课前课程管理——学生篇

训练课程与自学的环境不太一样，训练课程的 Linux 操作系统环境中，主要是由笔者预先帮大家规划好的环境来让大家做许多的练习，到第 15 章的时候，再让大家自己搭建一个环境，期待在期末的时候大家都具有自己建立好的属于自己的 Linux 操作系统的环境。不过，在训练过程中， 还是期望大家先预先安装好的虚拟化系统，这样在本书的学习时比较方便。 因此，如果老师/同学们没有办法获取商用的虚拟机环境，可以使用下面介绍的 VirtualBox 虚拟化软件来搭建你的 Linux 虚拟机在线环境。

0.1 前言

训练课程的主要内容大致上是通过商业的虚拟机软件来实现的，该软件在教学与实践操作上面比较方便。不过考虑到许多大专院校或者个人学习有经费不足的问题，要建立这样的环境确实有点难度。因此，笔者也使用 VirtualBox 的环境来让大家建立训练环境。只是这个训练环境与实际操作有些不同，其中硬盘的文件名是 /dev/sda 而不是虚拟加速的 /dev/vda，因此在许多的检测脚本上，结果会有点小问题，这部分请大家多多见谅。

笔者这里假设大家都已经具有使用 Windows 操作系统的经验，因此下面主要是以 Windows 操作系统的基础环境来说明如何建立一个虚拟化的训练系统。如果你已经具有使用 Linux 虚拟化的经验，请直接使用 Linux 中的 qemu-img 命令，将本书提供的下载文件中操作系统镜像文件（img）直接转成 qcow2 后，这样就可以通过 Linux 的 KVM 搭配 libvirt 与 qemu 软件，直接挂载该硬盘就能开始操作。如果你只有 Windows 或者是 MAC 操作系统，就使用下面介绍的 VirtualBox 来建立我们的上课环境。

如果您是计算中心的管理人员，而且需要替老师们建立好这个 Linux 操作系统的教学环境时，不需要进行多重操作系统的规划，只要将 VirtualBox 安装妥当，并且将硬盘的镜像文件 img 放置好即可。同时根据下面的流程建立好整个 VirtualBox 启动虚拟机的流程，直接将你的 Windows 系统中还原硬盘做好，之后同学们在开机后就可以直接使用了。当然，你的镜像文件（images）要放置在没有被还原的其他系统分区才行。

0.2：VirtualBox 的安装与设置

VirtualBox 是甲骨文（Oracle）公司根据 GPL 授权所发布的虚拟化软件，所以我们可以自由地从网络上面下载这款软件的最新版本来安装，VirtualBox 官网的链接如下：

https://www.virtualbox.org/

本书提供的下载压缩包中也包含了 Windows 下的 VirtualBox 安装程序，读者可以选择使用。如果读者使用的是非 Windows 操作系统，就请根据您自己的操作系统环境从上述的 VirtualBox 官网去选择要下载的相应版本。

在这里我们先对虚拟化系统的专有名词做个简单的解释：

- Hypervisor：虚拟机管理器，也称为虚拟机监视器，就是上面的 KVM，Xen...等软件的意思。
- Host：运行 Hypervisor 的主系统，就是实体计算机系统的意思。
- Virtual Machine：虚拟机（注意！是虚拟“机”不是虚拟“主机”），由 Hypervisor 所模拟出来的“假硬件”，简称为 VM。
- Guest：在虚拟机（VM）上运行的操作系统。

以我们为本书提供的训练教材来说，我们要在 Windows 系统上面建立虚拟化的 Linux 系统，因此 host 就是 Windows 操作系统， 虚拟机内的 guest OS 才是 Linux 操作系统之意。所以，当我们要下载 VirtualBox 时，务必请选择正确的 host 版本（在这个例子中，就是为 host 下载 Windows 的 VirtualBox 版本。）

0.2.1 安装前的准备工作

目前的 CentOS 7 已经不支持 32 位的硬件环境，但是许多的虚拟化系统还是只支持 32 位，好在 VirtualBox 是支持 64 位的。不过，我们要先在目标主机的 BIOS 中启动支持虚拟化的功能才行。在 BIOS 中设置虚拟化的支持其实很简单，只要重新启动目标计算机，进入 BIOS 之后，再进入高级（Advanced）设置模式，然后选择虚拟化（Virtualization）有关的设置即可，如图 0.1 所示，是在笔者的计算机的 BIOS 中设置虚拟化功能的示意图。

