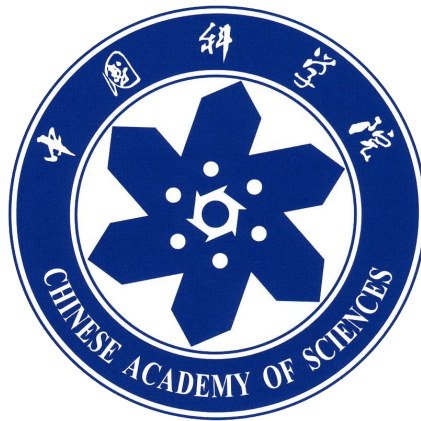


# 国科大操作系统研讨课任务书

## Project 0



版本：1.0

一、引言.....	3
二、开发环境快速搭建.....	4
2.1 windows 实验环境搭建.....	4
2.2 Linux 实验环境搭建.....	11
2.3 Mac 实验环境搭建.....	13
附录.....	21
2.1 实验环境详细搭建.....	21
2.2 使用 xshell 访问虚拟机.....	24
2.3 MIPS 相关知识.....	27
2.4 龙芯 QEMU 模拟器使用介绍.....	31
2.5 Linux 的使用.....	35
2.6 MAKEFILE 的使用.....	37

## 一、引言

操作系统是计算机系统的重要系统软件，也是计算机专业教学中的一个重要内容。该门课程内容复杂且抽象，我们认为掌握操作系统原理的最好方法就是自己编写一个操作系统，因此我们更希望的是同学们能够通过自己的努力去实现一个自己的操作系统，并在实现的过程中不断加深对操作系统的理解，而不是让大家只停留在理论课上的“纸上谈兵”。

通过该实验课，大家将由浅入深，从操作系统的引导到操作系统复杂的内部实现，一步一步，深刻理解操作系统中进程的管理、中断的处理、内存的管理、文件系统等相关知识，并将其实现出来。我们衷心希望每个选修完这门课的同学以后都能自豪的说：“我自己实现过一个完整的操作系统”。

最后，希望大家如果在实验的过程中发现有什么问题，请积极和老师、助教反映，提出自己的意见。对于学有余力的同学，非常欢迎大家能够加入我们，一起将国科大的操作系统实验课做的更好。接下来，我们将从环境搭建开始，一步一步实现我们的操作系统 UCAS-OS 吧！

二、开发环境快速搭建

为了让大家快速完成开发环境的搭建，我们已经将开发所需的环境集成到我们给所给的 VirtualBox 虚拟机镜像中，同学们只需要安装完成 VirtualBox 后导入我们所给的镜像，并简单的配置一下即可。当然，我们也在附录中给出了环境的具体流程，有感兴趣的同学可以了解一下。

环境搭建所需的工具，我们已经拷贝到发给大家的 SD 卡中，此外我们将工具也上传到了百度云中，**链接为：**  
**<https://pan.baidu.com/s/1UQrSMfjNJ4uRJeei-ElAiw> ， 提取码：2qzp**

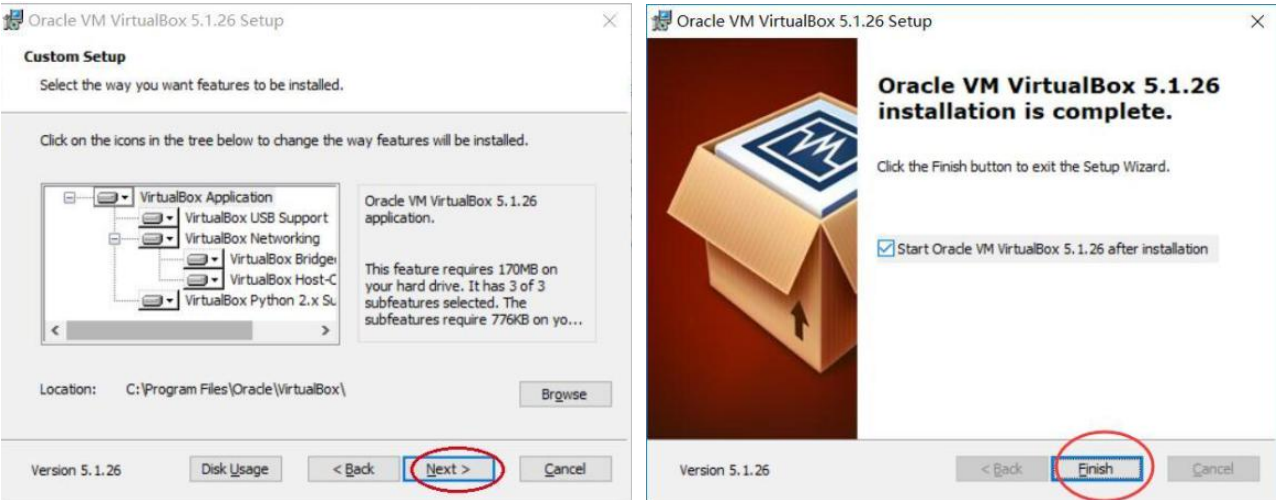
2.1 windows 实验环境搭建

2.1.1 所需文件

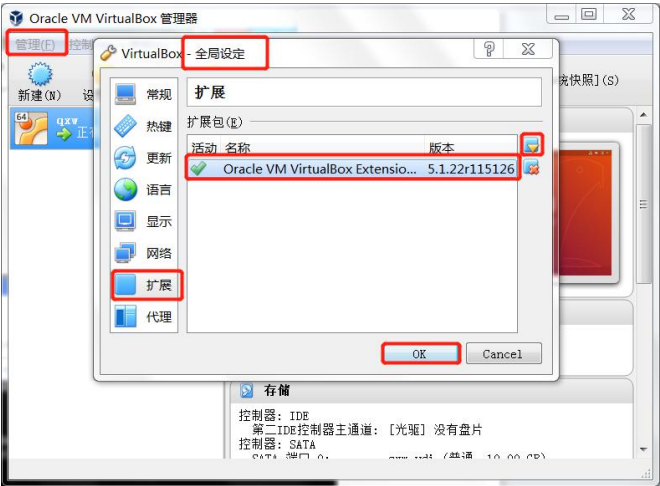
	文件名称	说明	对应网盘目录
1	VirtualBox-5.2.18-124319-Win.exe	VirtualBox windows 安装包	Windows_virtualbox
2	PL2303_Prolific_DriverInstaller_v110.exe	Windows7 串口驱动	windows 串口驱动
3	ubuntu_server_12.04.ova	VirtualBox 虚拟机镜像	已有交叉编译环境的镜像
4	Oracle_VM_VirtualBox_Extension_Pack-5.2.18.vbox-extpack	VirtualBox 拓展包	Windows_virtualbox

2.1.2 VirtualBox 软件的安装

(1) 右键点击安装包，选中以管理员方式运行安装包，完成快速安装。



(2) 安装 VirtualBox 扩展包。运行 VirtualBox->管理->全局设定->扩展->选择扩展包目录->安装->重启。

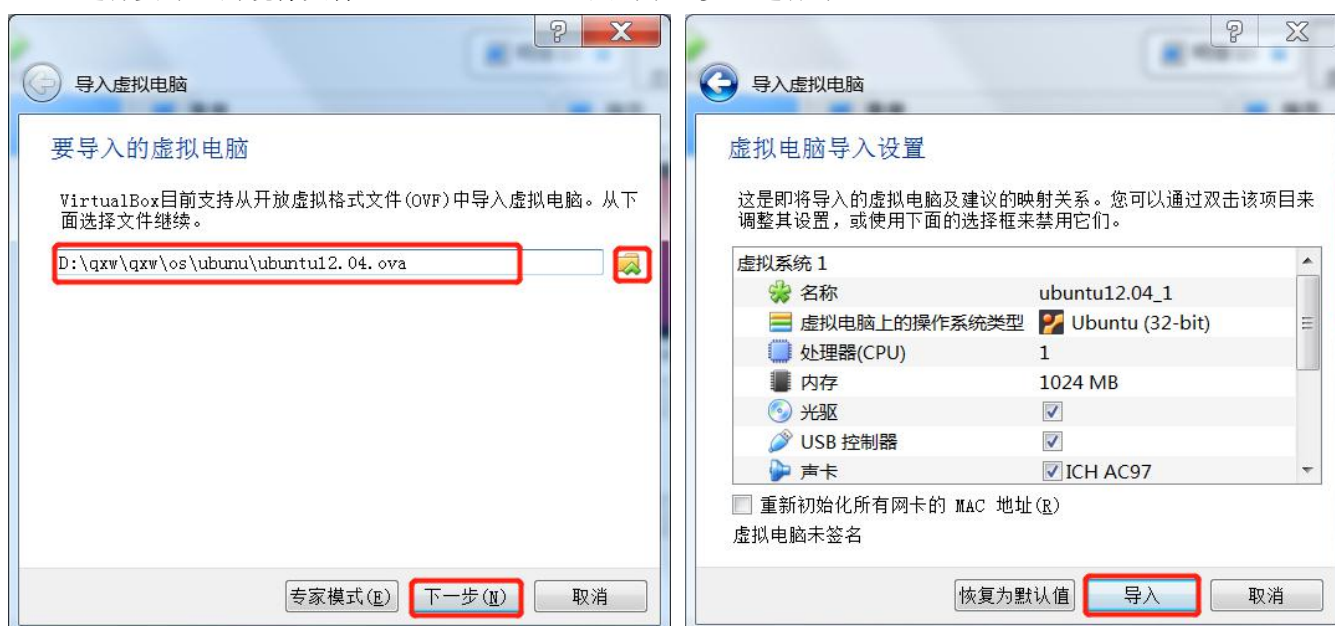


### 2.1.2 虚拟机镜像的导入

(1) 打开 VirtualBox 软件，点击管理选择导入虚拟电脑。



(2) 选择要导入的镜像文件 ubuntu12.04.ova，点击下一步，进行导入。



导入完成后，我们就可以登入已经配置好交叉编译环境的虚拟电脑了，**注意，用户名是 stu，密码是 123456！**

```
Ubuntu 12.04.4 LTS jinxu-VirtualBox tty1
jinxu-VirtualBox login: stu
Password:
Last login: Tue Sep 11 17:26:21 CST 2018 on tty1
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

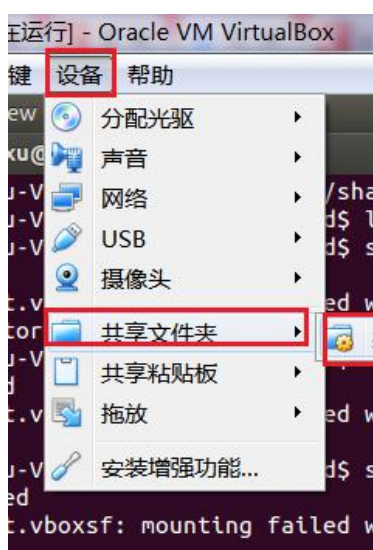
stu@jinxu-VirtualBox:~$ aa
```


