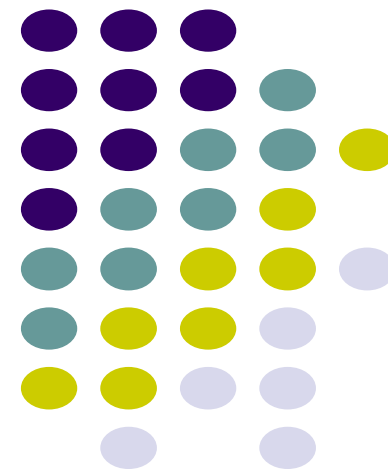
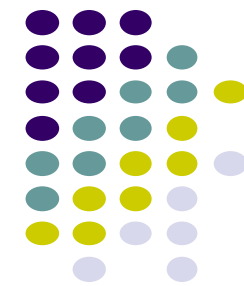


《计算机系统基础（四）：编程与调试实践》

从程序员角度认识系统



计算机系统基础—从程序员角度认识系统



- 目标：

培养学生的**系统能力**，使其成为一个“高效”程序员，在程序调试、性能提升、程序移植和健壮性等方面成为高手；建立扎实的计算机系统概念，为后续的OS、编译、体系结构等课程打下坚实基础。

实例1——字面量比较



ISO C90标准下，在32位系统上 以下C表达式的结果是什么？

$-2147483648 < 2147483647$

false（与事实不符）！

ISO C99标准下为true

以下关系表达式结果呢？

`int i = -2147483648;`

`i < 2147483647`

true！

理解该问题需要知道：

编译器如何处理字面量

高级语言中运算规则

高级语言与指令之间的对应

机器指令的执行过程

机器级数据的表示和运算

.....

$-2147483647-1 < 2147483647$ ，结果怎样？

编译器处理常量时默认的类型



- C90



范围	类型
$0 \sim 2^{31}-1$	int
$2^{31} \sim 2^{32}-1$	unsigned int
$2^{32} \sim 2^{63}-1$	long long
$2^{63} \sim 2^{64}-1$	unsigned long long

$2^{31} = 2147483648$, 机器数为 : 100 ... 0 (31个0)

- C99



范围	类型
$0 \sim 2^{31}-1$	int
$2^{31} \sim 2^{63}-1$	long long
$2^{63} \sim 2^{64}-1$	unsigned long long

实例1——字面量比较



ISO C90标准下，在32位系统上 以下C表达式的结果是什么？

$-2147483648 < 2147483647$

false（与事实不符）！

ISO C99标准下为true

以下关系表达式结果呢？

`int i = -2147483648;`

`i < 2147483647`

true！

理解该问题需要知道：

编译器如何处理字面量

高级语言中运算规则

高级语言与指令之间的对应

机器指令的执行过程

机器级数据的表示和运算

.....

$-2147483647-1 < 2147483647$ ，结果怎样？

实例2——运算规则



当用len=0调用sum函数时，其返回值应该是多少？

```
sum(int a[ ], unsigned len)
{
    int i, sum = 0;
    for (i = 0; i <= len-1; i++)
        sum += a[i];
    return sum;
}
```

当参数len为0时，返回值应该是0，但是在机器上执行时，却发生访存异常。但当len为int型时则正常。为什么？

理解该问题需要知道：

高级语言中运算规则

机器指令的含义和执行

计算机内部的运算电路

异常的检测和处理

虚拟地址空间

.....

实例3——整数乘法运算及溢出



```
/* 复制数组到堆中，count为数组元素个数 */  
int copy_array(int *array, int count) {  
    int i;  
    /* 在堆区申请一块内存 */  
    int *myarray = (int *) malloc(count*sizeof(int));  
    if (myarray == NULL)  
        return -1;  
    for (i = 0; i < count; i++)  
        myarray[i] = array[i];  
    return count;  
}
```

当 $\text{count}=2^{30}+1$ 时，程序会发生什么情况？

理解该问题需要知道：

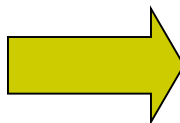
乘法运算及溢出

虚拟地址空间

存储空间映射

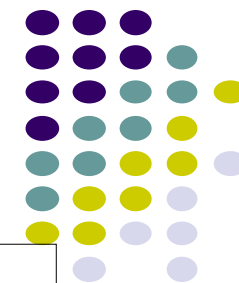
.....

当参数count很大时，则
 $\text{count} \times \text{sizeof}(\text{int})$ 会溢出。如
 $\text{count}=2^{30}+1$ 时，
 $\text{count} \times \text{sizeof}(\text{int})=4$ 。



堆 (heap) 中大量数据被破坏！

实例4——强弱符号



main.c

```
int d=100;
int x=200;
int main()
{
    p1( );
    printf ( "d=%d, x=%d\n" , d, x );
    return 0;
}
```

p1.c

```
double d;

void p1( )
{
    d=1.0;
}
```

打印结果是什么？

d=0 , x=1 072 693 248

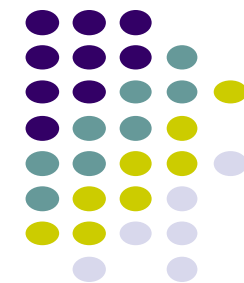
Why ?

理解该问题需要知道：

- 机器级数据的表示
- 变量的存储空间分配
- 数据的大端/小端存储方式
- 链接器的符号解析规则

.....

实例4——强弱符号



打印结果：d=0 , x=1 072 693 248

double型数1.0对应的机器数3FF0 0000 0000 0000H

IA-32是小端方式

	0	1	2	3
<i>&x</i>	00	00	F0	3F
<i>&d</i>	00	00	00	00

高



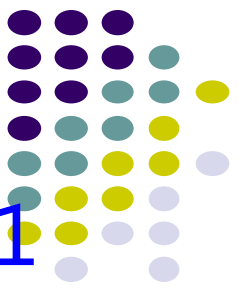
低

$$2^{30}-1-(2^{20}-1)=2^{30}-2^{20}$$

$$=1024*1024*1023$$

$$=1\ 072\ 693\ 248$$

实例5——浮点运算指令



- 使用老版本gcc -O2编译时，程序一输出0，程序二输出却是1

程序一：

```
#include<stdio.h>
double f(int x){
    return 1.0 / x;
}
void main(){
    double a,b;
    int i;
    a=f(10);
    b=f(10);
    i=a==b;
    printf("%d\n",i);
}
```

程序二：

```
#include<stdio.h>
double f(int x){
    return 1.0 / x;
}
void main(){
    double a,b,c;
    int i;
    a=f(10);
    b=f(10);
    c=f(10);
    i=a==b;
    printf("%d\n",i);
}
```

实例5——浮点运算指令



double f(int x)	8048328: 55	push %ebp
{	8048329: 89 e5	mov %esp,%ebp
return 1.0 / x ;	804832b: d9 e8	fld1
}	804832d: da 75 08	fidivl 0x8(%ebp)
	8048330: c9	leave
	8048331: c3	ret

两条重要指令的功能如下。

fld1 : 将常数1压入栈顶ST(0)

fidivl : 将指定存储单元操作数M[R[ebp]+8]中的int型数转换为double型 ,
再将ST(0)除以该数 , 并将结果存入ST(0)中

f(10)=0.1

0.1=0.00011[0011]B= 0.00011 0011 0011 0011 0011 0011 0011...B

实例5——浮点运算指令

08048334 <main>:

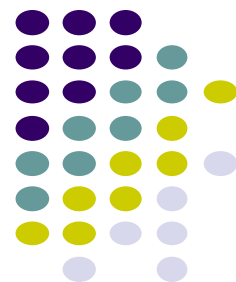
```
8048334: 55          push  %ebp
8048335: 89 e5       mov   %esp,%ebp
8048337: 83 ec 08    sub   $0x8,%esp
804833a: 83 e4 f0    and   $0xffffffff0,%esp
804833d: 83 ec 0c    sub   $0xc,%esp
8048340: 6a 0a       push  $0xa
8048342: e8 e1 ff ff ff call 8048328 <f> //计算a=f(10)
8048347: dd 5d f8    fstpl 0xffffffff8(%ebp) //a存入内存
804834a: c7 04 24 0a 00 00 00 movl $0xa,(%esp)
8048351: e8 d2 ff ff ff call 8048328 <f> //计算b=f(10)
8048356: dd 45 f8    fldl 0xffffffff8(%ebp) //a入栈顶
8048359: 58         pop   %eax
804835a: da e9       fucompp //比较ST(0)a和ST(1)b
804835c: df e0       fnstsw %ax //把FPU状态字送到AX
804835e: 80 e4 45    and   $0x45,%ah
8048361: 80 fc 40    cmp   $0x40,%ah
8048364: 0f 94 c0    sete  %al
8048367: 5a         pop   %edx
8048368: 0f b6 c0    movzbl %al,%eax
804836b: 50         push  %eax
804836c: 68 d8 83 04 08 push $0x80483d8
8048371: e8 f2 fe ff ff call 8048268 <_init+0x38>
8048376: c9         leave
8048377: c3         ret
```

```
...
a = f(10) ;
b = f(10) ;
i = a == b;
...
```