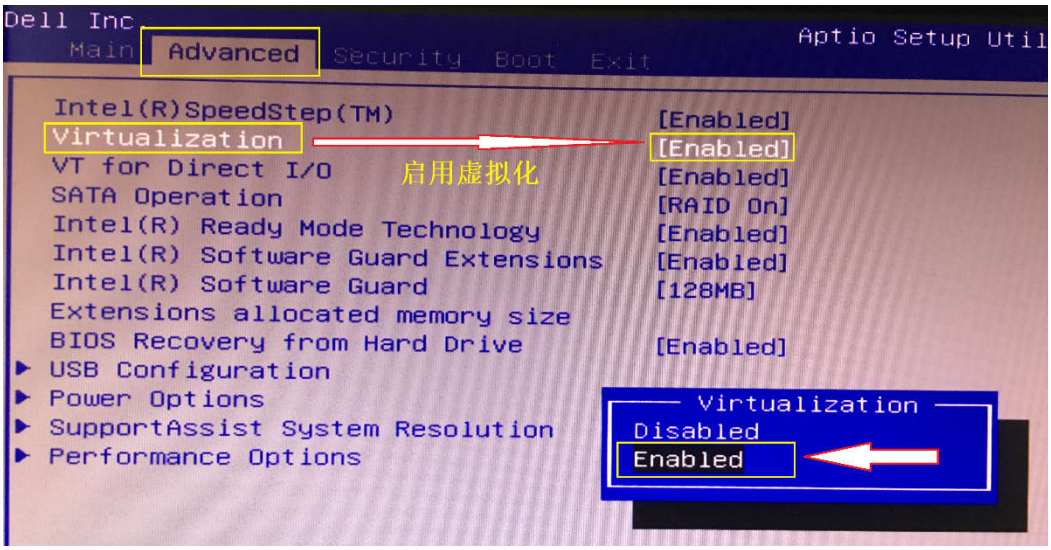

图 0.1 在 BIOS 中设置支持虚拟化的功能

随带说一下，如果我们这台主机将来还想要支持硬件的 PCI passthrough 功能，那么就需要启用“VT for Direct I/O”的功能。不过，在我们这个训练教材中，这个设置不是必须启用的（因为用不上），如图 0.2 所示。

图 0.2　在 BIOS 中启用 VT for Direct I/O 虚拟化功能

在 BIOS 中的设置结束之后，请从网上下载本书提供的压缩文件包，下载的网址为：

https://pan.baidu.com/s/1dyCF4sIrbOTrbkAgOmkArA

下载成功后，将里面的文件解压缩并复制到你的驱动器 D 中（确保你的驱动器 D 有足够的空余空间），这个压缩文件包应该包含下列软件：

- linux_os_train_class.vdi：就是主要的硬盘环境。
- VirtualBox-5.1.28-124319-Win.exe：VirtualBox 于 2018/08 下载的版本。
- readme.pdf：就是说明文件。
- linux_os_train_class.xml：启动 KVM 的 XML 设置（需要修改）。

至此，安装前的准备工作就完成了！

0.2.2 开始安装与设置

用鼠标双击 VirtualBox 安装文件，就可以进入安装界面，安装过程很简单，只要一直用鼠标单击“下一步”按钮即可。唯一要注意的是，VirtualBox 会安装一块虚拟网卡到我们的 Windows 系统上，在建立该虚拟网卡时， 我们的网络很可能会被暂时中断，因此记得安装时，需特别留意这一点。也就是说，不要有类似 ssh 的软件同时在运行，否则断线的话造成不必要的。