导入完虚拟机后，我们需要配置主机和虚拟机的共享文件夹，方便虚拟机和主机的文件传输。

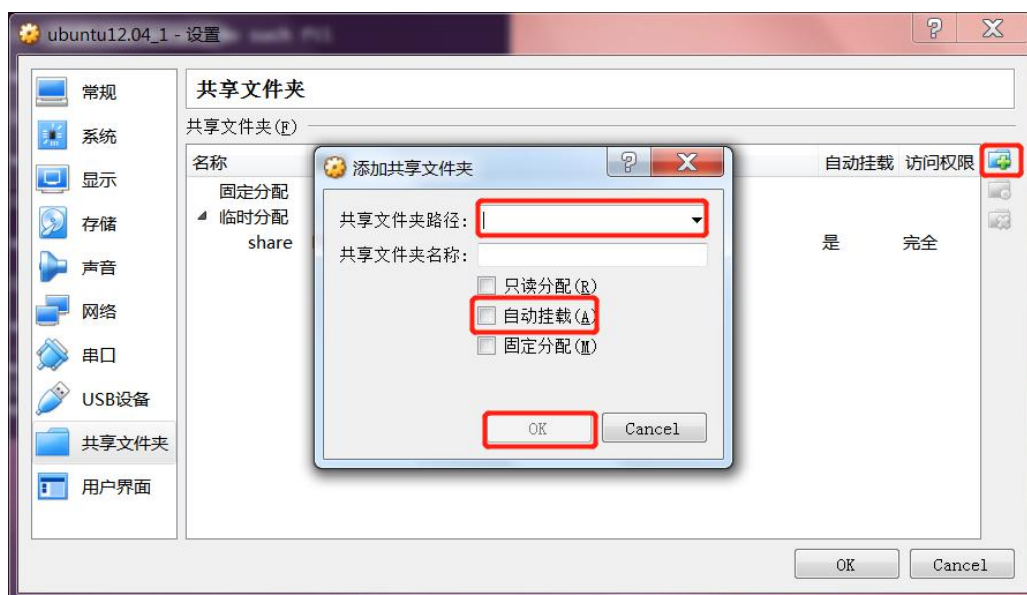
(3) 首先，在主机中找一个位置创建 share 文件夹，如图示例：



点击 VirtualBox 设备->共享文件夹->共享文件夹，如图所示



点击 ，在共享文件夹路径输入创建好的 share 文件夹路径，选择自动挂载，点击 ok。



(4) 在虚拟机 terminal 中输入命令：`sudo mount -t vboxsf share /mnt/shared`。注意，每次虚拟机开机后都需要输入此

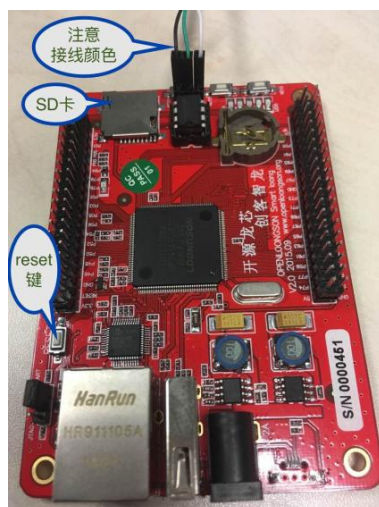


命令! 之后我们就可以在主机中的 share 文件夹中放入需要编译的文件, 在虚拟机 terminal 中输入 `cd /mnt/shared` 就可以进入我们共享的文件夹, 在 terminal 输入 `ls`, 就可以列出共享文件夹中的文件了。

```
minicom: cannot open /dev/ttyUSB0: No such file or directory
stu@stu-VirtualBox:~$ sudo mount -t vboxsf share /mnt/shared
sudo: unable to resolve host stu-VirtualBox
stu@stu-VirtualBox:~$ 
stu@stu-VirtualBox:~$ sudo mount -t vboxsf share /mnt/shared
sudo: unable to resolve host stu-VirtualBox
[sudo] password for stu:
stu@stu-VirtualBox:~$ cd /mnt/shared/finished_code1/
stu@stu-VirtualBox:/mnt/shared/finished_code1$ aaaaa_
```

### 2.1.3 开发板的连接

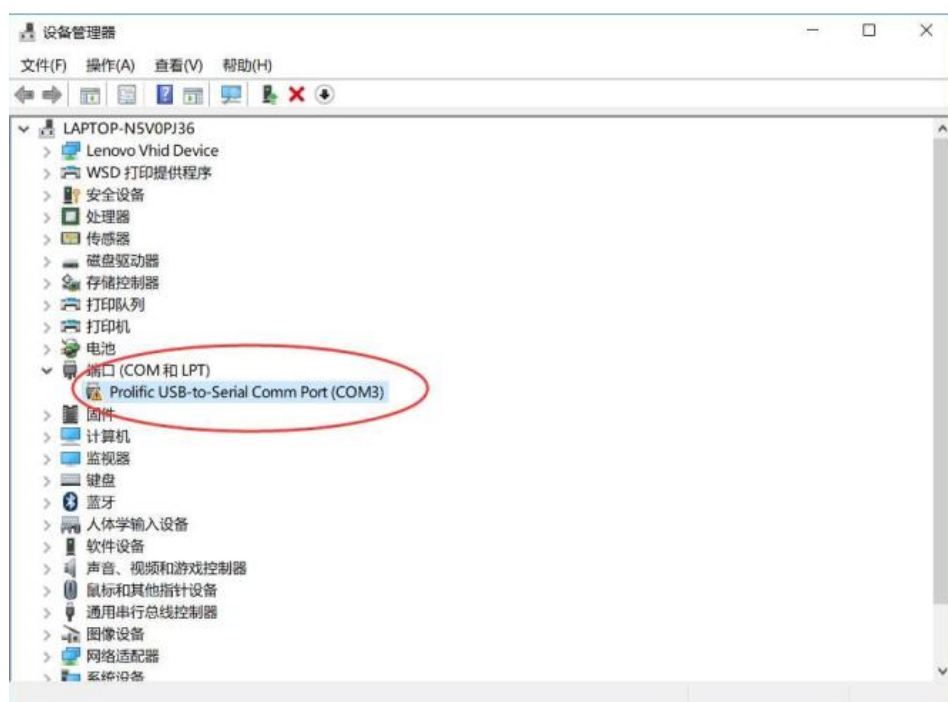
安装开发板串口驱动, 将串口线顺着 SD 卡的方向以黑、绿、白的顺序插入开发板, 同时将串口线 USB 端插入电脑。看到开发板上的两盏 LED 灯亮起说明开发板正常工作。



### 2.1.4 串口配置

Windows10 的驱动安装流程:

(1) 打开 windows 设备管理器, 下图黄色叹号表示设备异常, 驱动未(正确)安装。



(2) 右键选中串口设备，点击更新驱动程序

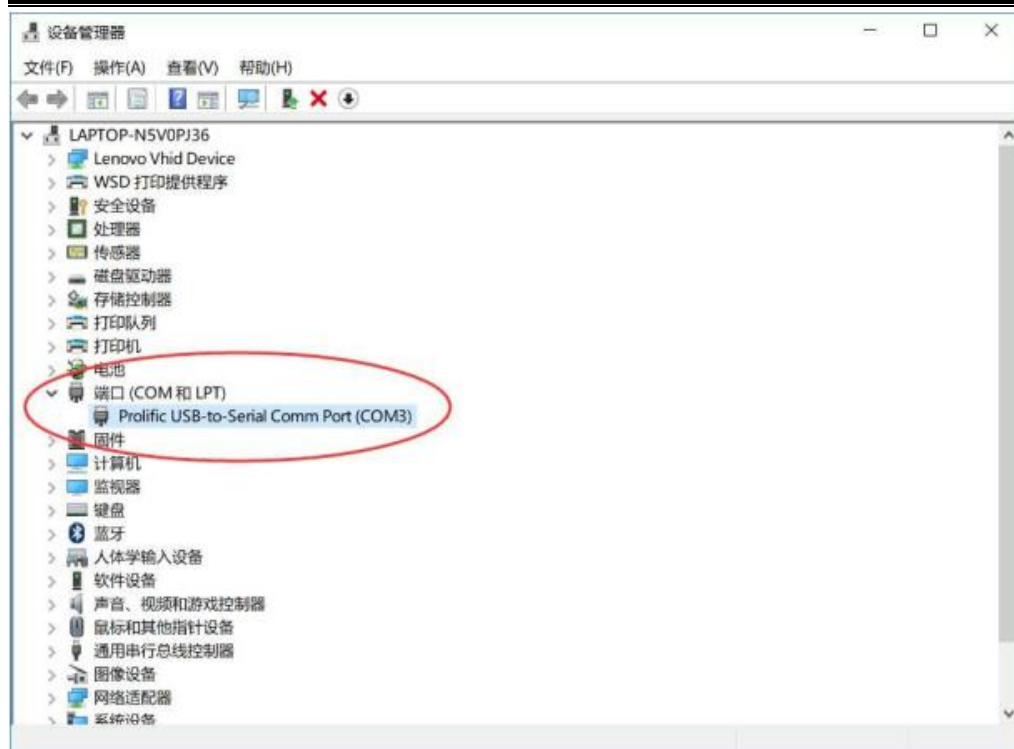


(3) 选择 2009/11/19 的驱动安装。



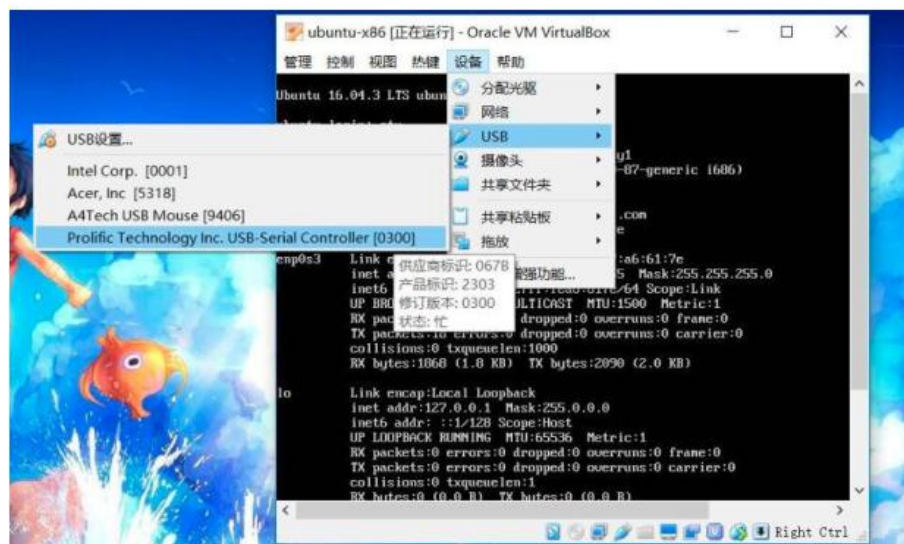
(4) 安装完成之后，开发板正确工作，黄色叹号消失。





Windows7 的安装：

(5) 使用 PL2303\_Prolific\_DriverInstaller\_v110.exe 安装串口驱动。将开发板串口线连接至电脑，同时检查驱动是否正常工作。打开 virtualbox，并打开虚拟机，并登陆。在虚拟机菜单的设备选项-->USB 中选择 USB-Serial Controller。