80位→64位

64位→80位

0.1是无限循环小数，无法精确表示，因而，比较时，a舍入过而b没有舍入过，故 a≠b



实例5——浮点运算指令

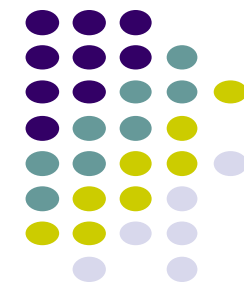


```
8048342: e8 e1 ff ff ff call 8048328 <f> //计算a
8048347: dd 5d f8 fstpl 0xffffffff8(%ebp) //把a存回内存
//a产生精度损失
804834a: c7 04 24 0a 00 00 00 movl $0xa,(%esp,1)
8048351: e8 d2 ff ff ff call 8048328 <f> //计算b
8048356: dd 5d f0 fstpl 0xffffffff0(%ebp) //把b存回内存
//b产生精度损失
8048359: c7 04 24 0a 00 00 00 movl $0xa,(%esp,1)
8048360: e8 c3 ff ff ff call 8048328 <f> //计算c
8048365: dd d8 fstp %st(0)
8048367: dd 45 f8 fldl 0xffffffff8(%ebp) //从内存中载入a
804836a: dd 45 f0 fldl 0xffffffff0(%ebp) //从内存中载入b
804836d: d9 c9 fxch %st(1)
804836f: 58 pop %eax
8048370: da e9 fucompp //比较a , b
8048372: df e0 fnstsw %ax
```

```
...
a = f(10) ;
b = f(10) ;
c = f(10) ;
i = a == b;
...
```

0.1是无限循环小数，无法精确表示，因而，比较时，a和b都是舍入过的，故 a=b！

小结



- ▣ 本次课介绍了几个C语言实例，并说明了用系统思维来分析编程问题的重要性。当然，要学会用系统思维方式来分析问题还需要我们掌握扎实的基本概念以及熟练使用各种工具！

《计算机系统基础（四）：编程与调试实践》

虚拟机、Linux及其上实验环境的安装

讲义目录

1. 安装虚拟机软件和创建虚拟机

- 安装VirtualBox虚拟机软件
- 创建和配置课程所用虚拟机

2. 在虚拟机中安装Linux系统

- 在虚拟机中安装和配置课程实验所用Linux系统平台

3. Linux实验环境配置

- 在Linux虚拟机系统中配置实验所需操作、开发和调试环境

1. 虚拟机软件及其安装

虚拟机（virtual machine）

- 虚拟机软件是运行于物理计算机及其操作系统上的一种提供了虚拟化（**virtualization**）功能的应用软件
 - 物理计算机可以是虚拟机软件支持的任意平台，例如基于**Intel**或 **AMD CPU**的**PC**
 - 物理计算机上可运行虚拟机软件所支持的任意操作系统，如**Windows, Mac OS X, Linux**等
- 虚拟机软件扩展了物理计算机系统的功能，使其上可在运行原操作系统（称为**host**操作系统或宿主操作系统）的同时，运行多个不同的操作系统（称为**guest**操作系统）——每个运行于一个虚拟机软件所创建的虚拟机（**virtual machine**）环境中。例如：
 - 在日常使用的**Windows**或**Mac**系统上运行**Linux**系统



为什么需要使用虚拟机

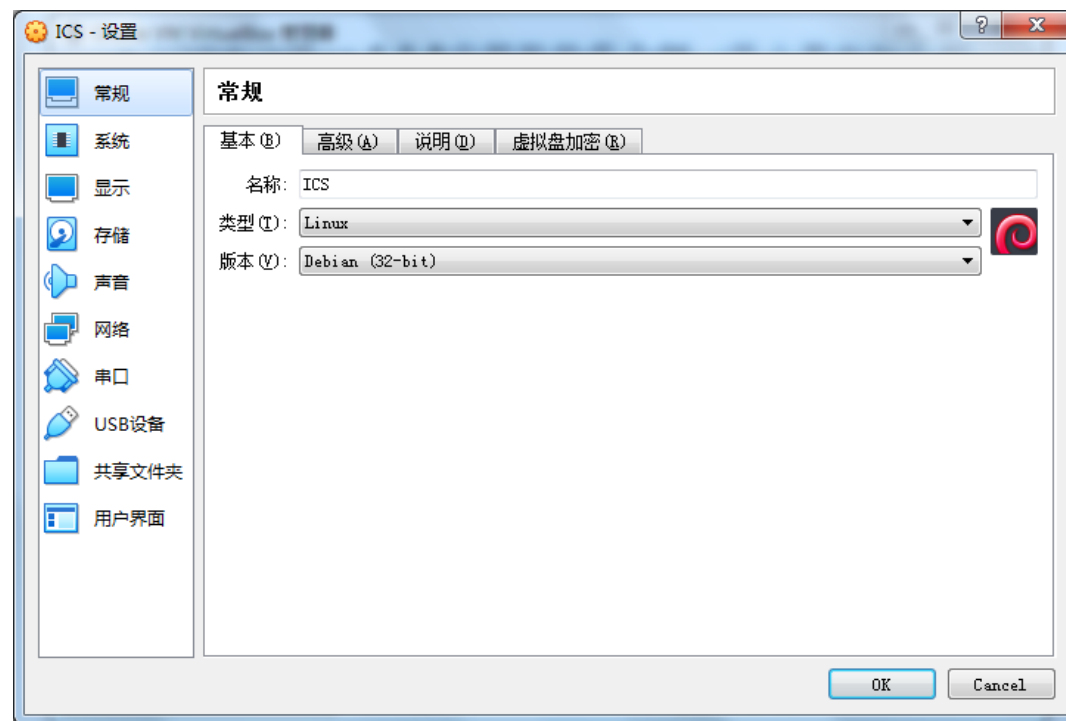
- 本课程实验需要在**Intel 32位x86**硬件平台和特定版本的**Linux**操作系统上开展
- 安装和使用虚拟机开展实验可有效解决：
 - 用户硬件平台和已有操作系统（**Windows、Mac OS X**等）不同于课程实验所需的上述**Linux**操作系统和平台
 - 实验同时需访问、使用原**Windows/Mac**操作系统上的数据或服务

VirtualBox虚拟机软件

- 常见虚拟机软件
 - Oracle VM VirtualBox: <https://www.virtualbox.org>
 - VMware: <https://www.vmware.com>
 - QEMU: <https://www.qemu.org>
 -
- 本实验说明主要针对**VirtualBox**虚拟机软件，建议将其用于课程中。其它虚拟机软件的安装和使用与**VirtualBox**存在不同程度上的类似，请自行参阅各软件说明文档
- 主要术语：
 - **Host**操作系统：物理计算机（相对于虚拟计算机）上安装运行的操作系统（例如**Windows**、**Mac OS X**），其中安装了虚拟机软件（例如**VirtualBox**）
 - **Guest**操作系统：虚拟机中运行的操作系统，例如可以是**Linux**、**Windows**等
 - 虚拟机（**Virtual machine - VM**）：虚拟机软件创建的、供**guest**操作系统在其中运行的特定环境

VirtualBox虚拟机软件

- 以VirtualBox 6.0.8虚拟机软件为例，其主界面如左图：



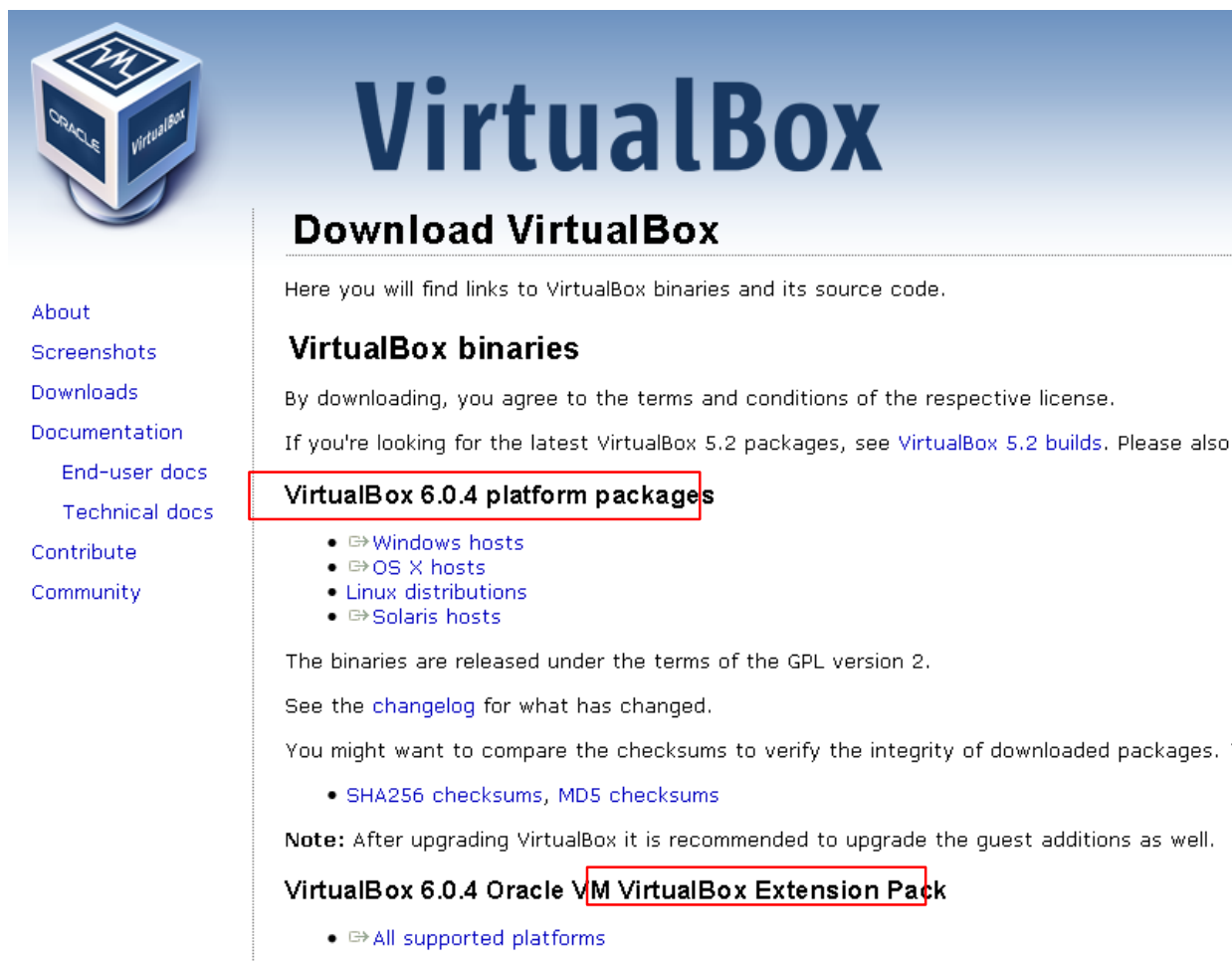
- 其中，已创建了一个名为“ICS”的虚拟机，其参数设置界面如右图所示，显示其中安装了Linux类型的guest操作系统