■ 软件安装

下面为按序安装流程的截图，没有特别的地方，就一直单击“下一步”按钮即可，如图 0.3 到图 0.10 所示。

图0.3　安装 VirtualBox 步骤 1

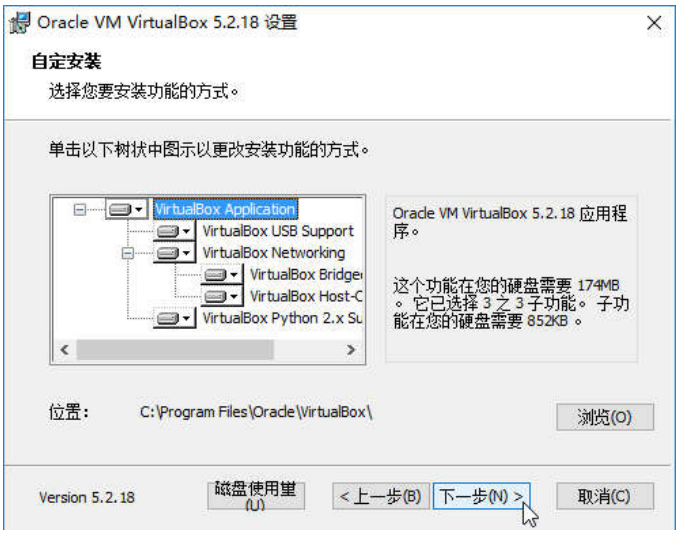

图0.4　安装 VirtualBox 步骤 2

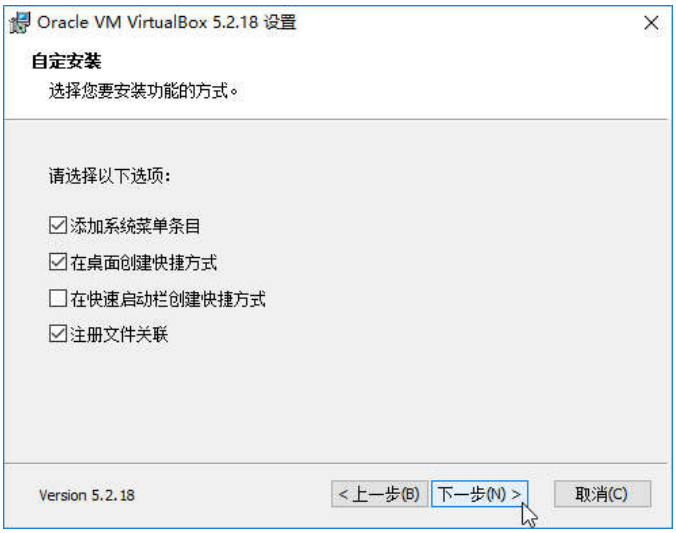

图0.5　安装 VirtualBox 步骤 3

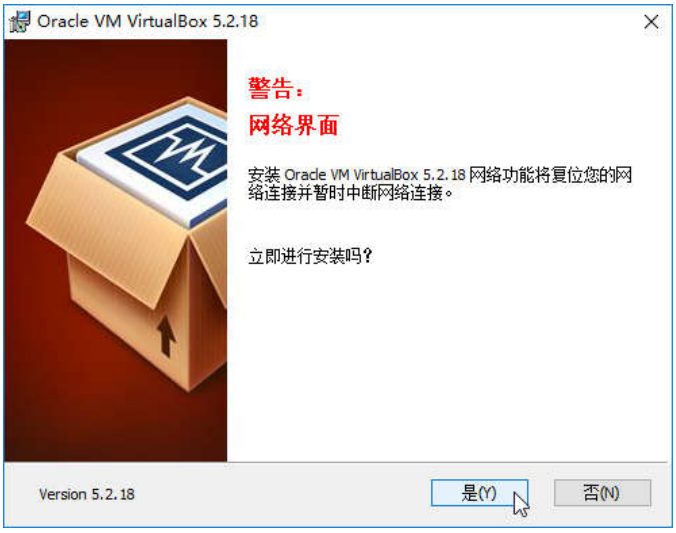

图0.6　安装 VirtualBox 步骤 4

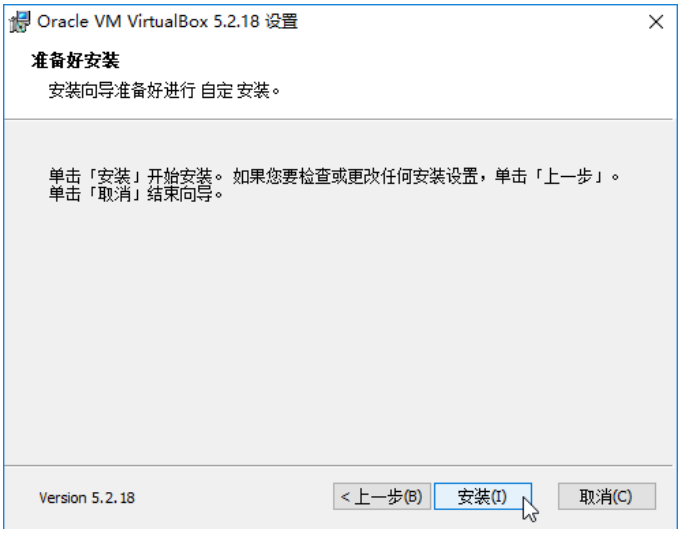

图0.7　安装 VirtualBox 步骤 5

图0.8　安装 VirtualBox 步骤 6

在图 0.8 中单击“是”按钮才能继续安装，不要按错了。

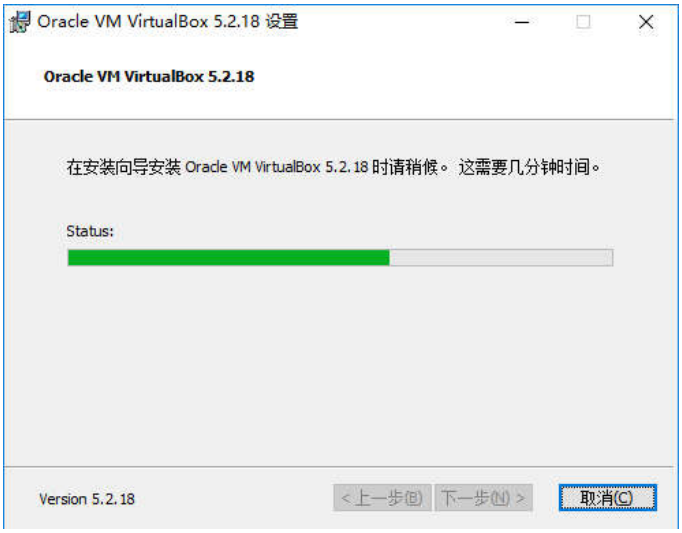

图0.9　安装 VirtualBox 步骤 7

图0.10　安装 VirtualBox 步骤 8

安装完毕之后，启动 VirtualBox，随后显示出如图 0.11 所示的界面。

图0.11　启动 VirtualBox 后的界面

■ 虚拟机的硬件配置及优化

安装完毕后，接下来单击“新建”按钮（如图 0.11），来创建一台虚拟机，于是显示出如图 0.12 所示的设置界面。因为我们的系统是 CentOS 也是来自于 Red Hat 系列，所以选择“Red HAT（64-bit）”这个选项。