成功连接之后，串口设备前会出现选中标识。



(6) 在虚拟机的命令行中输入 `sudo minicom -s` 命令便进入了 minicom 的配置界面。

```
+-----[configuration]-----+
| Filenames and paths          |
| File transfer protocols      |
| Serial port setup            |
| Modem and dialing            |
| Screen and keyboard          |
| Save setup as dfl             |
| Save setup as..              |
| Exit                         |
| Exit from Minicom            |
+-----+

```

(7) 进入到 Serial port setup 选项，修改 Serial Device 为 `/dev/ttyUSB0`，完成按回车返回主菜单即可。

```
+-----+
| A - Serial Device      : /dev/ttyUSB0 |
| B - Lockfile Location  : /var/lock    |
| C - Callin Program     :              |
| D - Callout Program    :              |
| E - Bps/Par/Bits       : 115200 8N1   |
| F - Hardware Flow Control : Yes       |
| G - Software Flow Control : No        |
|                           |
| Change which setting?  |
+-----+
| Screen and keyboard    |
| Save setup as dfl      |
| Save setup as..        |
| Exit                   |
| Exit from Minicom      |
+-----+

```

```
+-----[configuration]-----+
| Filenames and paths          |
| File transfer protocols      |
| Serial port setup            |
| Modem and dialing            |
| Screen and keyboard          |
| Save setup as dfl             |
| Save setup as..              |
| Exit                         |
| Exit from Minicom            |
+-----+

```

```
+-----[configuration]-----+
| Filenames and paths          |
| File transfer protocols      |
| Serial port setup            |
| Modem and dialing            |
| Screen and keyboard          |
| Save setup as dfl             |
| Save setup as..              |
| Exit                         |
| Exit from Minicom            |
+-----+

```

(9) 返回主菜单后，选择“Save setup as dfl”，将其保存为默认设置，然后选择 Exit 退出。在命令行中用 root 权限重新打开 minicom: `sudo minicom`，当屏幕出现以下界面代表环境配置成功。

```
Configuration [FCR,EL,NET]
GitHashNumber: bfd91c56aede4ecffd23199fa30e583bbe4f16f9
CommitAuthor: hammer19
CommitDate: Wed Jul 12 15:07:26 2017
userIP:
UserName: zhaoyangyang
MakeTime: Tue Aug 8 11:36:24 CST 2017.
Supported loaders [srec, elf, bin]
Supported filesystems [sdcard, mtd, net, fat, fs, disk, socket, tty, ram]
This software may be redistributed under the BSD copyright.
Copyright 2000-2002, Opsycon AB, Sweden.
Copyright 2005, ICT CAS.
CPU Loongson 1C300A OpenLoongson V2.0 @ 252.00 MHz / Bus @ 126.00 MHz
Memory size 32 MB ( 32 MB Low memory, 0 MB High memory) .
Primary Instruction cache size 16kb (32 line, 4 way)
Primary Data cache size 16kb (32 line, 4 way)
BEV in SR set to zero.
PMON>

```

## 2.2 Linux 实验环境搭建

对于 Linux 的环境配置比较简单，我们不需要使用虚拟机，只需要简单的安装交叉编译环境即可。

### 2.2.1 所需文件

	文件名称	说明	对应网盘目录
1	gcc-4.3-ls232.tar.gz	交叉编译工具	交叉编译工具链

### 2.2.2 串口配置

关于串口的相关驱动已经附加在 Linux 内核里，不需要安装额外的驱动，只需要安装 minicom 串口调试软件即可。

步骤如下：

- 1) 安装 minicom，一般而言 Linux 是自带 minicom 的，如果没有的话，需要我们手动安装一下，直接在终端输入指令，进行安装：**sudo apt-get install minicom**
- 2) 在连接板子之前，我们需要配置一下 minicom 对应的设备文件：**sudo minicom -s**，设置 Port 为 **/dev/ttyUSB0**。

```
+-----+
| A -   Serial Device       : /dev/ttyUSB0
| B -   Lockfile Location   : /var/lock
| C -   Callin Program      :
| D -   Callout Program     :
| E -   Bps/Par/Bits        : 115200 8N1
| F -   Hardware Flow Control : Yes
| G -   Software Flow Control : No
|
| Change which setting?
+-----+
| Screen and keyboard
| Save setup as dfl
| Save setup as..
| Exit
| Exit from Minicom
+-----+
```

- 3) 运行 minicom，连接龙芯开发板，如果开发板亮起 2 盏 LED 灯说明开发板连接正常（一个红色灯代表串口链接正常，一个绿色灯代表电源键供电正常），按下 restart 复位键，不出意外的话，屏幕会打印出初始化信息，并进入 PMON 界面。

```
Configuration [FCR,EL,NET]
GitHashNumber: bfd91c56aede4ecffd23199fa30e583bbe4f16f9
CommitAuthor: hammer19
CommitDate: Wed Jul 12 15:07:26 2017
userIP:
UsrName: zhaoyangyang
MakeTime: Tue Aug 8 11:36:24 CST 2017.
Supported loaders [srec, elf, bin]
Supported filesystems [sdcard, mtd, net, fat, fs, disk, socket, tty, ram]
This software may be redistributed under the BSD copyright.
Copyright 2000-2002, Opsycon AB, Sweden.
Copyright 2005, ICT CAS.
CPU Loongson 1C300A OpenLoongson V2.0 @ 252.00 MHz / Bus @ 126.00 MHz
Memory size 32 MB ( 32 MB Low memory, 0 MB High memory) .
Primary Instruction cache size 16kb (32 line, 4 way)
Primary Data cache size 16kb (32 line, 4 way)

BEV in SR set to zero.
PMON>
```

### 2.2.2 交叉编译环境安装

由于我们编译的程序是基于龙芯开发板的，所以需要进行交叉编译，关于交叉编译需要的工具，我们已经放到了 gcc-4.3-ls232.tar.gz 中，请根据以下步骤搭建环境：

- 1) 解压压缩包到/opt 目录下：**sudo tar zxvf gcc-4.3-ls232.tar.gz -C /opt**
- 2) 添加环境变量：  
打开文件 bashrc: **vi ~/.bashrc**  
在文件末尾添加语句：**export PATH=/opt/gcc-4.3-ls232/bin:\$PATH**
- 3) 重新刷新配置路径：**source ~/.bashrc**

4) 安装依赖库: `sudo apt-get install zlib1g:i386`

进行完以上步骤之后, 运行命令, 如果屏幕输出相关信息, 则说明交叉编译环境配置已经成功。

```
parallels@ubuntu:~$ mipsel-linux-gcc -v
Using built-in specs.
Target: mipsel-linux
Configured with: ../gcc-4.3.0/configure --prefix=/usr --target=mipsel-linux --host=i486-pc-linux-gnu --with-sysroot=/usr --enable-__cxa_atexit --enable-c99 --enable-shared
Thread model: posix
gcc version 4.3.0 (GCC)
```



2.3 Mac 实验环境搭建

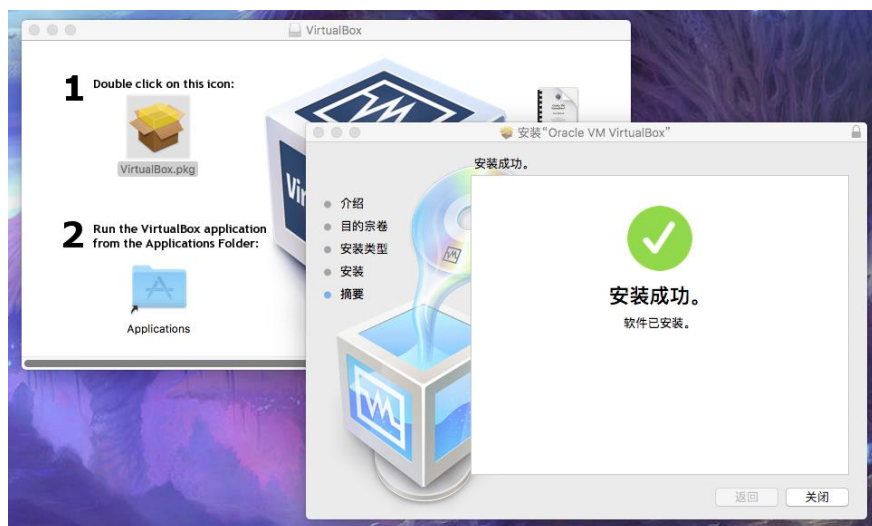
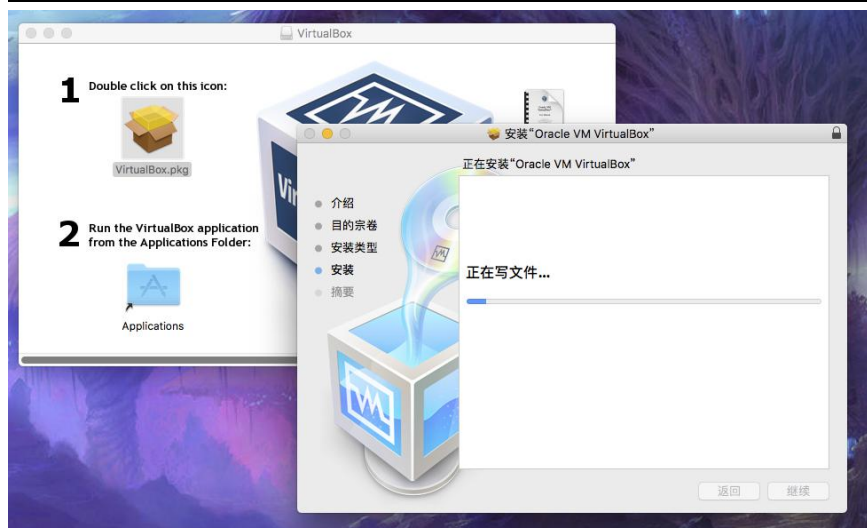
2.3.1 文件列表

	文件名称	说明	对应网盘目录
1	VirtualBox-5.2.18-124319-OSX.dmg	VirtualBox Mac 安装包	Mac_virtualbox
2	PL2303_MacOSX_1_6_1_20170620.zip	Mac 串口驱动	Mac 串口驱动
2	ubuntu_server_12.04.ova	VirtualBox 虚拟机镜像	已有交叉编译环境的镜像