安装和配置VirtualBox

- 按照实际所用Host操作系统的类型，从如下网址下载合适的VirtualBox版本并按照说明安装（可使用默认安装设置）：

- 注意在下载安装完成VirtualBox自身之后，另下载和安装VirtualBox Extension Pack，以更好地与Host操作系统集成

<https://www.virtualbox.org>

A screenshot of the VirtualBox website. The header features the VirtualBox logo and the text "VirtualBox". Below the header, there is a section titled "Download VirtualBox" with a sub-header "VirtualBox binaries". The main content area lists links for "VirtualBox 6.0.4 platform packages" and "VirtualBox 6.0.4 Oracle VM VirtualBox Extension Pack". The left sidebar contains links for "About", "Screenshots", "Downloads", "Documentation", "End-user docs", "Technical docs", "Contribute", and "Community".

VirtualBox

Download VirtualBox

Here you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

By downloading, you agree to the terms and conditions of the respective license.

If you're looking for the latest VirtualBox 5.2 packages, see [VirtualBox 5.2 builds](#). Please also

VirtualBox 6.0.4 platform packages

- [Windows hosts](#)
- [OS X hosts](#)
- [Linux distributions](#)
- [Solaris hosts](#)

The binaries are released under the terms of the GPL version 2.

See the [changelog](#) for what has changed.

You might want to compare the checksums to verify the integrity of downloaded packages. [i](#)

- [SHA256 checksums](#), [MD5 checksums](#)

Note: After upgrading VirtualBox it is recommended to upgrade the guest additions as well.

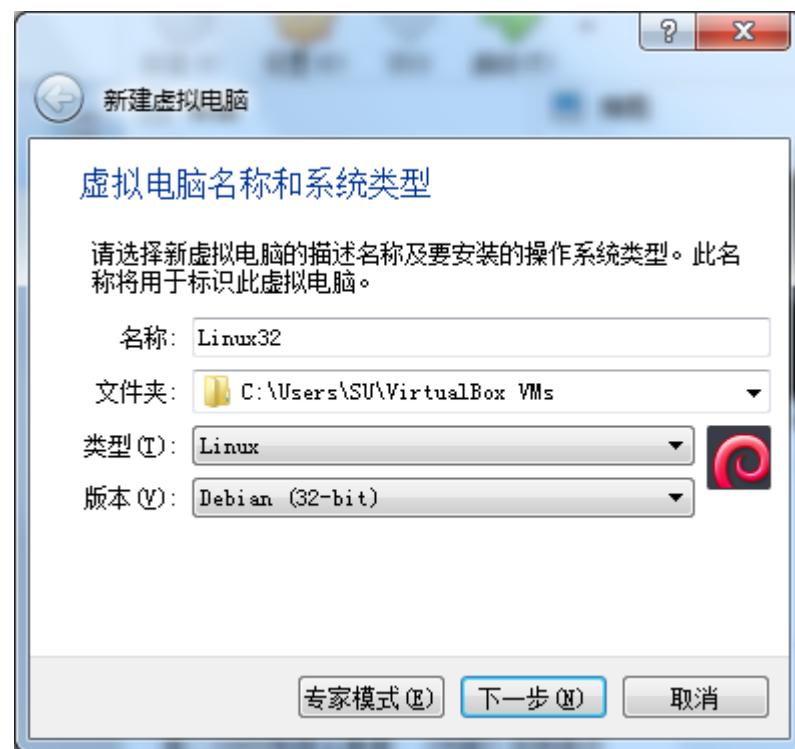
VirtualBox 6.0.4 Oracle VM VirtualBox Extension Pack

- [All supported platforms](#)

创建和配置VirtualBox虚拟机

创建和配置VirtualBox虚拟机

- 在安装好的VirtualBox软件中选择“新建”按钮，为实验将使用的Linux系统创建一个新的虚拟机
- 在右图所示新建虚拟机基本信息框中，输入或选择如下设置：
 - 名称任选（此处为“Linux32”）
 - 类型选“Linux”
 - 版本选“Debian (32-bit)”
 - 注意务必选择“32-bit”版本
 - 后续实验说明主要针对Debian 32-bit Linux
 - 选择创建虚拟硬盘
- 选择“下一步”按钮继续配置