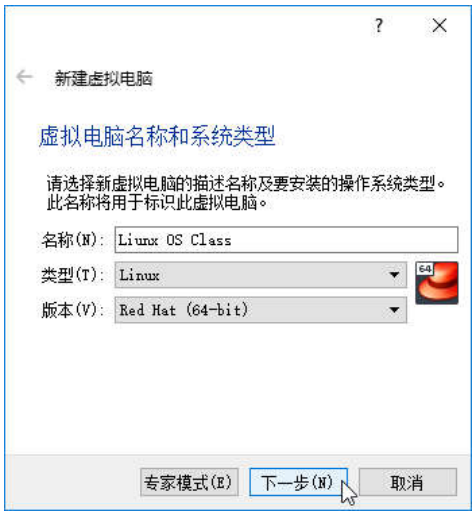

图0.12　单击“新建”虚拟机按钮后出现的设置界面

训练机大约需要 2GB（见图 0.13）到 3GB 的内存就够用了！如果你的实体主机（host）只有 4GB 内存的话，那么只申请 1.5GB 也行。

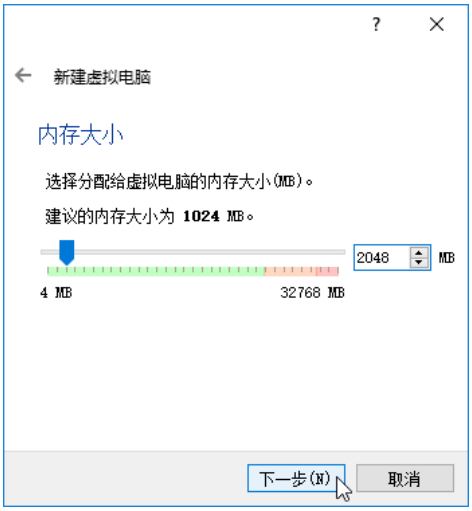

图0.13　为新建的虚拟机分配内存

图0.14所示的这个步骤很重要，请选择正确的虚拟硬盘文件，就是本书提下载的压缩文件包中的虚拟硬盘文件（文件名为linux_os_train_class.vdi）。如果你未来想要重复使用这个训练系统，请将这个虚拟硬盘文件复制多份保存好，一旦出错就可以将虚拟硬盘换掉或者将虚拟硬盘文件再复制过来。