2.3.2 VirtualBox 的安装

VritualBox 的安装在 Mac 上的安装同 windows 一样，只需要使用我们提供的安装包即可。





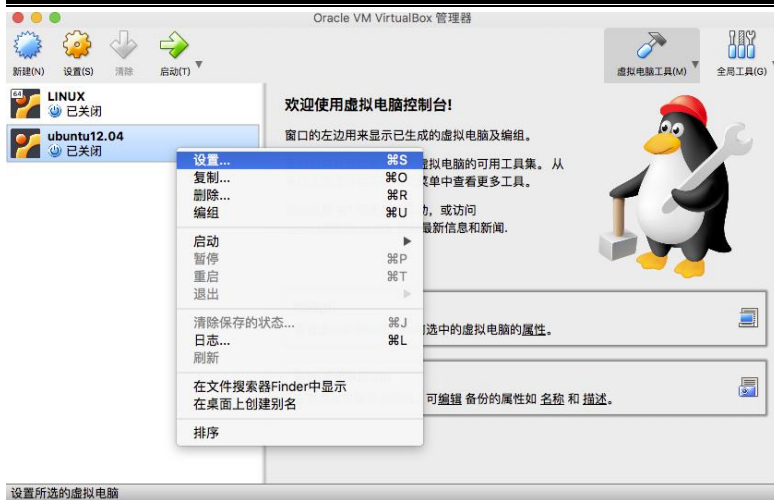
### 2.3.3 虚拟机镜像的导入

使用我们给出的已经配置完成的虚拟机镜像导入 VirtualBox 中。

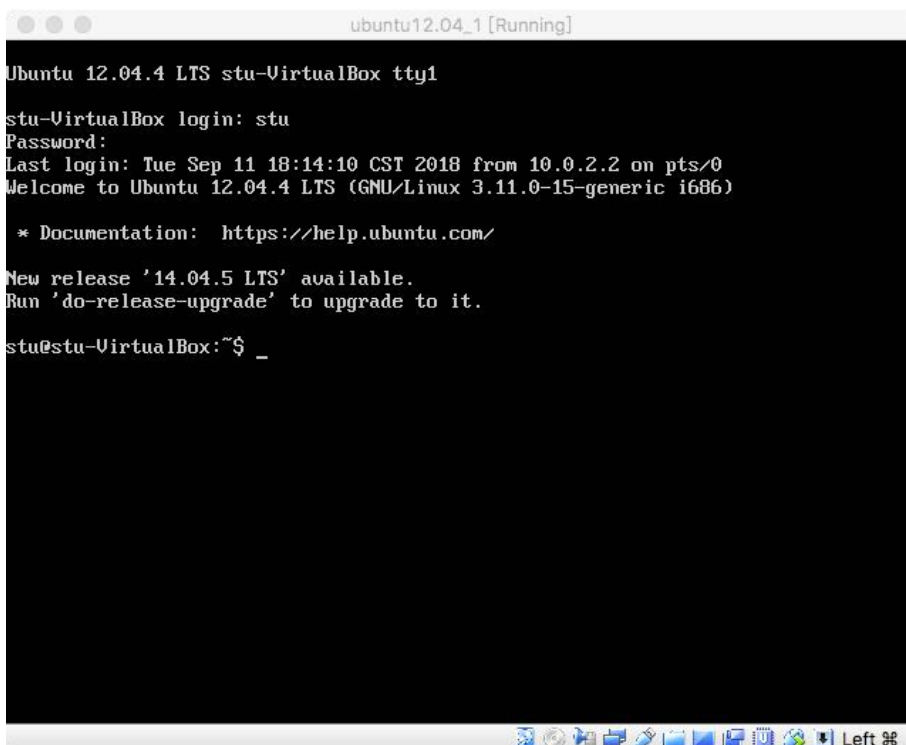






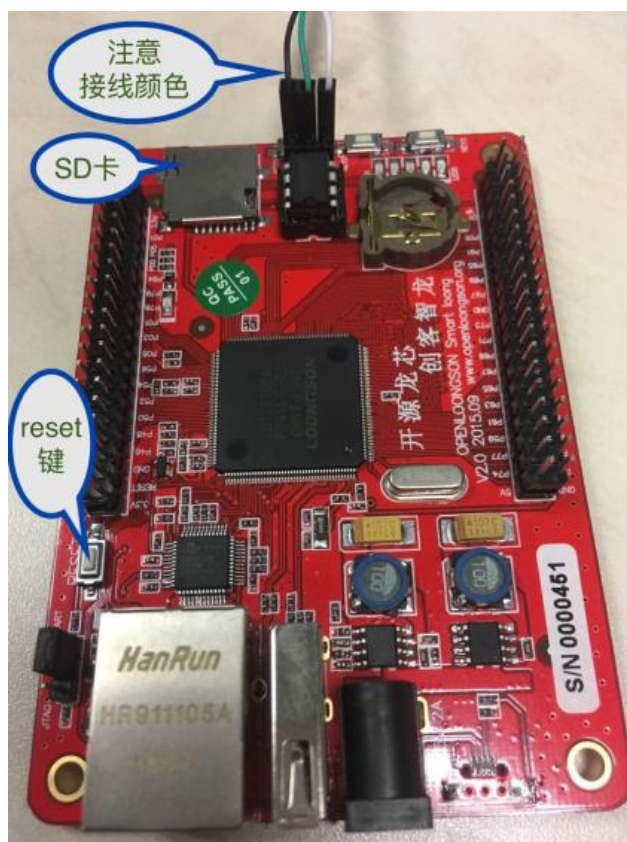


安装完成后登陆，账号为 **stu**，密码为 **123456**，如下图：



### 2.3.4 开发板的连接

安装开发板串口驱动，将串口线顺着 SD 卡的方向以黑、绿、白的顺序插入开发板，同时将串口线 USB 端插入电脑。看到开发板上的两盏 LED 灯亮起说明开发板正常工作。



### 2.3.5 串口驱动安装与配置

同 windows 一样，mac 也需要安装串口驱动，只需要直接双击安装驱动程序即可。安装完需要重启电脑，之后重新连接开发板。

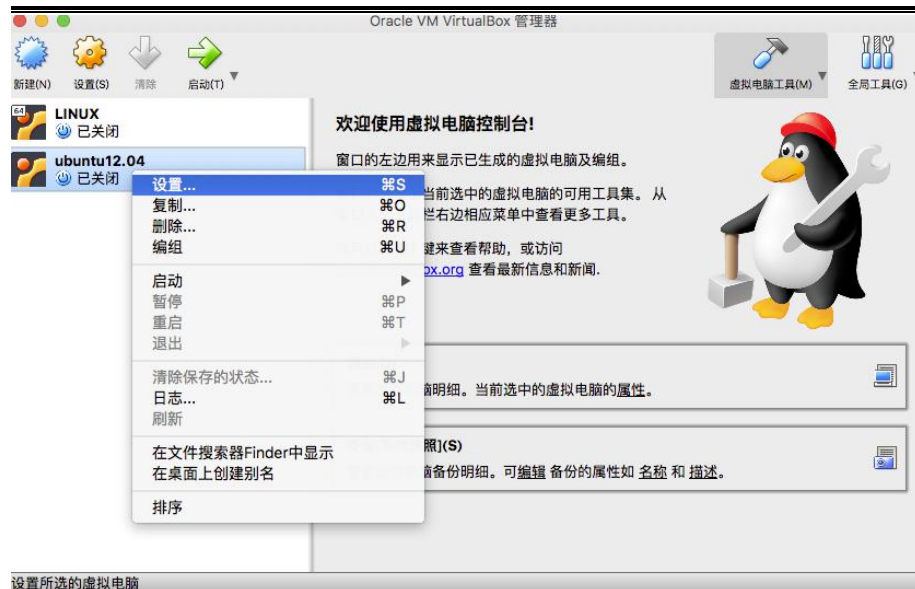




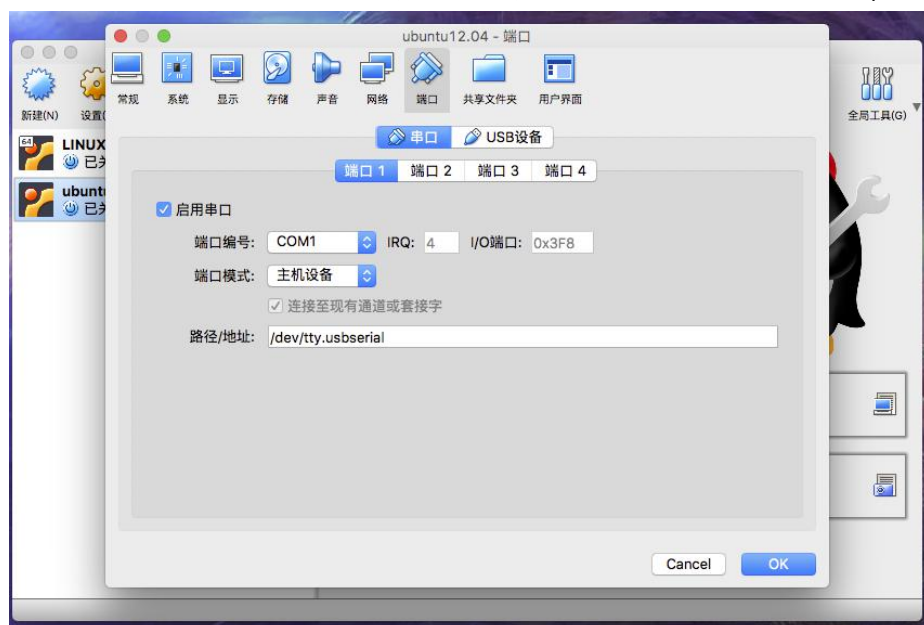
### 2.3.6 虚拟机串口配置

为了让虚拟机可以连接到串口设备，我们需要对虚拟机进行配置。

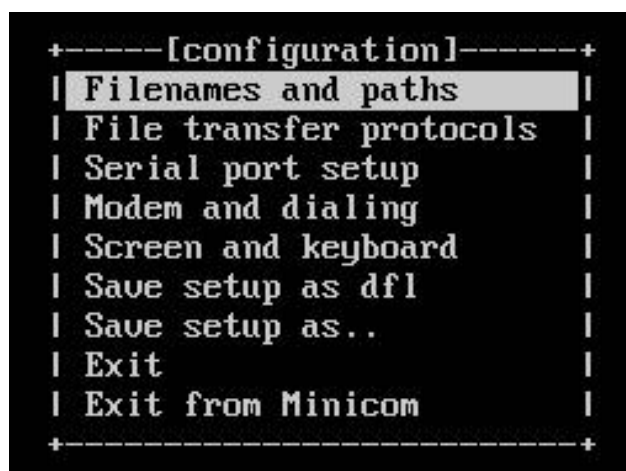
(1) 右键虚拟机，进入设置。



(2) 将串口 1 (COM1) 启用, 并将端口模式设为主机设备, 设置路径/地址为 `/dev/tty.usbserial`。



(3) 配置 minicom 的串口地址, 输入 `sudo minicom -s` 进入到配置界面, 进入 serial port setup 选项, 修改 Serial Device 为 `/dev/ttyS0`。





```

ubuntu12.04_1 [Running]
+-----+
| A -   Serial Device       : /dev/tty8      |
| B - Lockfile Location    : /var/lock       |
| C -   Callin Program     :                 |
| D -   Callout Program    :                 |
| E -   Bps/Par/Bits       : 115200 8N1     |
| F - Hardware Flow Control: Yes            |
| G - Software Flow Control: No            |
|                                     |
|   Change which setting? _             |
+-----+
| Screen and keyboard |
| Save setup as dfl   |
| Save setup as..    |
| Exit                |
| Exit from Minicom   |
+-----+