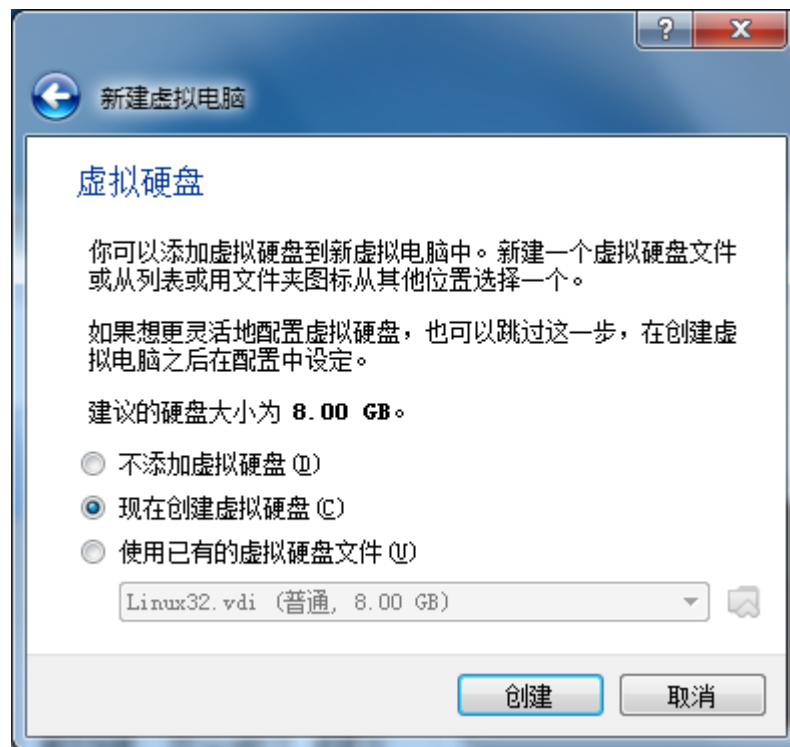


创建和配置VirtualBox虚拟机

- 在下图所示配置虚拟机内存的对话框中，输入合适的内存大小
 - 1024 MB已可满足课程实验需要

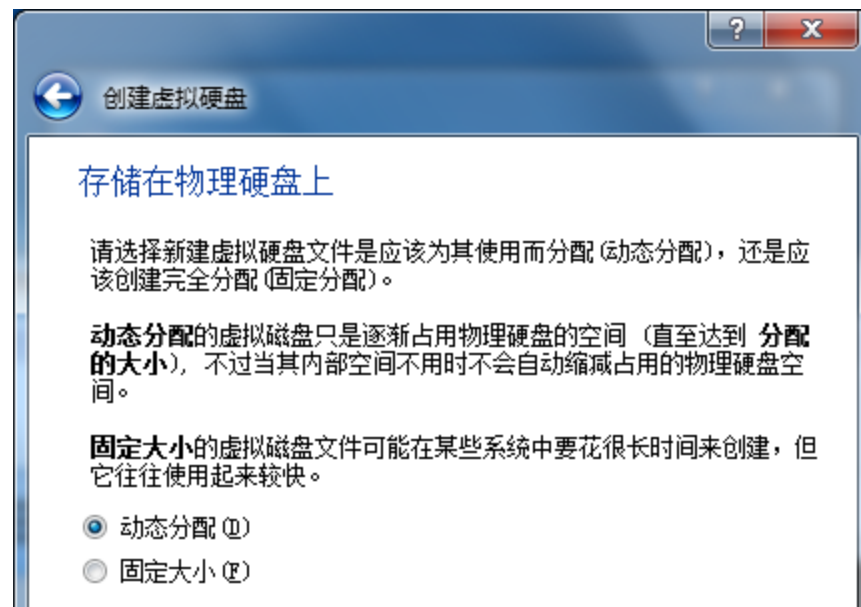
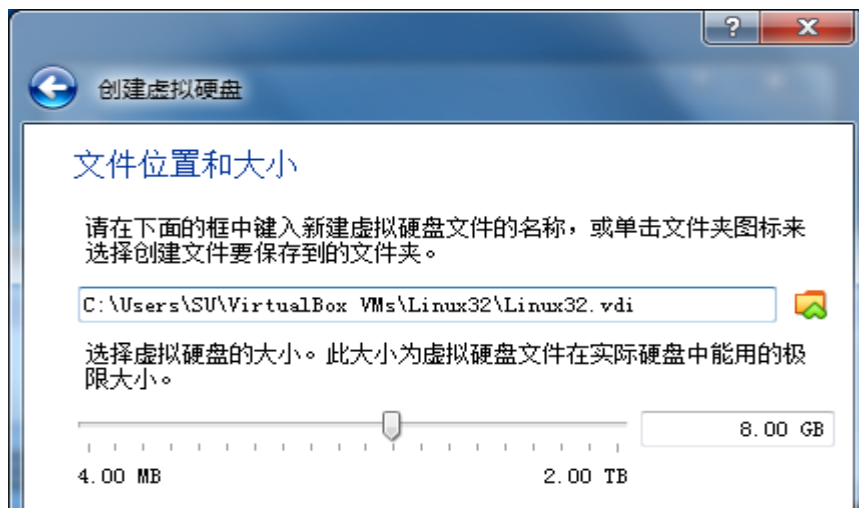
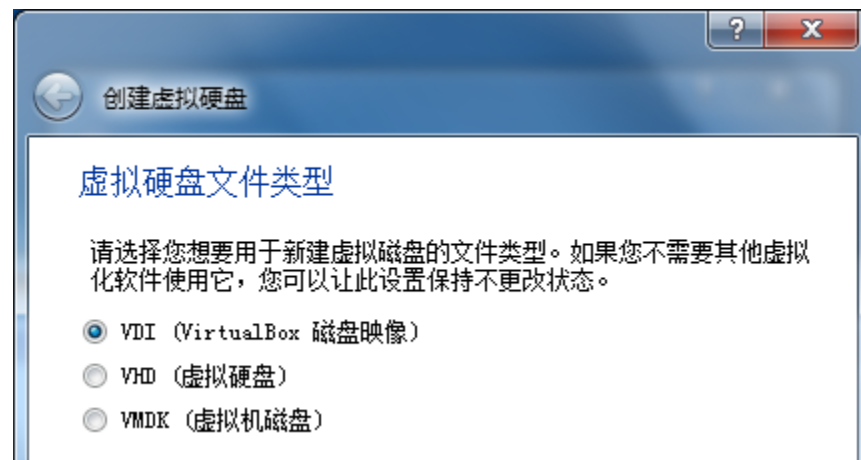


- 在下图所示虚拟硬盘信息框中，选择“**现在创建虚拟硬盘**”
 - 选择“创建”按钮



创建和配置VirtualBox虚拟机

- 在右图所示的创建虚拟硬盘向导显示的**虚拟硬盘文件类型**设置对话框中，可接受默认选项-**VDI**
- 在右图所示虚拟硬盘存储选项对话框中，可接受默认选项-**动态分配**：
- 在虚拟硬盘文件设置对话框中：
 - 在编辑框中输入或选择虚拟机的虚拟硬盘文件的保存目录和文件名
 - 选择合适的“文件大小”——即对应虚拟机中的虚拟硬盘的大小，默认的**8GB**可满足基本实验需求
- 选择“创建”按钮



2. Linux虚拟机安装

安装Linux虚拟机

- 从下列网站下载 **【本学期的Debian 10.2】** 稳定版Debian 32-bit (i386) Linux安装ISO文件（CD或DVD均可）：

<https://www.debian.org/CD/http-ftp/>

✓ DVD → i386: 下载DVD-1 ISO——包含较为完整的软件包集合

- 例如: **debian-10.2.0-i386-DVD-1.iso**

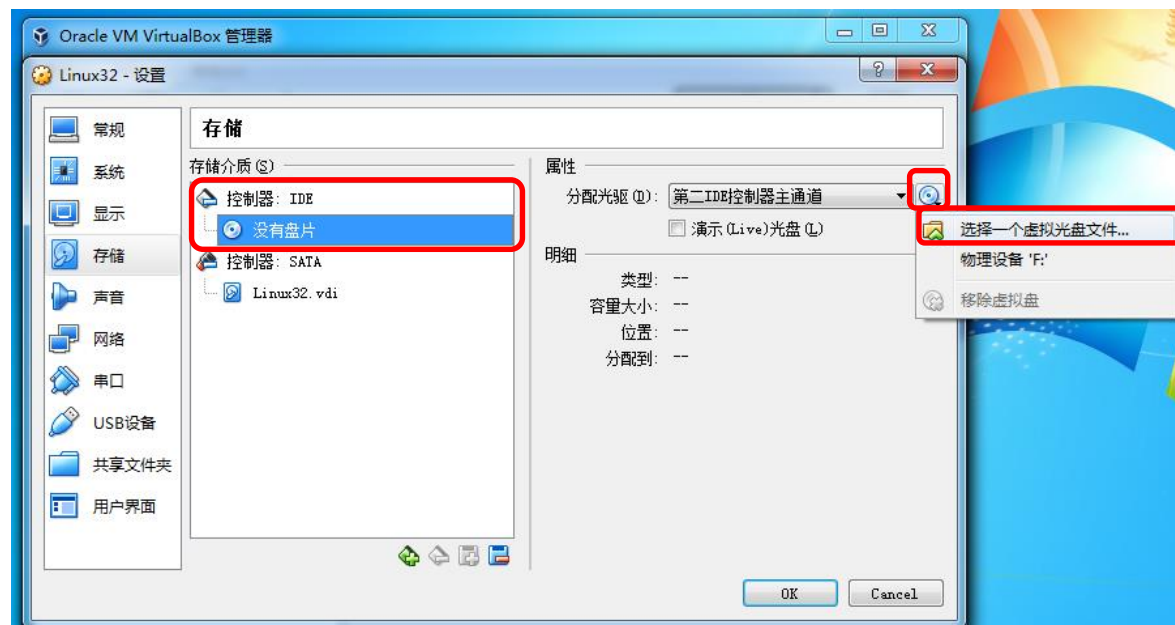
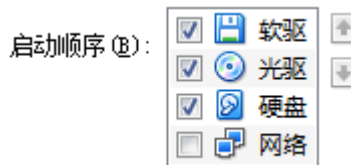
✓ CD → i386: 下载CD-1 ISO——包含精简的基本软件包集合

- 例如: **debian-10.2.0-i386-xfce-CD-1.iso**

- 如果安装时有互联网连接，可选择**CD ISO**安装，安装过程更短且占用更少硬盘空间（**CD**中未包含的软件包可从互联网下载）；否则，建议选择**DVD ISO**安装，以包含实验可能需要的所有软件包
- 在随后说明中，用**debian-i386.iso**指代上述两种可用ISO形式中的任一个，不同版本Debian Linux的安装过程大同小异

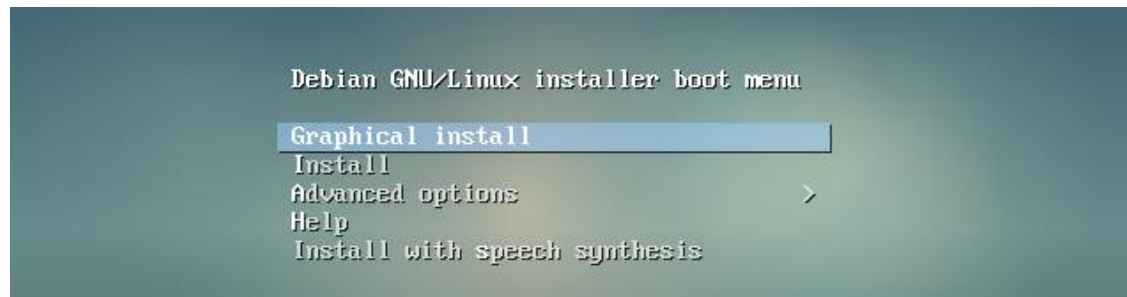
安装Linux虚拟机

- 在新建的Linux虚拟机中装载Linux安装光盘debian-i386.iso:
 1. 在VirtualBox管理器中选中新建的Linux虚拟机，选择“设置”按钮
 2. 在设置对话框左侧栏中选择“存储”，在右侧选择“控制器：IDE”下标有“没有盘片”文字的光盘图标
 3. 点击“属性”栏的“分配光驱”栏右侧的光盘图标，在弹出菜单中选择“选择一个虚拟光盘文件...”，如下图所示
 4. 在弹出的文件选择对话框中选中下载的debian-i386.iso文件
 5. 确认在设置对话框的“系统”页面的“启动顺序”中光驱优先于硬盘（默认如此）——以便从光驱启动安装



安装Linux虚拟机

1. 在VirtualBox管理器中选中Linux虚拟机，选择“启动”按钮
2. 在虚拟机窗口中的如下Debian Linux安装引导菜单界面中选择“**Graphical Install**”（图形安装界面）或“**Install**”（文字安装界面）——以下说明以图形安装界面为例