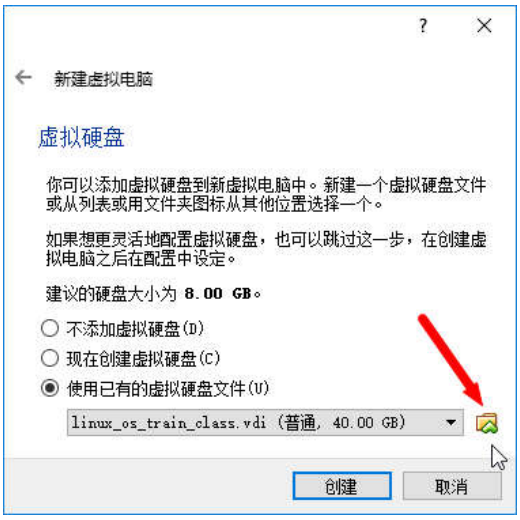

图0.14　为新建的虚拟机设置虚拟硬盘

全部设置完成后，VirtualBox 就会显示如图 0.15 所示的界面。

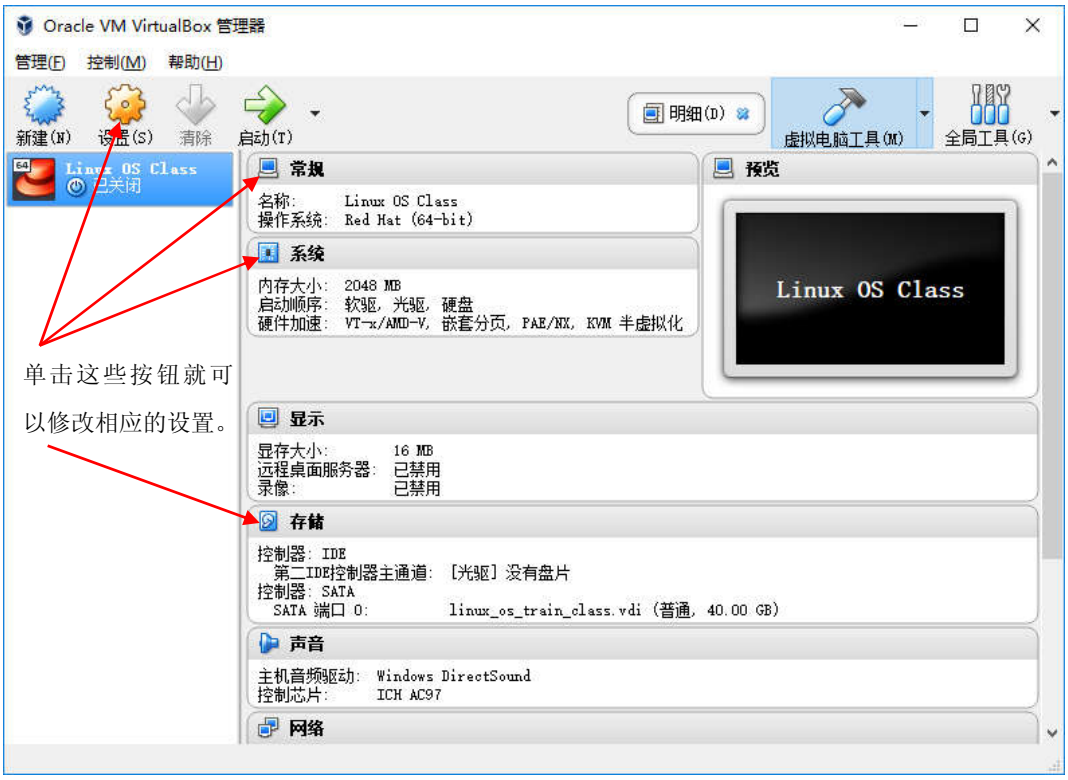

图0.15　完成新建虚拟机的全部设置好后出现的界面

然后单击“设置”按钮，再单击“系统”选项来优化一下系统的设置，如图 0.16 到图 0.18 所示。

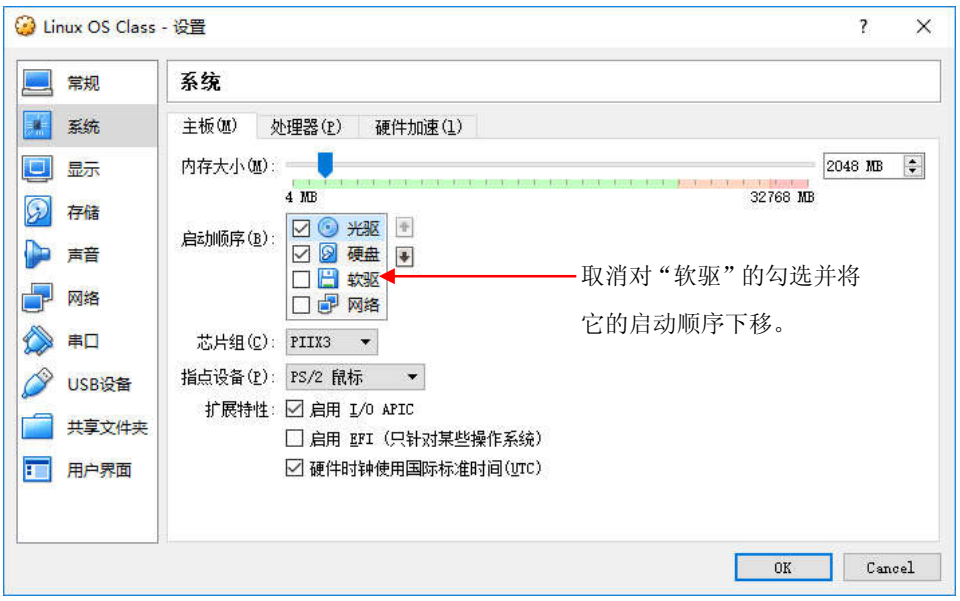

图0.16　优化虚拟机的系统启动顺序

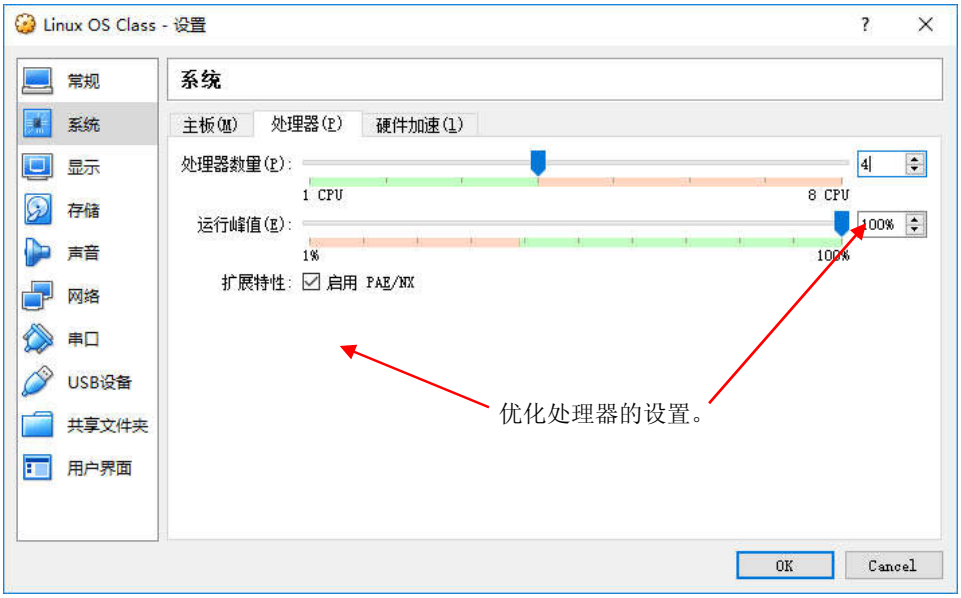

图0.17　优化虚拟机处理器的设置

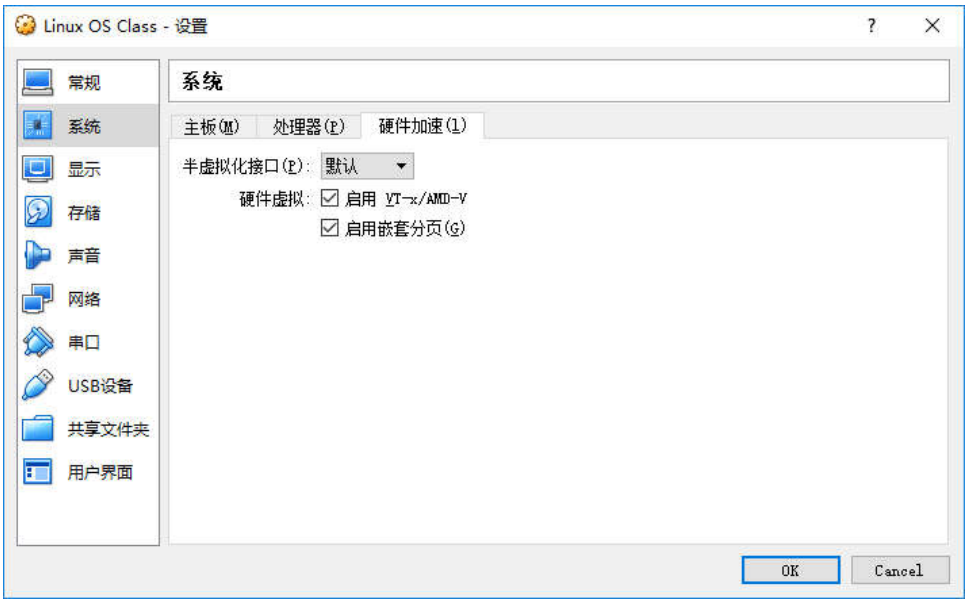

图0.18　优化虚拟机的硬件加速设置

系统默认只使用一颗处理器（CPU），那样的话你的系统就会跑得慢吞吞的，所以处理器的优化设置很重要！接下来再调整一下其他的设置，包括显示功能，如图 0.19 所示。