```

(4) 修改 minicom 设置后，退出 minicom，重新打开 minicom: `sudo minicom`，对开发板进行复位（reset），如果出现以下界面，代表环境配置成功。

```

ubuntu12.04_1 [Running]
bootp=8000bbd8
Version = 0x1237
MacAddr = 0x0 0x55 0x7b 0xb5 0x7d 0xf7
====>enter synopGMAC_mac_init:1000
====>full duplex
====>100M
====>enter synopGMAC_mac_init:1000
====>full duplex
====>100M

Configuration [FCR,EL,NET]
GitHashNumber:
CommitAuthor:
CommitDate:
userIP: 10.0.2.15
UsrName: jinxu
MakeTime: Thu Sep 6 17:13:02 CST 2018.
Supported loaders [srec, elf, bin]
Supported filesystems [sdcard, mtd, net, fat, fs, disk, socket, tty, ram]
This software may be redistributed under the BSD copyright.
Copyright 2000-2002, Opsycon AB, Sweden.
Copyright 2005, ICT CAS.
CPU Loongson 1C300A OpenLoongson V2.0 @ 252.00 MHz / Bus @ 126.00 MHz
Memory size 32 MB ( 32 MB Low memory, 0 MB High memory) .
Primary Instruction cache size 16kb (32 line, 4 way)
Primary Data cache size 16kb (32 line, 4 way)

BEV in SR set to zero.
PMON>
CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.5 | UT102 | Online 00:00

```



附录

2.1 实验环境详细搭建

在第二节我们给出了配置环境的快速搭建，在 windows 和 mac 环境下，大家只需要安装完 VirtualBox 后导入我们已经配置完成的虚拟机镜像即可，但是有些同学已经有了虚拟机或者想自己去一步一步完成环境的配置。因此我们在这一节给出更为详细的环境配置。有需要的同学可以通过这一节的内容去一步一步的配置虚拟机及交叉编译环境。当然，对于已经完成快速搭建且对环境配置不感兴趣的同学可以跳过本节内容。

搭建环境所需工具的百度云链接为：<https://pan.baidu.com/s/1uA7HJI03CQkEk3vk3BVQUg>，提取码：knf1

2.1.1 所需文件

以下的文件大家请根据自己的系统自己选择：

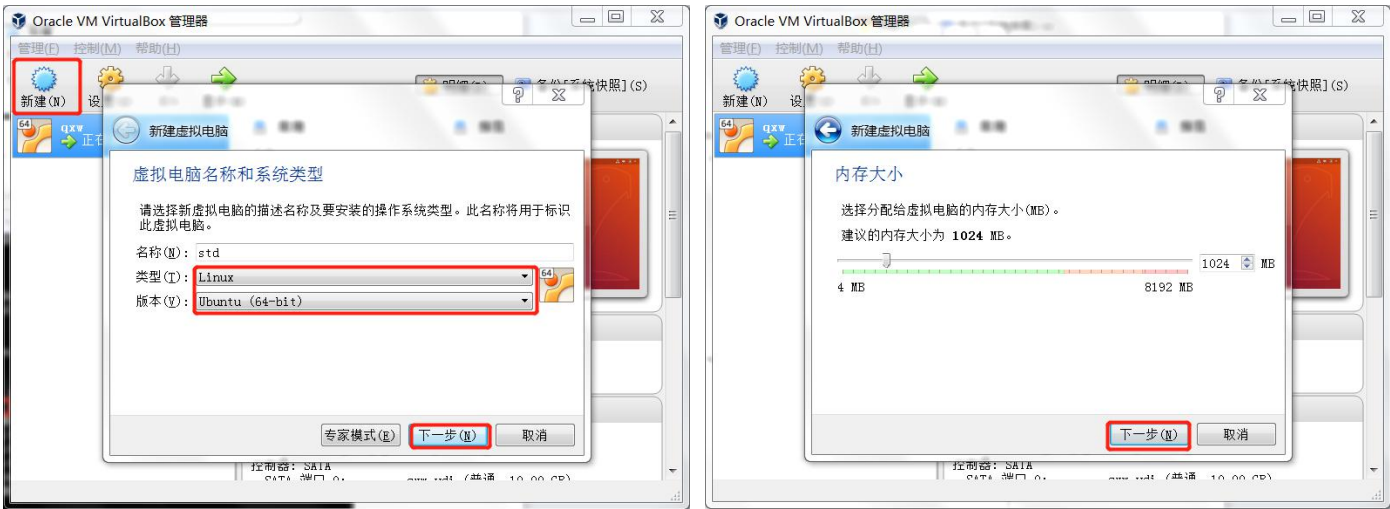
	文件名称	说明
1	VirtualBox-5.2.18-124319-OSX.dmg	VirtualBox Mac 安装包
2	VirtualBox-5.2.18-124319-Win.exe	VirtualBox Windows 安装包
3	Oracle_VM_VirtualBox_Extension_Pack-5.2.18.vbox-extpack	VirtualBox 拓展包
4	gcc-4.3-ls232.tar.gz	交叉编译工具
5	ubuntu-14.04.3-desktop-amd64.iso	Ubuntu 镜像（64bits）
6	PL2303_MacOSX_1_6_1_20170620.zip	Mac 串口驱动
7	PL2303_Prolific_DriverInstaller_v110.exe	Windows7 串口驱动