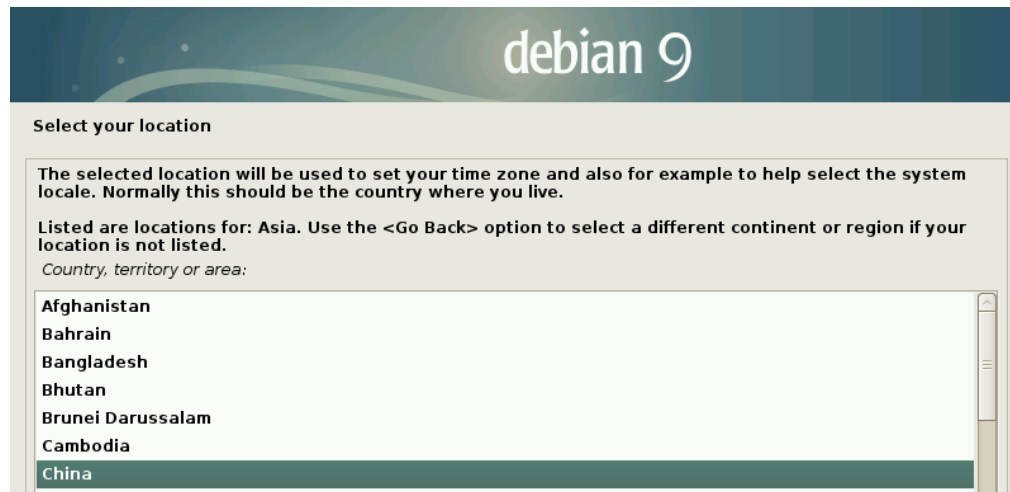


3. 【建议】选择English作为安装和系统的默认语言

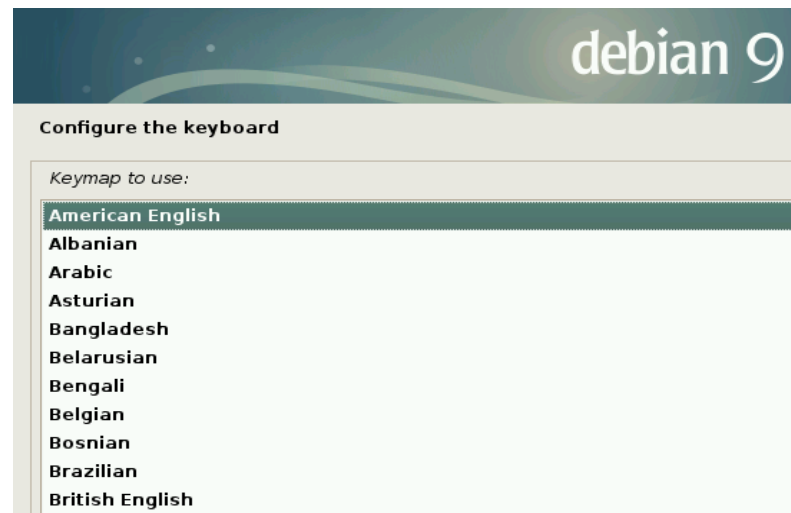
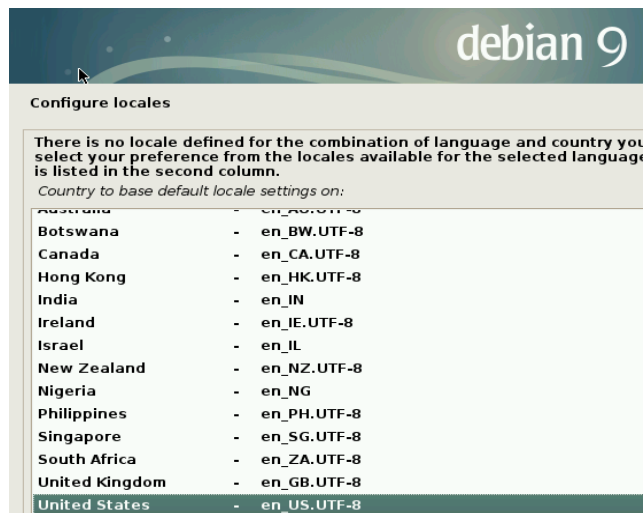


安装Linux虚拟机

4. 选择“other→Asia→China”作为所在地区

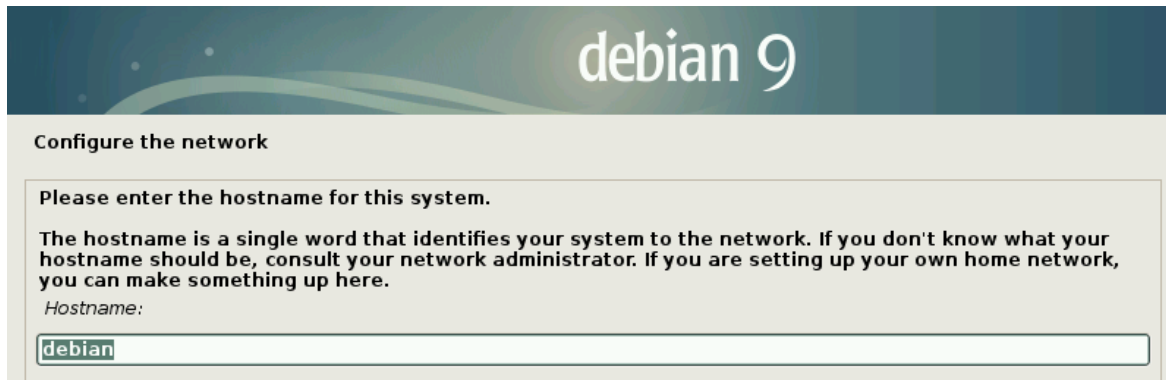


5. 选择locale为“United States – en_US.UTF-8”，选择keymap为“American English”



安装Linux虚拟机

6. 网络主机名（Hostname）和域名（Domain name）可任意设置：



debian 9

Configure the network

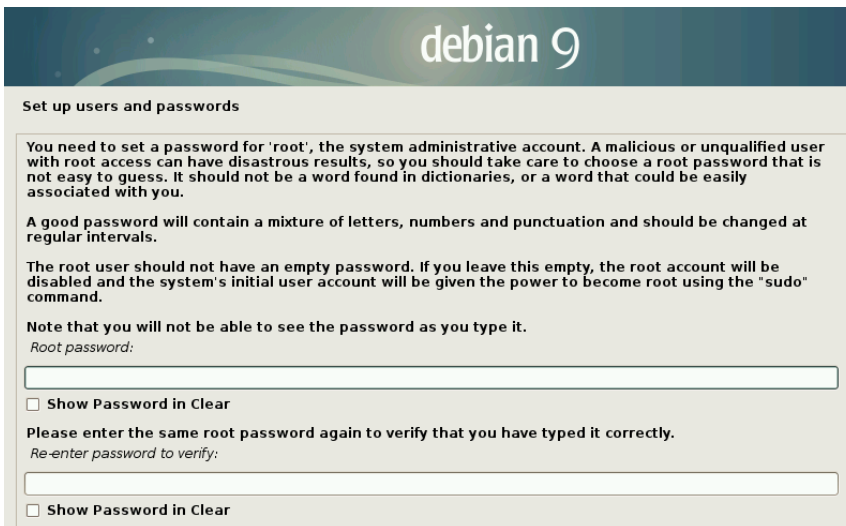
Please enter the hostname for this system.

The hostname is a single word that identifies your system to the network. If you don't know what your hostname should be, consult your network administrator. If you are setting up your own home network, you can make something up here.

Hostname:

debian

7. 设置合适的系统管理员（root）的登录密码，以及首个系统普通用户的用户名和登录密码



debian 9

Set up users and passwords

You need to set a password for 'root', the system administrative account. A malicious or unqualified user with root access can have disastrous results, so you should take care to choose a root password that is not easy to guess. It should not be a word found in dictionaries, or a word that could be easily associated with you.

A good password will contain a mixture of letters, numbers and punctuation and should be changed at regular intervals.

The root user should not have an empty password. If you leave this empty, the root account will be disabled and the system's initial user account will be given the power to become root using the "sudo" command.

Note that you will not be able to see the password as you type it.

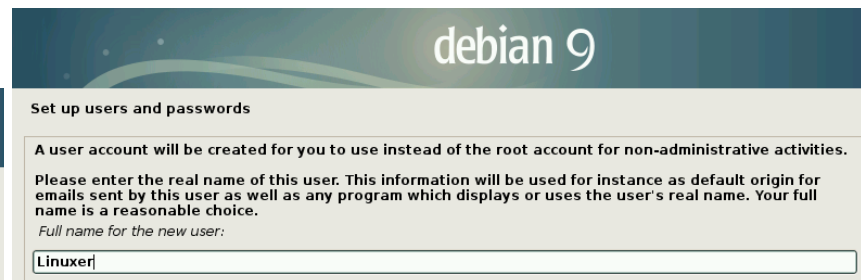
Root password:

☐ Show Password in Clear

Please enter the same root password again to verify that you have typed it correctly.

