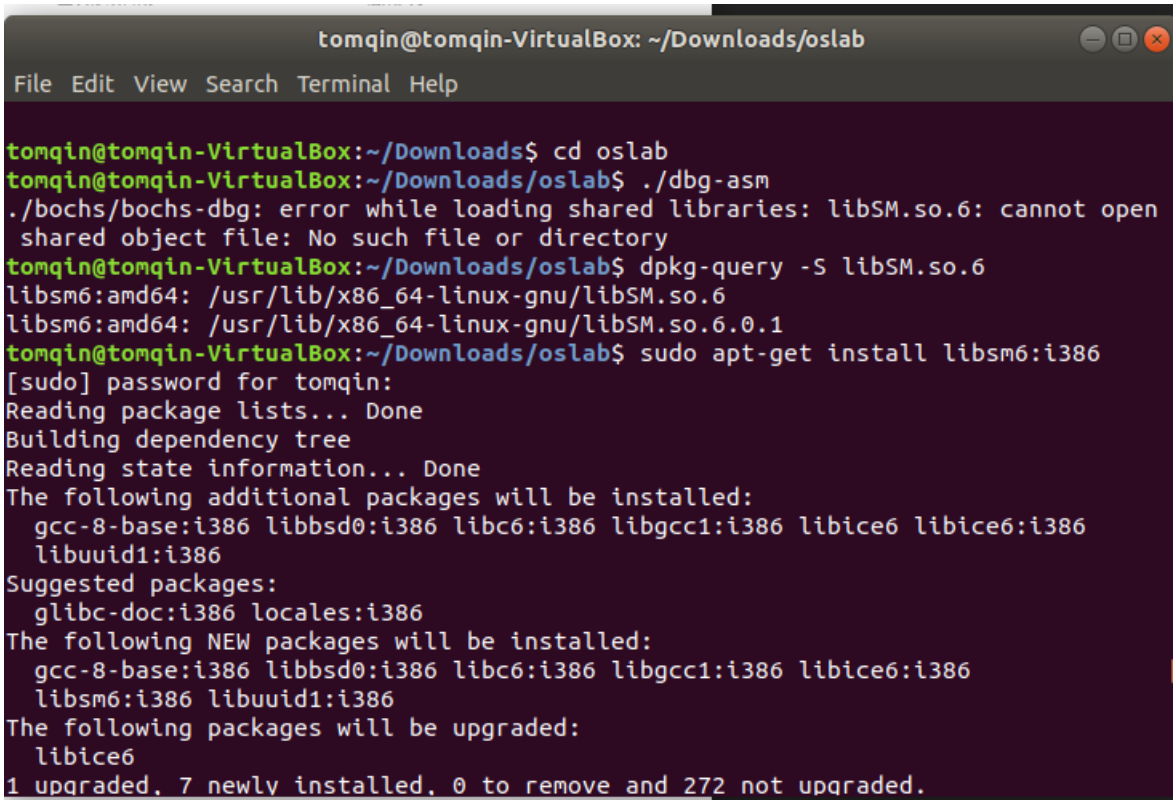

Figure 20

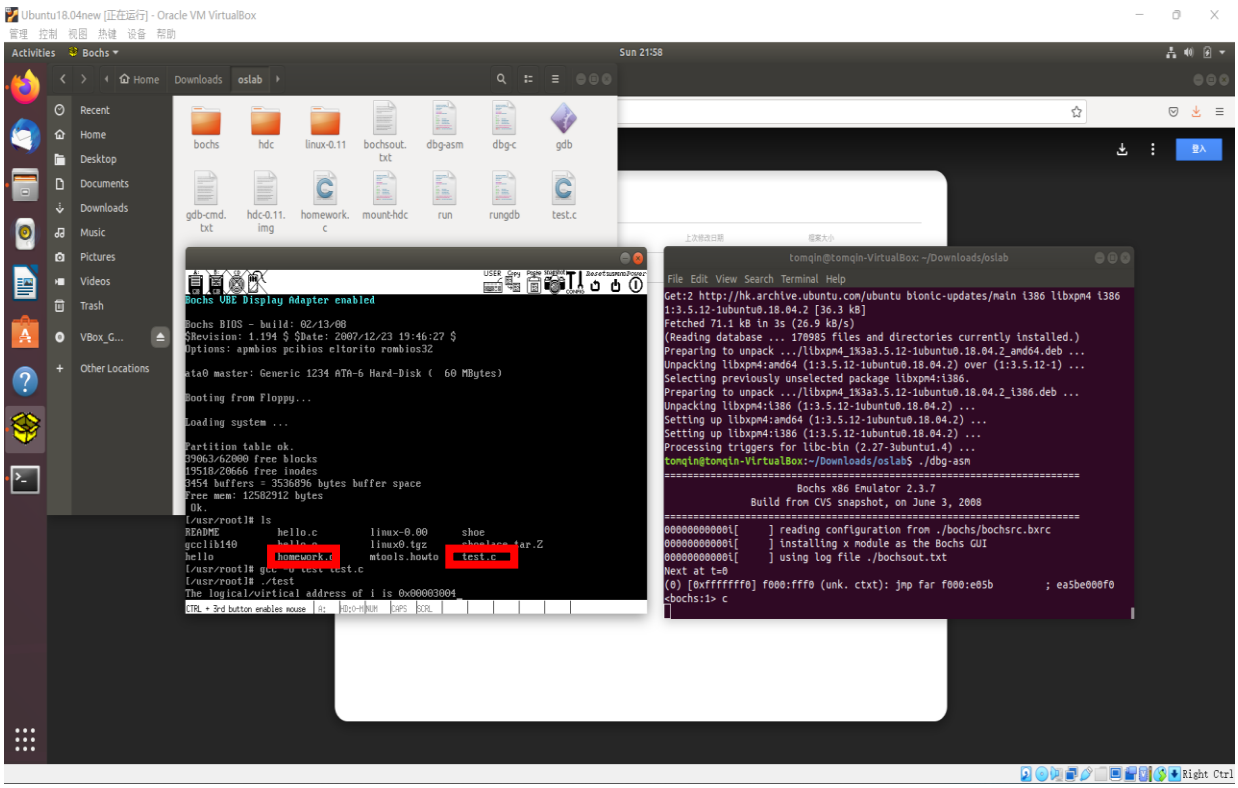

Figure 21

6. Some basic commands in Bochs

- **Ctrl+C and c**

Ctrl+c means pausing the program and it would show what it executes before pausing. c means continuing running the program from the place where the program is paused by Ctrl+c.

- **sreg**

sreg is the command used for seeing the status of the segment registers. As figure 22 shows, we could see the segment descriptors and segment selectors in different segment registers. Taking ‘ds’ for example, it is the data segment, and the segment descriptor of the data segment is stored in dh and dl, which is 0x10c0f300 and 0x00003fff. The segment selector of the data segment in LDT is stored in s, which is 0x0017.

```
<bochs:9> sreg
cs:s=0x000f, dl=0x00000003, dh=0x10c0fa00, valid=1
ds:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=3
ss:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
es:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
fs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
gs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
```

Figure 22

- **creg**