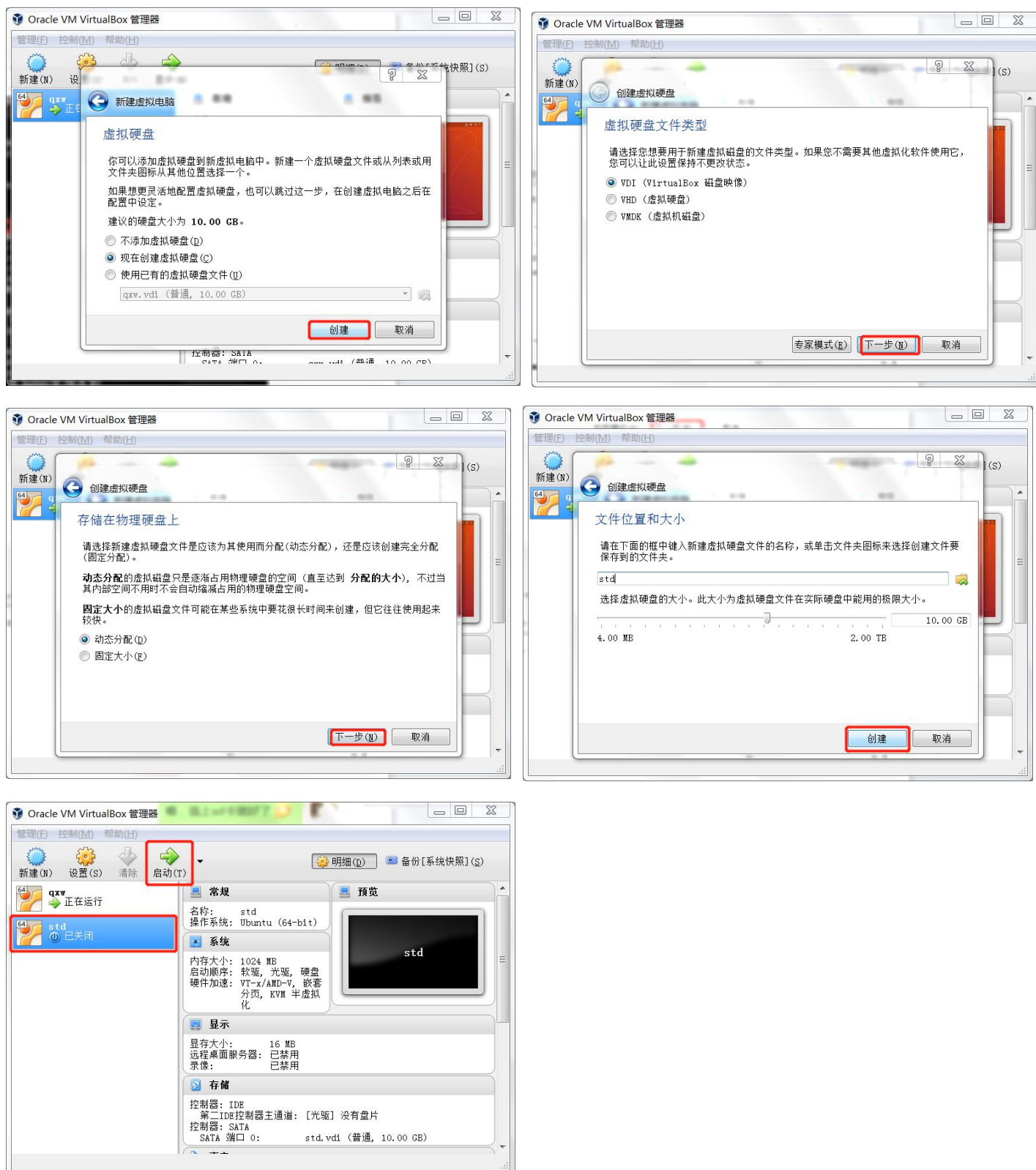
2.1.2 安装 VirtualBox 软件

在之前的小节中我们已经介绍了 windows 和 mac 下 VirtualBox 软件的安装，请大家根据自己的系统进行安装，在这里不进行赘述。

2.1.3 Ubuntu 安装

（1）打开 VirtualBox 程序，新建一个虚拟电脑。选择类型为 Linux，版本为 Ubuntu(64-bit)。如下图所示创建新建一个虚拟电脑。





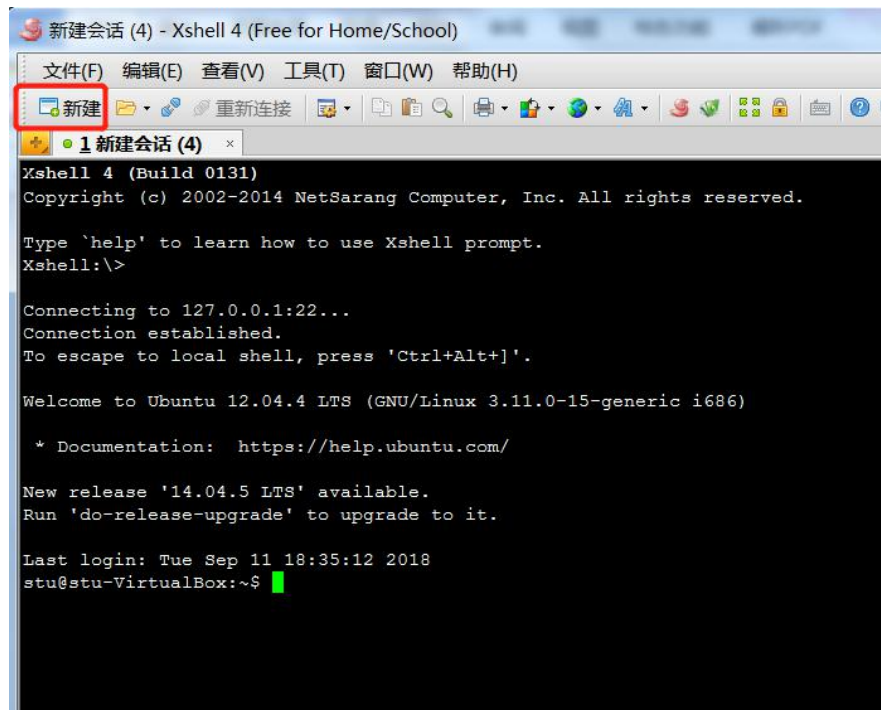
(2) 启动虚拟机，选择启动盘，即下载好的 Ubuntu 系统镜像，点击启动，安装 Ubuntu 系统，设置好用户名和密码。



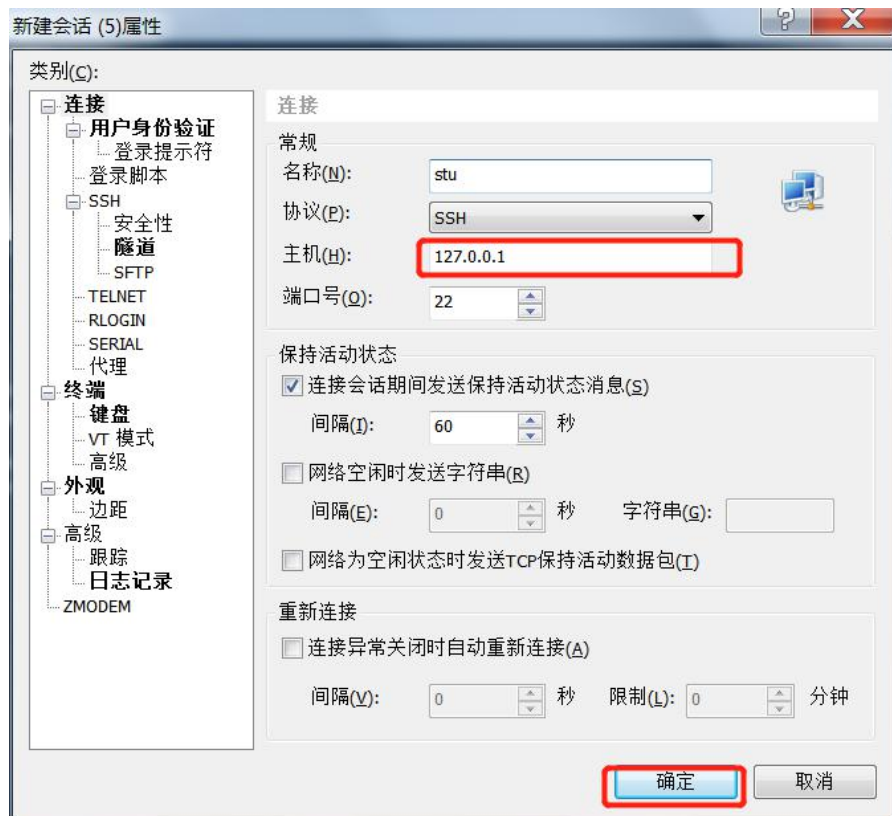
## 2.2 使用 xshell 访问虚拟机

在之前的教程中，我们都是直接在虚拟机中进行操作，但是考虑到有的同学电脑配置问题，可能出现虚拟机太卡的问题，因此我们在打开虚拟机后，可以在主机使用 xshell 工具访问虚拟机。以下为使用 xshell 建立主机和虚拟机的连接过程：

(1) 打开 xshell，点击新建，如图：

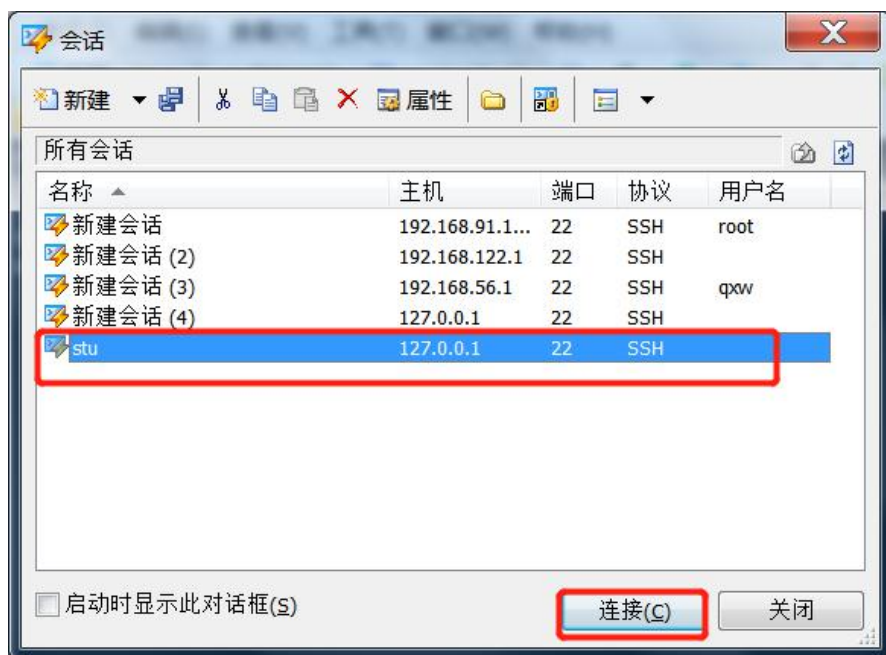


(2) 在弹出的窗口中主机输入 127.0.0.1，端口:22，点击确定。





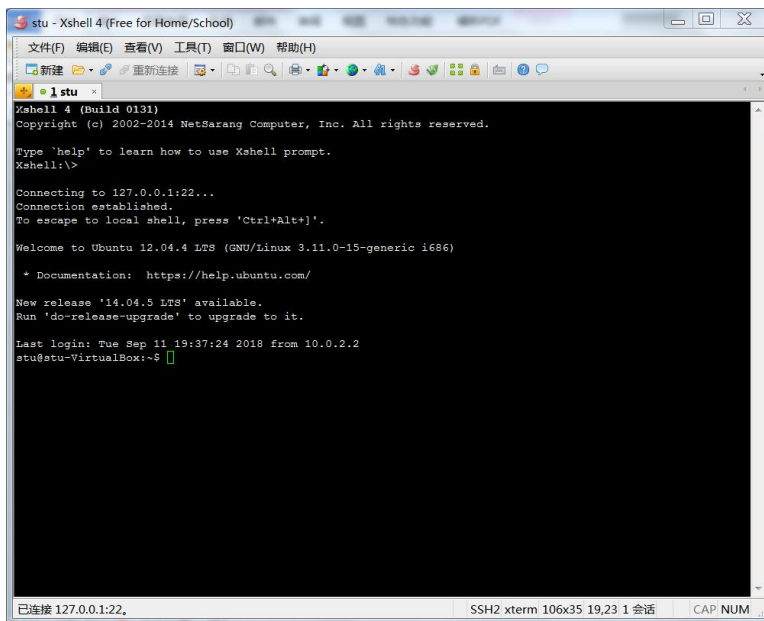
(3) 在弹出的会话框中选择刚建的 stu，点击连接。



(4) 输入用户名：stu，输入密码：123456，点击确定



(5) 完成 xshell 连接虚拟机，如图





## 2.3 MIPS 相关知识

### 2.3.1 MIPS 通用寄存器

在 MIPS 处理器中，算术指令的操作数只能为寄存器，MIPS 中的寄存器都是 32 位操作数。MIPS 提供了 32 个通用寄存器，其用法如下表所示：

寄存器编号	寄存器名	用法
\$0	\$zero	常数 0
\$1	\$at	保留给寄存器处理大常数
\$2~\$3	\$v0~\$v1	存放函数的返回值
\$4~\$7	\$a0~\$a3	存放函数调用参数
\$8~\$15	\$t0~\$t7	临时寄存器
\$16~\$23	\$s0~\$s7	存储寄存器，可存放变量
\$24~\$25	\$t8~\$t9	临时寄存器
\$26~\$27	\$k0~\$k1	用于保存中断信息
\$28	\$gp	全局指针
\$29	\$sp	栈指针
\$30	\$fp	帧指针
\$31	\$ra	子程序调用返回地址

### 2.3.2 MIPS 常用指令

MIPS 汇编指令格式一般如下：

标号：操作码 操作数 1 操作数 2 操作数 3 #注释

操作码表示执行什么操作，操作数表示执行的对象。其中 “标号：” 与 “#注释” 可以不写。

MIPS 常用的汇编指令如下表所示：

指令类型	指令	例子	含义
算术运算指令	加法	add \$s0,\$s1,\$s2	$\$s0 = \$s1 + \$s2$
	减法	sub \$s0,\$s1,\$s2	$\$s0 = \$s1 - \$s2$
	加立即数	addi \$s0,\$s1,10	$\$s0 = \$s1 + 10$
数据存入与读取	读出字	lw \$s0,offset(\$s1)	$\$s0 = \text{memory}[\text{offset} + \$s1]$
	存入字	sw \$s0,offset(\$s1)	$\text{memory}[\text{offset} + \$s1] = \$s0$
	读出半字	lh \$s0,offset(\$s1)	$\$s0 = \text{memory}[\text{offset} + \$s1]$