Re-enter password to verify:

☐ Show Password in Clear



debian 9

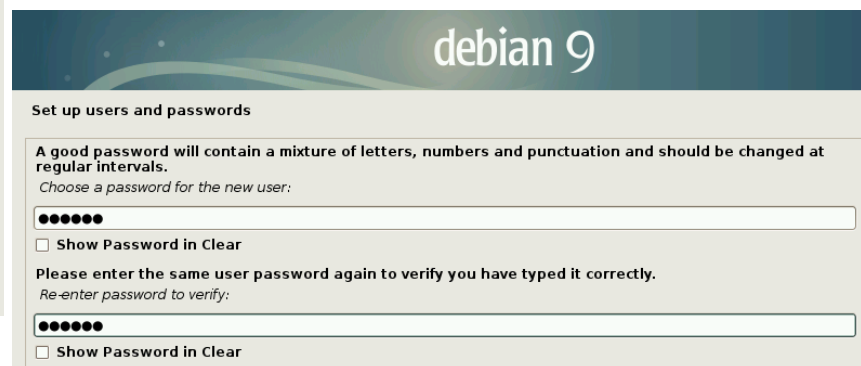
Set up users and passwords

A user account will be created for you to use instead of the root account for non-administrative activities.

Please enter the real name of this user. This information will be used for instance as default origin for emails sent by this user as well as any program which displays or uses the user's real name. Your full name is a reasonable choice.

Full name for the new user:

Linuxer



debian 9

Set up users and passwords

A good password will contain a mixture of letters, numbers and punctuation and should be changed at regular intervals.

Choose a password for the new user:

••••••

☐ Show Password in Clear

Please enter the same user password again to verify you have typed it correctly.

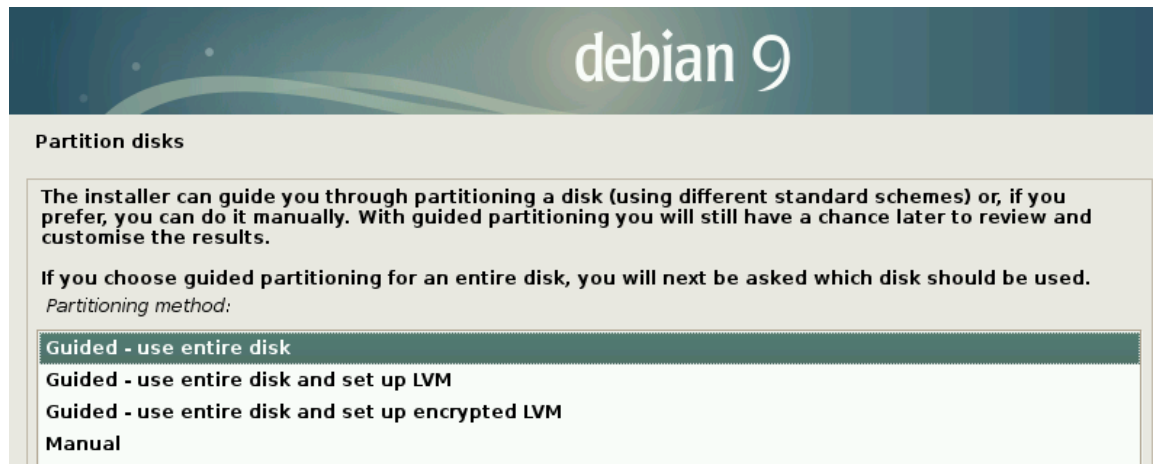
Re-enter password to verify:

••••••

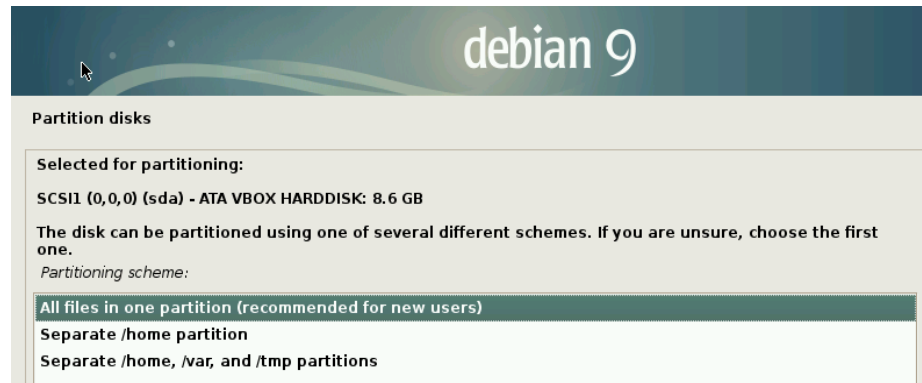
☐ Show Password in Clear

安装Linux虚拟机

8. 在虚拟硬盘上创建安装分区，可选择“**Guided – use entire disk**”：

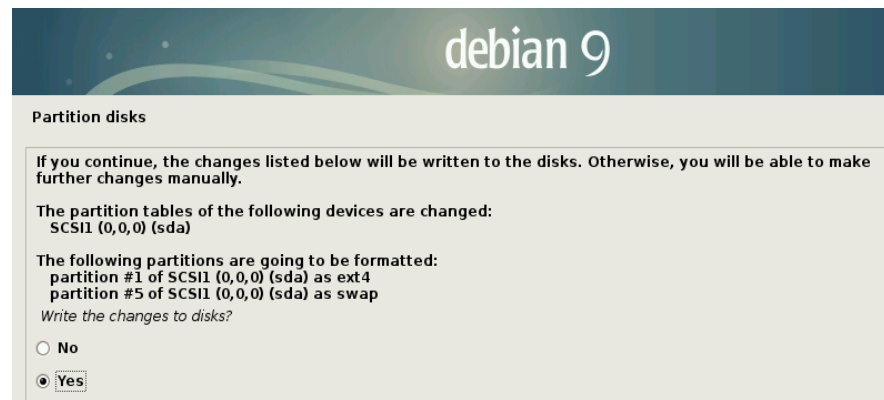
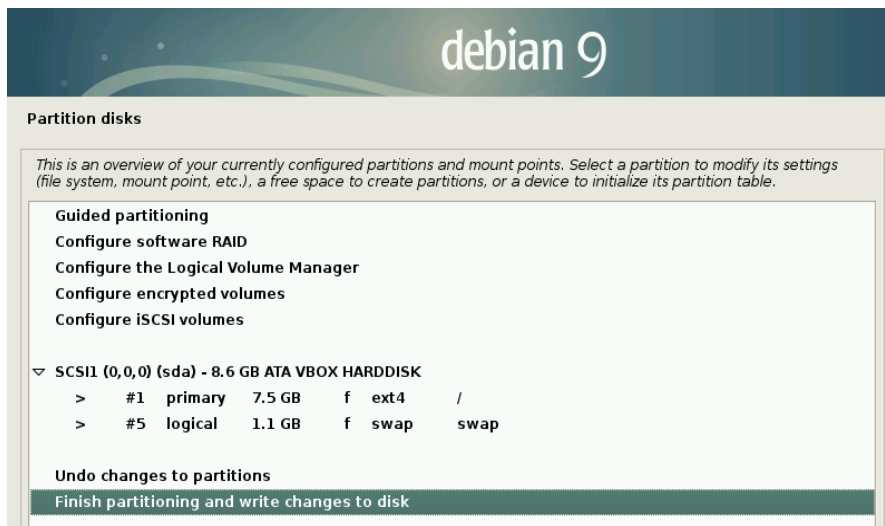


9. 选择虚拟机中配置的唯一虚拟硬盘和默认的“**All files in one partition**”分区方案（熟悉的话也可在前步选择进行手工**Manual**分区）

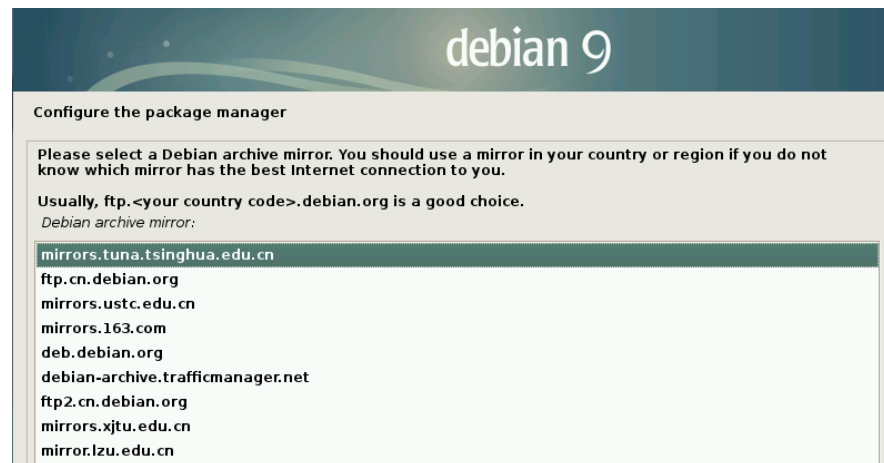
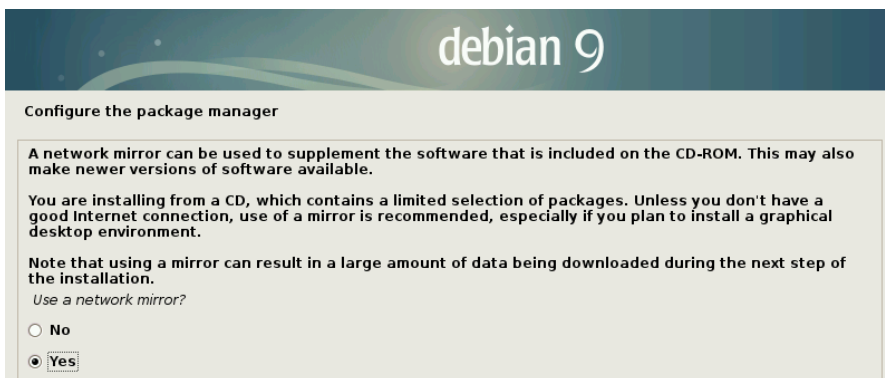