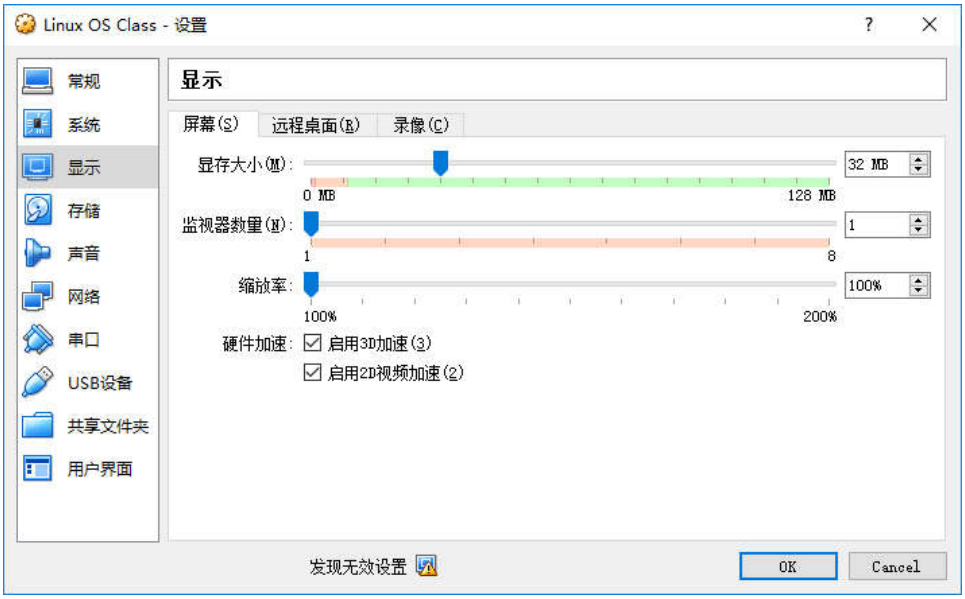

图0.19　调整虚拟机的显示设置

还要注意一下网络设置，因为我们的练习环境可能会与其他用户以及服务器互相沟通，因此最好能够使用桥接网卡。整个网络设置如图 0.20 所示。

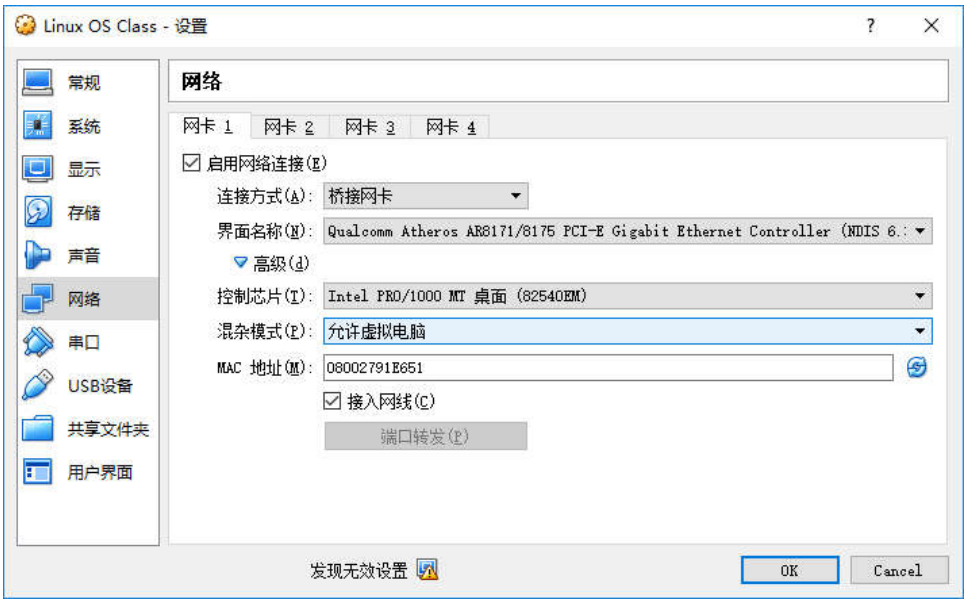

图0.20　虚拟机的网络设置

最终完成后，VirtualBox 的界面将如图 0.21 所示，然后单击“启动”按钮，就可以启动我们要使用的含有 Linux 训练教材的运行环境了！

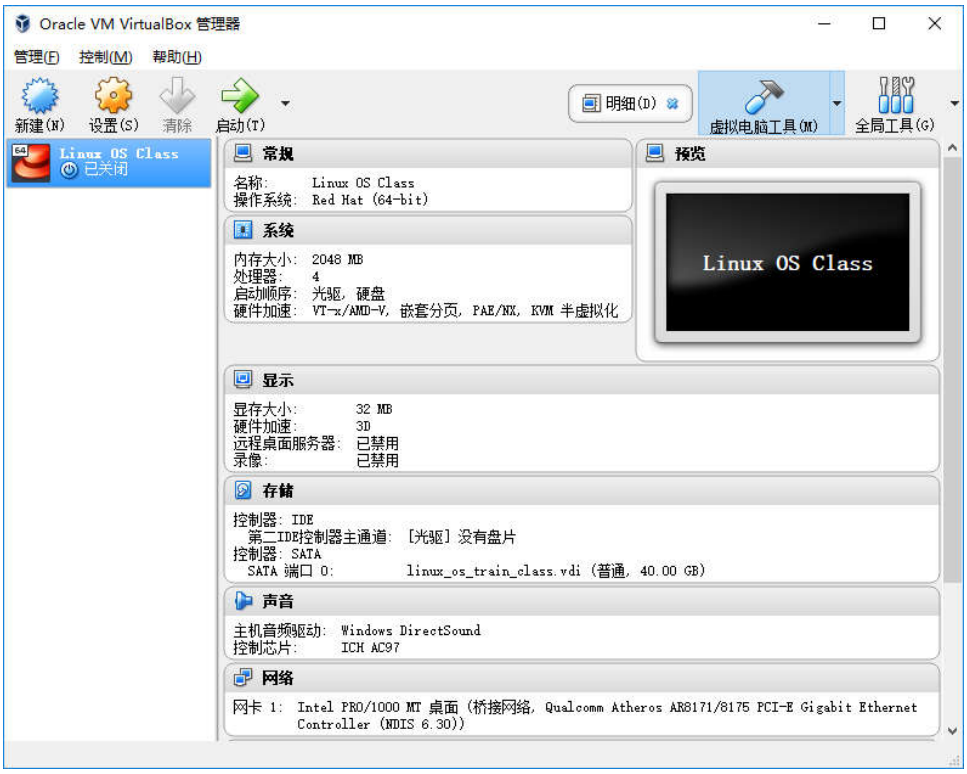

图0.21　虚拟机的各项设置完成

0.3 使用 KVM host 处理训练教材

假设我们已经安装好了 Linux Server（服务器），并且我们在计算机的 BIOS 中也已经设置好了对虚拟化的支持，这时我们只需要安装 libvirtd 这个服务，同时加上 qemu-kvm 和 qemu-img 这两个软件，就可以开始设置我们的 Linux 训练机器了。这样我们在虚拟机上所使用的操作系统就会和本书描述的内容一模一样，遵循教材的内容学习并进行操作训练就会比较顺畅。

如果读者手边有一台支持虚拟化的主机，那么也可以直接从网络上下载 CentOS 7 的镜像文件，然后按照网上的安装教学， 使用支持图形用户界面（GUI）的服务器安装方式安装 CentOS 7，安装完毕之后，也就能使用 Linux KVM 的虚拟化软件了。

如果读者还是不知道怎么启用虚拟化，那么请到实体计算机上去查看下列的数据，如果有的话，那就表示可以进行虚拟化操作了。

```
# 检查一下 CPU 有没有支持虚拟化的功能！如果找不到 vmx 的话，请到 BIOS 进行调整。
[root@study ~]# cat /proc/cpuinfo | grep vmx
flags : fpu vme de pse tsc msr pae ... vmx smx ... cqm_occup_llc
# 要确认 libvirtd 是处于启动的状态才行！
```

```
[root@study ~]# systemctl status libvirtd
● libvirtd.service - Virtualization daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2017-09-28 12:09:04 CST; 1h 42min ago
     Docs: man:libvirtd(8)
           http://libvirt.org
 Main PID: 2250 (libvirtd)
    CGroup: /system.slice/libvirtd.service
            └─2250 /usr/sbin/libvirtd
# 确认 virsh list 程序是可行的
[root@study ~]# virsh list