	读出无符号半字	lhu \$s0,offset(\$s1)	\$s0=memory[offset+\$s1]
	存储字节	sb \$s0,offset(\$s1)	memory[offset+\$s1]= \$s0
	读取字节	lb \$s0,offset(\$s1)	\$s0=memory[offset+\$s1]
	读取立即数到高半字	lui \$s0,30	\$s0=30*2^16
	读取无符号字节	lbu \$s0,offset(\$s1)	\$s0=memory[offset+\$s1]
逻辑运算	与	and \$s0,\$s1,\$s2	\$s0=\$s1&\$s2
	或	or \$s0,\$s1,\$s2	\$s0=\$s1 \$s2
	或非	nor \$s0,\$s1,\$s2	\$s0=~(\$s1 \$s2)
	与立即数	andi \$s0,\$s1,10	\$s0=\$s1&10
	或立即数	ori \$s0,\$s1,10	\$s0=\$s1 10
	逻辑左移	sll \$s0,\$s1,10	\$s0=\$s1<<10
	逻辑右移	srl \$s0,\$s1,10	\$s0=\$s1>>10
条件跳转	相等转移	beq \$s0,\$s1,15	if(\$s0==\$s1) goto PC=PC+4+4*15
	不相等转移	bne \$s0,\$s1,15	if(\$s0!= \$s1) goto PC=PC+4+4*15
	小于设置	slt \$s0,\$s1,\$s2	if(\$s1<\$s2) \$s0=1 else \$s0=0
	低于设置	sltu \$s0,\$s1,\$s2	if(\$s1<\$s2) \$s0=1 else \$s0=0
	小于常数设置	slti \$s0,\$s1,20	if(\$s1<20) \$s0=1 else \$s0=0
	低于常数设置	sltiu \$s0,\$s1,10	if(\$s1<10) \$s0=1 else \$s0=0
无条件跳转	直接跳转	j 2000	goto 2000*4
	间接跳转	jr \$ra	goto \$ra
	跳转并连接	jal 2000	\$ra=PC+4 goto 2000*4

其中有符号与无符号的差别是：有符号运算产生溢出时微处理器会产生异常，无符号运算发生溢出时不会产生异常。

### 2.3.3 CP0 寄存器及指令

在 MIPS 体系结构中，最多支持 4 个协处理器(Co-Processor)。其中，协处理器 CP0 是体系结构中必须实现的。它起到控制 CPU 的作用。MMU、异常处理、乘除法等功能，都依赖于协处理器 CP0 来实现。它是 MIPS 的精髓之一，也是打开 MIPS 特权级模式的大门。

CP0 寄存器用于控制处理器的状态改变并报告处理器的当前状态。这些寄存器通过 MFC0 指令来读或 MTC0 指令来写。对 CP0 的主要操作有以下的指令：

mfc0 rt, rd 将 CP0 中的 rd 寄存器内容传输到 rt 通用寄存器；

mtc0 rt, rd 将 rt 通用寄存器中内容传输到 CP0 中寄存器 rd；

### 2.3.4 伪指令

如下表所示，列出了 MIPS 常用的伪指令。

伪指令	含义
.rdata	只读数据
.align 2	两字节对齐
.globl status	全局变量 status
.set	设置代码的属性
.type @function	定义的是函数，这里的话可以看出是数据还是函数
.set noreorder	不让代码优化
.set reorder	优化代码
.extern g_data	对外部变量的引用
main:	标号 main
.set push	将当前的属性保存起来，是的设置的属性仅作用于当前的代码
.set pop	回复之前保存的属性
.set noat	不使用\$at 寄存器，当用到\$at 寄存器时不会报警
.set at	使用\$at 寄存器，当用到\$at 寄存器时会报警,但是不会报错
.text	代码段
.data	数据段
.ent	用于标记函数的起点
.end	用于标记函数结束的位置
.set macro A	定义宏 A
.endm	结束宏定义
.ascii "HELLO\0"	定义字符串不附带空结束符

.asciz "HELLO"	定义字符串附带空结束符
----------------	-------------

### 2.3.5 MIPS 程序的一个例子

我们给大家提供一个 MIPS 程序的例子。例子如下：

```

22  .globl asm_start           #定义全局变量asm_start
23  .align 2                  #2字节对齐
24  .type asm_start, @function #定义的asm_start是函数
25  .ent asm_start, 0         #开始asm_start
26  asm_start:                #asm_start标号
27  .frame sp, 0, ra
28      mtc0    $0, CP0_STATUS #将$0通用寄存器中内容传输到CP0中寄存器CP0_STATUS中
29      mtc0    $0, CP0_WATCHLO #将$0通用寄存器中内容传输到CP0中寄存器CP0_WATCHLO中
30      mtc0    $0, CP0_WATCHHI #将$0通用寄存器中内容传输到CP0中寄存器CP0_WATCHHI中
31
32      mfc0    t0, CP0_CONFIG #将CP0中的CP0_CONFIG寄存器内容传输到t0通用寄存器
33      and     t0, ~0x7        #将t0通用寄存器中内容与上~0x7, 并将结果存入t0中
34      ori     t0, 0x2         #将t0通用寄存器中内容或上0x2, 并将结果存入t0中
35      mtc0    t0, CP0_CONFIG #将t0通用寄存器中内容传输到CP0中寄存器CP0_CONFIG中
36
37      jr      ra              #跳转到ra寄存器指向的位置
38  .end asm_start
39  .size asm_start, .- asm_start
40

```

## 2.4 龙芯 QEMU 模拟器使用介绍

### 2.4.1 什么是 QEMU

#### 2.4.1.1 QEMU 模拟器简介

QEMU 是一个通过软件来模拟机器架构的模拟器。在这样的一个模拟器中，一个软件变量可能就来模拟一个硬件上的寄存器，而硬件上的各种运算也会通过软件运算来模拟。本课程提供给同学们的 QEMUloongson 就是用来模拟龙芯开发板的一个模拟器，功能与龙芯基本相同，包括 CPU、接口、网络、内存地址空间等。同学们可以将自己写的操作系统运行在 QEMU 上，方便调试自己的代码。

使用模拟器进行调试的优点是：QEMUloongson 可以接入 GDB，通过 GDB 来单步调试、设置断点、输出系统寄存器内容、输出地址空间内容等等，方便大家定位 bug 和判断 bug 原因。

#### 2.4.1.1 学习使用模拟器

学习使用 QEMU 和 GDB 会花一点时间，但是会大大降低同学们解决代码 bug 的难度。由于 QEMU 上的功能与输出基本和开发板相同，所以同学们可以在 QEMU 上调试代码，通过之后再上开发板调试。这样不仅可以减少反复 sd 卡拷贝的过程，也可以利用模拟器与 gdb 的连接，更方便的调试代码中的 bug。

学习使用模拟器对于同学们未来的研究生涯也会有帮助。由于硬件平台无法修改，而且调试复杂，所以使用软件模拟器来进行硬件和结构方面的研究会是一种常用的手段。在本课程中了解模拟器的概念和用法，对同学们未来的研究也会有一定帮助。同时在模拟器与 gdb 的配合使用中，同学们也可以熟悉 Linux 的代码调试利器：gdb 的使用。

但是必须要提示同学们的是，模拟器只能帮助代码的调试，而各个 project 的设计必须由同学们自己仔细思考，一个良好的代码设计才是又快又好完成代码的最重要基础。

### 2.4.2 需要的工具

在课程网站上我们提供的 QEMUloongson.rar 中，包含了 QEMU 的运行程序（bin 文件夹下的程序）和运行脚本（run\_pmon.sh），以及和开发板上相同功能的 PMON（bios 文件夹下的 gzram.bin）。此外同学们可以自己安装 gdb-multiarch，由于龙芯平台使用 MIPS 架构，所以一般的 gdb 并不能支持，需要使用支持多架构的 gdb-multiarch。如果同学们在 P0 中使用的是我们提供的带交叉编译环境的 ubuntu 镜像，那么其中已经安装好了 gdb-multiarch。如果同学的环境下没有安装，可以通过输入 `apt-get install gdb-multiarch` 命令安装。如果同学们使用的 Linux 是 CentOS 或 Fedora 等，可以使用 QEMUloongson.rar 压缩包中包含的 mipsel-linux-gdb 来实现相同功能。

### 2.4.3 使用方法

在这一节中，我们将介绍 QEMU 模拟器的使用方法。

#### 2.4.3.1 制作 USB 镜像盘

类似于开发板，我们需要一个 SD 卡来存储同学们自己编写的操作系统代码，并由 PMON 读进内存。但是由于 QEMUloongson 模拟器的功能限制，我们这里使用一个虚拟的 USB 盘来代替 SD 卡。

首先，我们需要制作一个大一点的空磁盘镜像，输入指令

```
dd if=/dev/zero of=disk bs=512 count=1M
```

这里之所以要先制作一个 512MB 容量的空盘，是为了防止同学们直接用代码来制作磁盘镜像，使得磁盘镜像过小，一旦读取超过磁盘大小的偏移位置就会报错。

接着我们将同学们的代码写入磁盘，类似于 P1 中介绍的 make floppy 操作，即输入以下命令：

```
./createimage -extended bootblock kernel
```

```
dd if=image of=disk conv=notrunc
```

注意第二条命令的 if 是 createimage 出来的 image 文件，of 后面是上一步中建好的空磁盘镜像，请同学们自己使用的时候注意文件路径。

```
stu@stu-VirtualBox:~/QEMUloongson$ dd if=image of=disk conv=notrunc
2+0 records in
2+0 records out
1024 bytes (1.0 kB) copied, 0.000607305 s, 1.7 MB/s
stu@stu-VirtualBox:~/QEMUloongson$
```

输出入上图所示。至此，我们就成功的把自己编写的操作系统代码写入了一个供 QEMU 使用的 USB 盘镜像。

#### 2.4.3.2 启动 QEMU 模拟器

接下来，使用我们提供的 run\_pmon.sh 脚本，同学们就可以启动 QEMU 模拟器并搭载上上一小节中建好的 USB 盘镜像了。

run\_pmon.sh 中的内容如下（前面的一些开头带有#的脚本不会生效，只是备用）：

```
SERIAL=2 ./qemu/bin/qemu-system-mipsel -M ls1c -vnc 127.0.0.1:0 -kernel ./bios/gzram -gdb
tcp::50010 -m 32 -usb -drive file=disk,id=a,if=none -device usb-storage,bus=usb-bus.1,driv
e=a -serial stdio "$@"
```

其中 -kernel 后面的文件是我们提供的 pmon 文件，-drive file=后面的文件是 USB 镜像，请同学们注意这两个参数的文件路径是否正确。

检查无误后，就可以启动 run\_pmon.sh 脚本了，但是在此之前，同学们需要注意 QEMU 执行程序是否有可执行权限。需要输入以下命令：

```
chmod +x qemu/bin/qemu-system-mipsel
```

然后，我们就可以通过以下命令启动运行脚本了。

```
sh run_pmon.sh
```

经过一段前面的输出之后，同学们应该就看到和开发板上相同的 PMON 命令提示符了，如下：

```
CPU Loongson 1C300A QEMUloongson V2.0 @ 240.00 MHz / Bus @ 120.00 MHz
Memory size 32 MB ( 32 MB Low memory, 0 MB High memory) .
Primary Instruction cache size 16kb (32 line, 4 way)
Primary Data cache size 16kb (32 line, 4 way)

BEV in SR set to zero.
PMON>
```