安装Linux虚拟机

10. 接受并确认默认的分区方案：



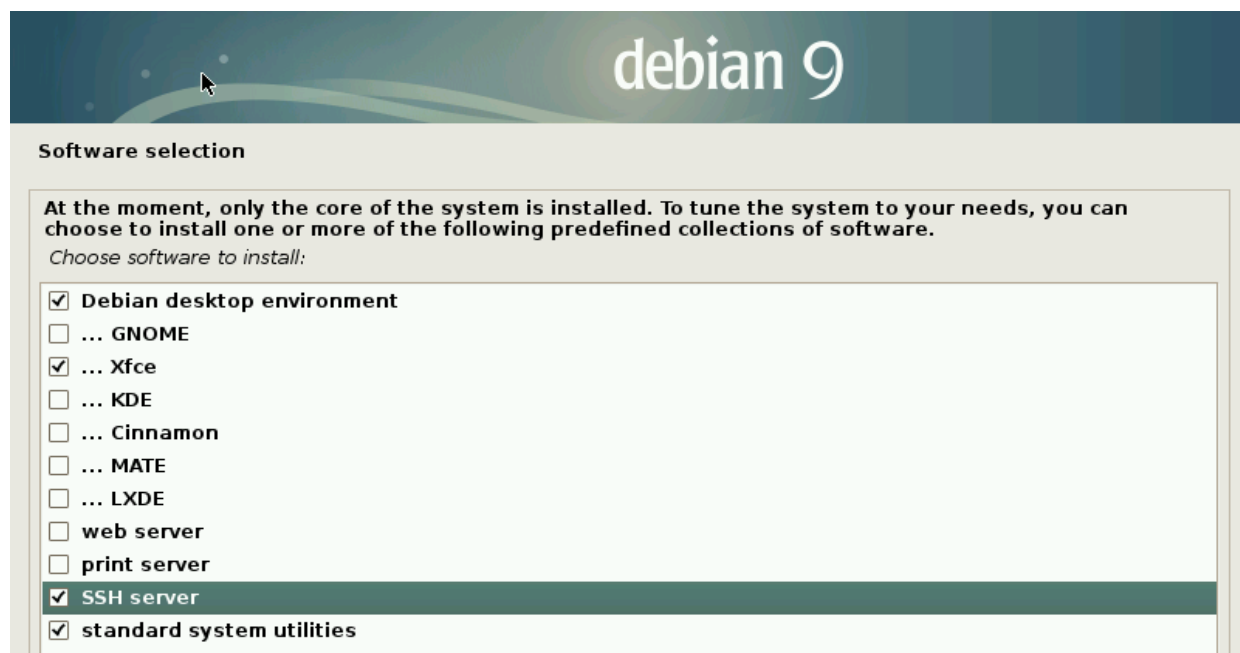
11. 如果有经常的互联网连接可用，可设置使用网络镜像（**network mirror**），以在需要安装缺少的软件包时，可去邻近的软件包发行镜像站点下载：



安装Linux虚拟机

12. 选择需安装的软件包——其中：

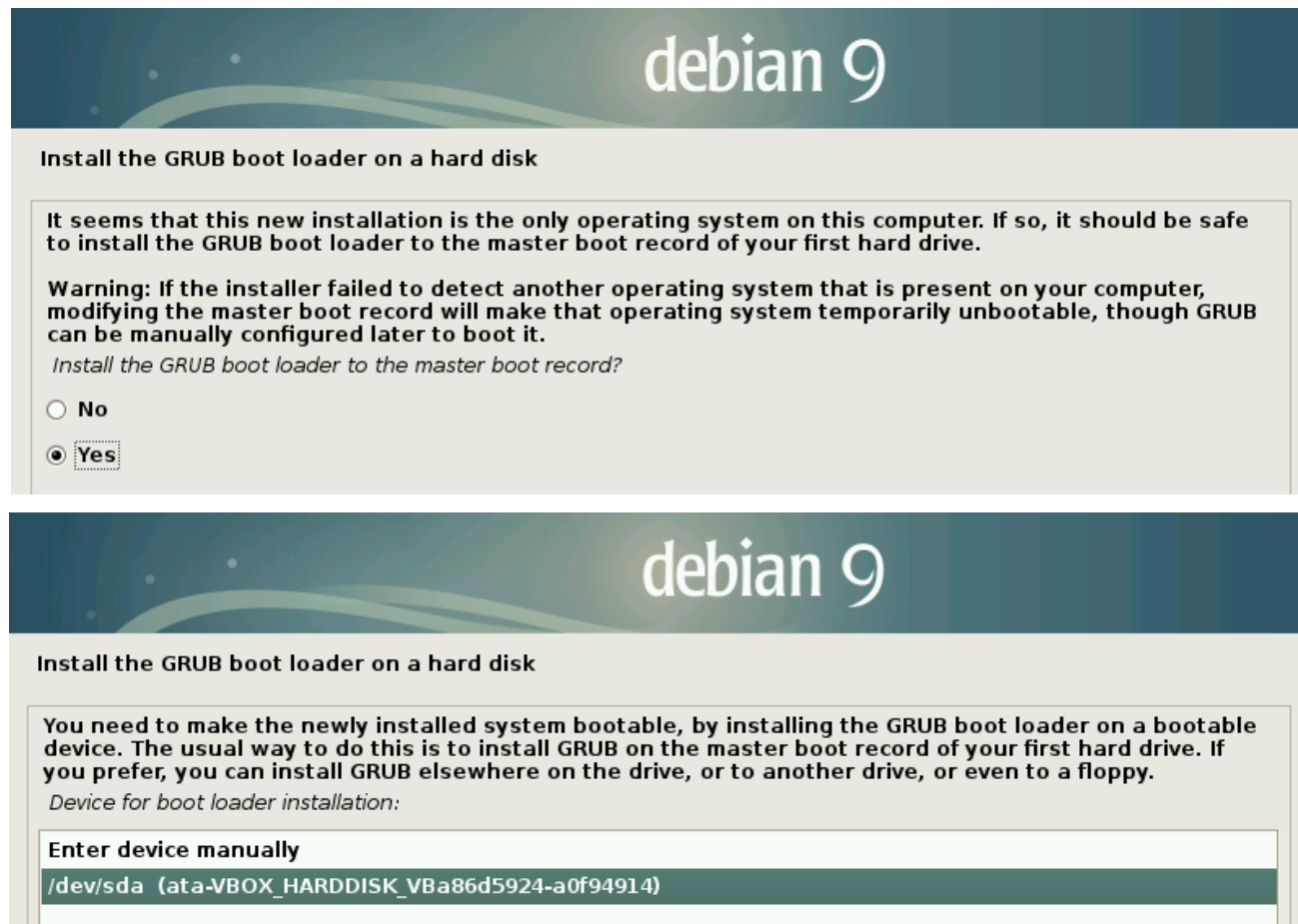
- ✓ **standard system utilities**（提供了基本的文本命令行操作界面和基本的系统/应用软件）和**SSH server**（用于从Host操作系统等其它系统远程访问和操作Linux虚拟机）需要安装
- ✓ **Debian desktop environment**及其具体类型如**Xfce**选装（提供了图形化Linux使用环境）