creg is the command used for seeing the status of the control registers, like cr0-cr3 (Figure 23) We should notice that cr3 is the register that stores the starting address of the first-level table, which is 0x00000000.

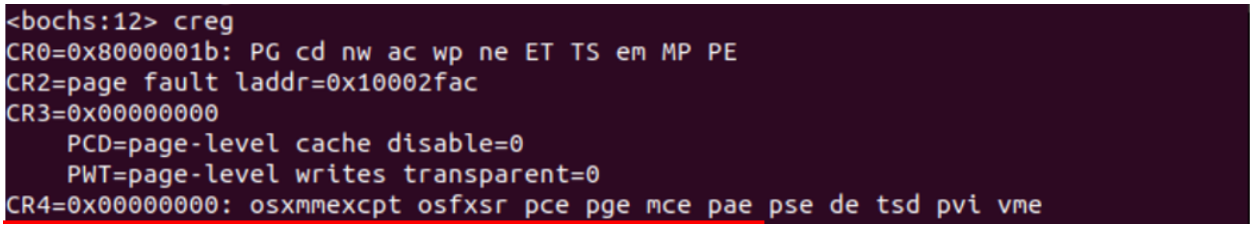

Figure 23

- **xp**

xp is command used for seeing the contents in the memory cells with the given physical address. For example, if we inputs:

xp /w 0x00fa6000

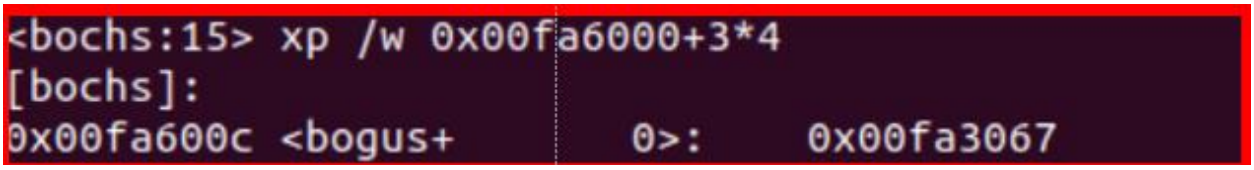

Figure 24

/w means that you could see one word from the memory cell with the physical address 0x00fa6000+3*4. And one word in Bochs represents 4 bytes (32bits). The address is arranged by bytes. If the starting address of the page table is 0x00fa600, and one page item occupies 4 bytes (32 bits), the starting address of the 4-th page item is 0x00fa600+3*4

- **setpmem**

setpmem is command used for setting the corresponding memory cell with the given number. For example, if we inputs:

setpmem 0x00fa3004 4 0

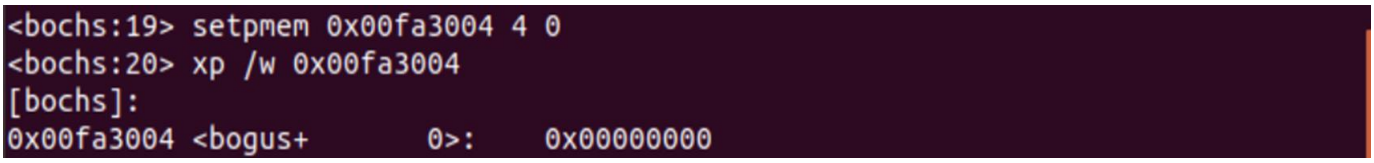

Figure 25

Then it would set the contents in the memory cells with 4 bytes starting from the physical address 0x00fa3004 zero.

- **exit**

  This command would lead to the exit of the Bochs Emulator.