至此，QEMU 模拟器的启动已经完成，同学们可以在熟悉的 PMON 命令符下输入 loadboot 命令，看看和在开发板上是否完全一样：)

#### 2.4.3.3 连接 gdb 工具

通过前面的步骤，我们已经可以看出，QEMU 模拟器的功能和开发板上基本一样。但是只做到这一点并不能让同学们更好的调试自己的代码，所以连接 gdb 是必要的。

在 QEMU 已经启动的情况下，在另一个窗口中输入 gdb-multiarch 命令，看到以下输出即进入 gdb 程序：

```
stu@stu-VirtualBox:~$ gdb-multiarch
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>.
(gdb)
```



然后在 gdb 命令提示符后，我们就可以使用一些 gdb 的命令了。首先我们先来连接正在运行的 QEMU 模拟器。输入下面的命令分别设置架构名称和远程端口：

```
set arch mips
```

```
target remote localhost:50010
```

可以看到下面的输出：

```
(gdb) set arch mips
The target architecture is assumed to be mips
(gdb) target remote localhost:50010
Remote debugging using localhost:50010
0x8004503c in ?? ()
(gdb) █
```

这样就已经成功的连接上 QEMU 模拟器了。要注意的是，使用 target remote 连接成功之后，QEMU 模拟器会进入暂停执行的状态，我们可以在 gdb 命令提示符后面输入 c 命令，QEMU 模拟器就可以继续运行，输出下图所示：

```
(gdb) c
Continuing.
```

后面可能会继续打印一些 warning，但是同学们切回 QEMU 的界面，应该可以发现 QEMU 模拟器已经可以继续运行，并接收输入了。

#### 2.4.3.4 gdb 调试常用命令

通过上一步的操作，同学们已经将 gdb 和 QEMU 模拟器连接了起来，可以使用 gdb 来对 QEMU 模拟器进行调试了。这里介绍一些调试中常用的命令和它们的效果，同学们也可以继续钻研和讨论更多的用法。

(1) 设置断点命令：b。范例如下：

```
(gdb) b *0xa0800000
warning: GDB can't find the start of the function at 0x8004503c.
Breakpoint 1 at 0xa0800000
(gdb) c
Continuing.

Breakpoint 1, 0xa0800000 in ?? ()
(gdb) █
```

注意 b 命令后面参数中的地址，前面要带一个\*号。

设置断点成功之后就会显示 Breakpoint 1 at XXXX，这里的 1 是断点的编号，再设置一个断点就会变成 2 号。

使用 c 继续执行之后，QEMU 模拟器就会在运行到断点位置是停下，停下之后，同学们就可以使用后面介绍的一些指令来查看当前的状态了。

(2) 单步运行：si。范例如下：

```
(gdb) si
warning: GDB can't find the start of the function at 0xa0800000.
warning: GDB can't find the start of the function at 0xa0800004.
0xa0800004 in ?? ()
(gdb)
warning: GDB can't find the start of the function at 0xa0800008.
0xa0800008 in ?? ()
(gdb)
```

注意：范例中后面的几个 gdb 命令提示符后面没有输入 si，是因为我在这里使用敲了回车。在 gdb 中，直接敲回车键相当于重复之前的命令。

在范例中我们可以看到，代码从 0xa080000c 之后跳到了 0x8007b980，通过单步执行我们有效的跟踪了代码执行的跳转过程。

(3) 查看寄存器内容：i r。范例如下：

```
(gdb) i r
      zero      at      v0      v1      a0      a1      a2      a3
R0    00000000 ffffffff 8000bb38 a0800000 a0800080 801ffd28 8000b3d8 8000b3cc
      t0        t1        t2        t3        t4        t5        t6        t7
R8    8007b980 00000001 00000000 800c0000 800c0000 800b3eb4 8008b694 ffffffff
      s0        s1        s2        s3        s4        s5        s6        s7
R16   00000200 0000000f 00000200 8000b938 80090f18 8009a7d4 8000bb68 00000000
      t8        t9        k0        k1        gp        sp        s8        ra
R24   00000170 8000b430 00000000 00000000 80850000 8000b928 00000001 a0800014
      sr        lo        hi        bad      cause     pc
      00000000 00000000 00000000 00000000 40008000 8007b980
      fsr       fir
      00000000 00739300
(gdb)
```

通过这样的命令，我们可以输出所有寄存器的内容。

也可以在 i r 后面空格再输入寄存器名，则只输出指定寄存器的内容。

(4) 查看内存内容：x，命令格式为 x/nfu [addr] (n 是内存单元个数，f 是显示格式，u 是内存单元大小)，

范例如下：(gdb) x/10i 0xa0800000

```
0xa0800000: lui      t0,0xa080
0xa0800004: lw       t0,156(t0)
0xa0800008: lui      a0,0xa080
0xa080000c: jalr     t0
0xa0800010: addiu    a0,a0,128
0xa0800014: nop
0xa0800018: lui      t0,0xa080
0xa080001c: lw       t0,152(t0)
0xa0800020: lui      a0,0xa080
0xa0800024: lw       a0,164(a0)
(gdb) x/20x 0xa0800000
0xa0800000: 0x3c08a080      0x8d08009c      0x3c04a080      0x0100f809
0xa0800010: 0x24840080      0x00000000      0x3c08a080      0x8d080098
0xa0800020: 0x3c04a080      0x8c8400a4      0x24050200      0x3c060001
0xa0800030: 0x0100f809      0x34c64000      0x00000000      0x3c08a080
0xa0800040: 0x8d0800a8      0x0100f809      0x00000000      0x03e00008
(gdb)
```

通过改变 x 命令后面的参数，可以灵活的查看内存中的内容，比如范例中的 10 和 20 就是查看的指定地址之后的内存长度，i 和 x 则代表将内存内容以汇编指令的格式或者十六进制数据的格式打印出来。

(5) 退出 gdb：q

## 2.5 Linux 的使用

### 2.5.1 Linux 介绍

Linux 是一套免费使用和自由传播的操作系统，是一个基于 POSIX 和 Unix 的多用户、多任务、支持多线程和多 CPU 的操作系统。Linux 系统性能稳定，而且是开源软件，它与其他操作系统相比，具有源码开放、没有版权、社区用户多等特点。源码开放这一特点使得用户可以自由裁剪，在科研工作中具有重要作用。

在本课程中，我们使用 Linux 系统对代码进行编辑、编译，并用来连接开发板。通过学习熟悉 Linux 的基本使用方法，可以更高效的完成本课程的任务。

### 2.5.2 文件夹操作

本节将列出主要的 Linux 系统文件夹命令。

命令名称	命令作用
ls	显示当前目录下的所有文件夹和文件
mkdir	建立一个新的文件夹
cd	进入某一层文件夹
pwd	打印当前文件夹路径

**绝对路径和相对路径：**文件夹命令的操作对象为指定路径的文件夹。在 Linux 系统中，有绝对路径和相对路径两种指定路径的方式。简单的来说，以/开头的路径会被 Linux 系统识别为绝对路径，否则为相对路径。因此，绝对路径指的是从根目录/开始的路径，而相对路径则代表**当前路径**下的路径。（使用 pwd 可以打印**当前路径**）

需要注意的是，Linux 中有几个符号表示着特殊的路径。

路径符号	路径含义
~	家路径，在本课程提供的镜像中，代表/home/stu
.	当前路径
..	上一级路径

命令使用举例：

命令内容	命令效果
cd ..	返回上一级
ls /home/stu	打印/home/stu 路径下的所有文件夹和文件
mkdir stu	在当前路径下创建名为 stu 的文件夹

### 2.5.3 文件命令

本节将列出主要的 Linux 系统文件夹命令。

命令名称	命令作用
cat	查看一个文件的内容
less	支持上下滚动行的查看文件内容的命令
grep	查找文件内容
chmod	改变数据权限
cp	copy, 拷贝
rm	remove, 删除

**-r 递归参数：**Linux 系统的很多命令都支持-r 参数，例如 cp 命令，cp test.c /home/stu 是将 test.c 文件拷贝到路径

/home/stu, `cp -r ./ /home/stu` 是将本文件夹全部拷贝至路径/home/stu。

**管道符**：管道符用于将管道符前面一个命令的输出作为后面一个命令的输入。例如 `ls /home | grep 'stu'` 的效果相当于在/home 路径下的文件夹和文件名称中查找是否有匹配 stu 字符串的内容。

命令使用举例：

命令内容	命令作用
<code>less test.c</code>	显示 test.c 文件的内容
<code>rm -r stu</code>	删除 stu 文件夹
<code>grep 'hello' test.c</code>	在 test.c 中查找匹配字符串 hello 的内容
<code>chmod +x test.sh</code>	给 test.sh 文件赋予执行权限

### 2.5.4 vim 操作

vim 是一个功能强大的文本编辑器，本课程中，推荐在 Linux 系统中使用 vim 来编辑文件。vim 也可以用来直接创建新文件，只要后面的文件名并不存在，就会创建一个新的文件。例如 `vim test.c`。

刚刚启动的 vim 程序处于**命令模式**，输入的字符都被 vim 视为命令而非编辑的文字。使用 `i,a,s` 可以进入**编辑模式**。在编辑模式中，任何字符的输入都会被认为普通输入的文字，除非使用 `esc` 键退出编辑模式回到命令模式。

在命令模式下，可以使用冒号键使屏幕下方出现一个冒号，在此时输入的任何字符都会出现在冒号后面。输入 `w` 然后回车可以保存当前编辑的文件，输入 `q` 然后回车可以退出 vim。类似的有 `q!` 和 `wq` 命令，分别是不保存退出和保存且退出。

在命令模式下还可以使用复制粘贴功能。分别是 `d`、`y`、`p` 键。

粘贴：`p`；小写 `p` 粘贴到当前游标之后，大写 `P` 粘贴到当前游标之前

剪切：`d`；`dd` 剪切当前行，`d+数字+上下方向键` 剪切当前行和之前之后的 `n` 行

复制：`y`；`yy` 复制当前行，`y+数字+上下方向键` 复制当前行和之前之后的 `n` 行

在命令模式下按 `/` 键会进入查找模式，通过输入要查找的字符或字符串可以查找，通过 `n` 或 `N` 查找下一个或上一个。

## 2.6 MAKEFILE 的使用

makefile 的规则如下:

target ... : prerequisites ...

command

其中, **target** 也就是一个目标文件, 可以是 **Object File**, 也可以是执行文件。还可以是一个标签(**Label**)。**prerequisites** 是要生成那个 **target** 所需要的文件或是目标。

**command** 也就是 **make** 需要执行的命令 (任意的 **Shell** 命令)。**command** 前面是一个 **tab** 键!

一个简单的例子如下:

```
main: main.o add.o sub.o
main.o: main.c
    gcc -c main.c -o main.o
add.o: add.c
    gcc -c add.c -o add.o
sub.o: sub.c
    gcc -c sub.c -o sub.o

.PHONY: clean
clean:
    -rm -rf *.o
```