 Id Name State
----------------------------------------------------
# 检查已经安装了 qemu-img 以及 qemu-kvm 等软件!
[root@study ~]# rpm -q qemu-img qemu-kvm
qemu-img-1.5.3-105.el7_2.7.x86_64
qemu-kvm-1.5.3-105.el7_2.7.x86_64
```

■ 转换硬盘格式

假设我们已经把本书提供的下载压缩包中的虚拟硬盘文件已经放置到 /home/vbirdlinux 目录下，接下来我们要先转换硬盘格式才行。因为压缩包中虚拟硬盘文件主要是给 VirtualBox 使用的，所以我们要先将该格式转成 KVM 可以读取的 qcow2 格式。处理的方式其实非常简单，如下：

```
[root@study ~]# cd /home/vbirdlinux
[root@study vbirdlinux]# ll
-rw-rw-r-- 1 root root 3984753152 Sep 28 13:44 linux_os_train_class.vdi

[root@study vbirdlinux]# qemu-img convert -O qcow2 linux_os_train_class.vdi
linux_os_train_class.qcow2 -p (100.00/100%)

[root@study vbirdlinux]# ll
```

```
-rw-r--r-- 1 root root 3964010496 Sep 28 14:02 linux_os_train_class.qcow2
-rw-rw-r-- 1 root root 3984753152 Sep 28 13:44 linux_os_train_class.vdi

[root@study vbirdlinux]# qemu-img info linux_os_train_class.qcow2
image: linux_os_train_class.qcow2
file format: qcow2
virtual size: 40G (42949672960 bytes)
disk size: 7.0G
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

上面的 linux_os_train_class.qcow2 就是课程提供的上机训练用的硬盘文件，这个硬盘文件最好不要被误删或被改动了！因为如果万一硬盘文件出错了，我们可以通过这个硬盘文件来重新生成上机训练用的硬盘文件。为了加快硬盘处理的速度，我们可以通过 qemu-img 提供的 backing_file 功能。如果把这个硬盘文件的名称变成 linux_os_train_class.qcow2.raw 之后，就可以增加一个快照文件（backing_file），两个文件的内容会一摸一样，但是容量会相差很多。

```
[root@study vbirdlinux]# linux_os_train_class.qcow2 linux_os_train_class.qcow2.raw
[root@study vbirdlinux]# qemu-img create -f qcow2 -o backing_file =
linux_os_train_class.qcow2.raw \

> linux_os_train_class.qcow2
Formatting 'linux_os_train_class.qcow2', fmt=qcow2 size=42949672960 backing_file =
'linux_os_train_class.qcow2.raw' encryption = off cluster_size=65536 lazy_refcounts=off

[root@study vbirdlinux]# ll
-rw-r--r-- 1 root root 197632 Sep 28 14:08 linux_os_train_class.qcow2
-rw-r--r-- 1 root root 3964010496 Sep 28 14:02 linux_os_train_class.qcow2.raw
-rw-rw-r-- 1 root root 3984753152 Sep 28 13:44 linux_os_train_class.vdi

[root@study vbirdlinux]# qemu-img info linux_os_train_class.qcow2
image: linux_os_train_class.qcow2
file format: qcow2
virtual size: 40G (42949672960 bytes)
disk size: 196K
cluster_size: 65536
```

```
backing file: linux_os_train_class.qcow2.raw
Format specific information:
    compat: 1.1
    lazy refcounts: false
```

■ 编辑虚拟机（VM）硬件配置文件

接下来请打开 "linux_os_train_class.xml" 这个文件，因为我们放置文件的位置与相关的设置与笔者当初的设置并不相同，所以这里要修改一下才行。

```
[root@study vbirdlinux]# ll
-rw-rw-r-- 1 root root 2577 Sep 28 14:11 linux_os_train_class.xml

[root@study vbirdlinux]# vim linux_os_train_class.xml
<domain type='kvm' xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'>
  <name>vm_linux</name>
  <memory>2097152</memory>
  <currentMemory>2097152</currentMemory>
  <vcpu>4</vcpu>
    <cpu mode='host-passthrough'>
      <arch>x86_64</arch>
      <model>SandyBridge</model>
      <vendor>Intel</vendor>
      <topology sockets='1' cores='4' threads='2'/>
    </cpu>
  <os>
    <type arch='x86_64'>hvm</type>
    <boot dev='cdrom'/>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
```

```
<devices>
    <emulator>/usr/libexec/qemu-kvm</emulator>
    <disk type='file' device='disk'>
        <driver name='qemu' type='qcow2' cache='none' io='native' copy_on_read='off' />
        <source file="/home/vbirdlinux/linux_os_train_class.qcow2"/>
        <target dev='vda' bus='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
    </disk>
    <disk type='file' device='cdrom'>
        <source file="/>
        <target dev='hdc' bus='ide'/>
        <readonly/>
    </disk>
    <interface type='bridge'>
        <mac address='52:54:00:11:11:33'/>
        <source bridge='br0'/>
        <model type='virtio'/>
        <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
    </interface>
    <serial type='pty'>
        <target port='0'/>
    </serial>
    <console type='pty'>
        <target port='0'/>
    </console>
    <input type='tablet' bus='usb'/>
    <input type='mouse' bus='ps2'/>
    <graphics type='spice' port='5701' autoport='no' listen='0.0.0.0'>
        <image compression='auto_glz' />
        <jpeg compression='auto' />
        <zlib compression='auto' />
        <playback compression='on' />
        <streaming mode='all' />
    </graphics>
    <graphics type='vnc' port='5901' autoport='no' listen='0.0.0.0'>
        <image compression='auto_glz' />
        <jpeg compression='auto' />
        <zlib compression='auto' />
```

```
      <playback compression='on' />
      <streaming mode='all' />
    </graphics>
    <video>
      <model type='qxl' vram='32768' heads='1'/>
      <acceleration accel3d='yes' accel2d='yes'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
    </video>
    <sound model='ich6'/>
    <memballoon model='virtio'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
    </memballoon>
  </devices>
</domain>
```

如果 KVM host 已经安装好了桥接设备，即是 br0 这一个桥接界面，如果我们输入 ifconfig 后发现只有 virbr0 时，就将上述的网桥改成 virbr0 即可。接着就是同时启动 spice 与 vnc 两个界面，它们采用默认端口，如果不想要使用上述的端口，请自行更改该端口。此外，如果想要从外网连接到这个系统，那么最好加上密码。目前默认是没有加密码的，如果要加密码，修改下面这一行：

```
[root@study vbirdlinux]# vim linux_os_train_class.xml
    <graphics type='spice' port='5701' autoport='no' listen='0.0.0.0' passwd="your_passwd">
```

如果一切都准备妥当，就可以启动虚拟机了：

```
[root@study vbirdlinux]# virsh create linux_os_train_class.xml
Domain vm_linux created from linux_os_train_class.xml

[root@study vbirdlinux]# virsh list
 Id    Name                           State
----------------------------------------------------
 2     vm_linux                       running

[root@study vbirdlinux]# netstat -tlunp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:5901            0.0.0.0:*               LISTEN      7134/qemu-kvm
tcp        0      0 0.0.0.0:5701            0.0.0.0:*               LISTEN      7134/qemu-kvm
```

此时，只要在 Linux 的图形界面中启动 remove-viewer，然后输入“spice://localhost:5701”即可进入图形页面。如果是使用外部的 IP，则需要让防火墙放行，之后在 remove viewer 软件中输入“spice://your_kvm_host_ip:5701”即可获取上课用的训练教材。