安装Linux虚拟机

13. 安装Linux启动装载器：GRUB boot loader

- ✓ 可如下缺省安装至虚拟硬盘（`/dev/sda`）的主引导记录（**master boot record, MBR**）中



14. 结束安装，重启并开始使用Linux虚拟机

3. 实验环境安装和使用

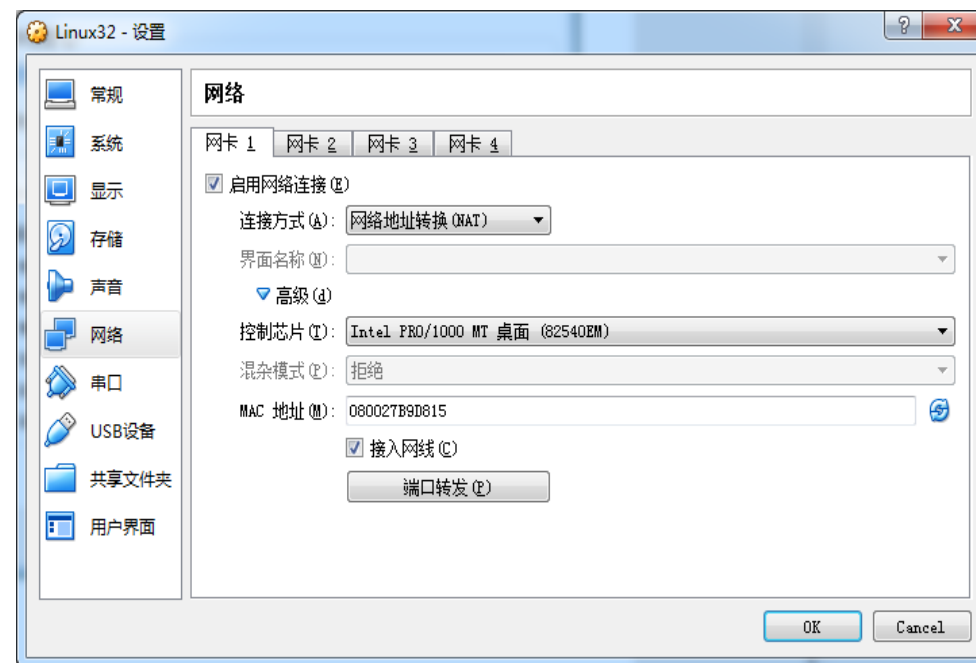
配置Linux虚拟机使用SSH端口转发

- 在VirtualBox管理器界面中选中Linux虚拟机，选择“设置”按钮



➤ 设置SSH端口转发【可选配置】

- 在左栏中选择“网络”
- 如右图所示，点选“高级”，再点选“端口转发”
- 在“端口转发规则”对话框中，点选右侧加号按钮，在新增的规则行中按下图进行参数设置

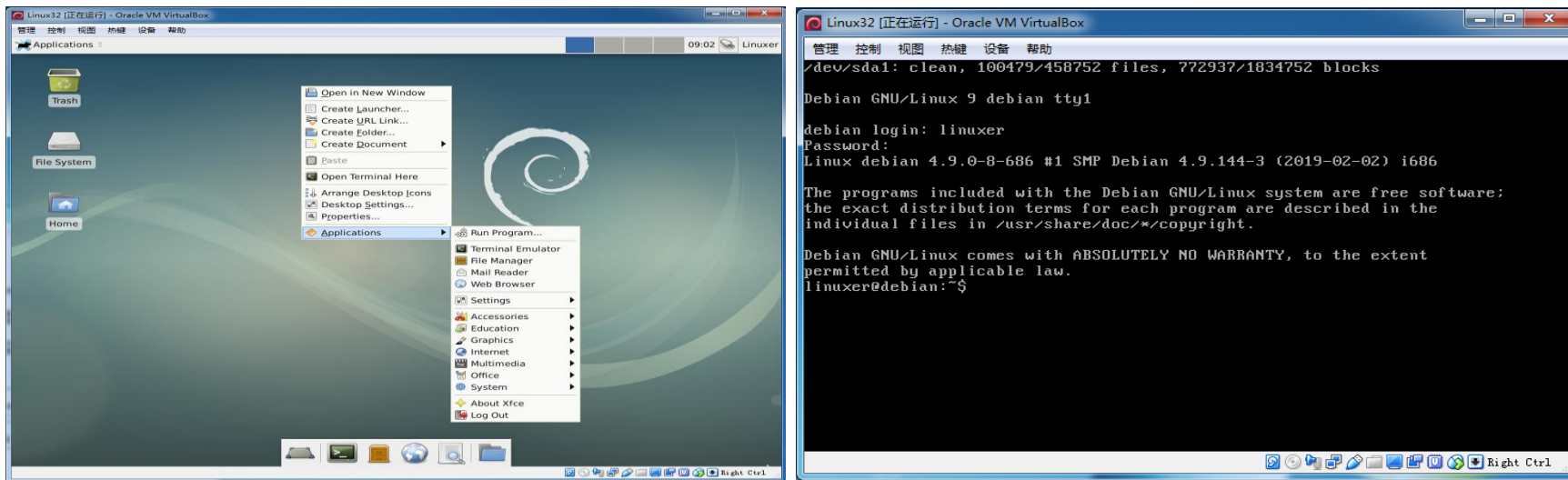


如上设置SSH端口转发，将方便从Host操作系统访问Linux虚拟机的命令行界面以及在两者之间传输文件



使用Linux虚拟机

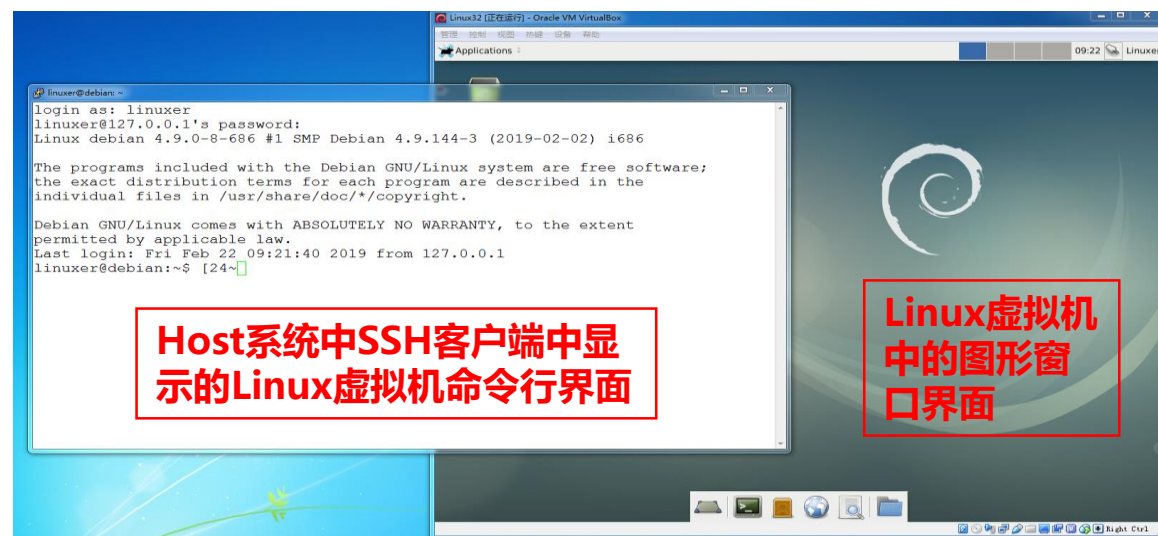
- 方式一：直接使用虚拟机窗口中的Linux用户界面



- 建议在Linux虚拟机中安装VirtualBox Guest Additions增强功能以更方便地使用虚拟机（参考VirtualBox 用户手册中相应说明），例如对虚拟机窗口缩放的支持

使用Linux虚拟机

- 方式二：使用SSH软件在Host操作系统（如Windows/Mac）中访问和操作Linux虚拟机系统
 - SSH（Secure SHell）软件提供了对网络上一个运行有SSH服务的Linux系统的远程访问功能，包括在远程Linux系统上执行命令、远程与本地系统之间输入/输出的转发、数据传送等
 - Host系统与Linux虚拟机系统位于VirtualBox创建的同一虚拟网络中（虚拟机默认网络设置为NAT）
 - 需要在Linux虚拟机中安装openssh-server（和openssh-sftp-server）软件包（或在安装Linux系统时选择SSH Server组件）
 - 在Host系统中安装使用SSH客户端软件（例如PuTTY）远程连接到Linux虚拟机系统——SSH客户端中输入的命令将在虚拟机上执行，命令输出将返回至SSH客户端显示



使用Linux虚拟机

- 启动Linux虚拟机：使用VirtualBox管理器中的“启动”按钮或菜单
 - 在启动完成后出现的文本命令行或图形桌面登录界面中，输入安装时设置的用户名和口令以登录进系统
- 关闭Linux虚拟机：
 - 以root权限在命令行中执行“**poweroff**”（关机）、“**reboot**”（重启）等命令
 - 使用图形桌面环境中的相应菜单
- 常用命令行工具或命令（**shell**中输入执行，命令名区分大小写）：
 - **cd, ls, cp, rm, mkdir, rmdir**: 文件、目录操作命令
 - **cat, more, less, vi / vim / nano**: 查看和编辑文本文件内容
 - **tar, zip**: 文件/目录打包和压缩工具
 - **gcc, gdb, objdump ...**: 程序开发和调试工具
 - **df, du**: 查看空闲或已用的硬盘分区（已挂载文件系统）空间
 -

使用Linux虚拟机

Linux中系统级的管理和控制命令（例如修改系统配置、安装软件等）在执行时往往需要执行者具有管理员**root**权限。可通过下述两种途径获得相应权限：

- **su命令**：从当前用户登录环境和权限切换到管理员**root**（此时**su**命令后不加用户名参数）或其它指定用户（**su**命令后跟目标用户名作为参数）的环境与权限
 - 使用**exit**命令可退出当前环境，返回到调用**su**命令之前的用户环境
- **sudo命令**：允许在当前普通用户环境中以管理员**root** 或其它指定用户的身份及其权限执行指定的命令
 - 例如：**sudo poweroff**（执行**poweroff**命令需要管理员权限）
 - 需先安装**sudo**软件包
 - 配置文件：**/etc/sudoers**，列出允许使用**sudo**命令的用户及允许其执行的命令集合
 - **sudo**命令执行中（以目标用户权限执行相应命令前），需先输入当前用户口令以验证身份并判断当前用户是否具有相应**sudo**权限

使用Linux虚拟机

安装软件——使用Debian软件包管理系统APT

- **APT软件包发布源**：包含有**Debian**软件包集合的网站（或本地）目录
 - 可使用**APT**工具从系统配置中指定的**APT**软件包发布源，在线下载（需有互联网连接）安装指定名称的软件包
 - **APT**软件包发布源配置文件：**/etc/apt/sources.list**，其中列出源站点域名及相应目录、软件包集合类别，可手工修改以使用邻近的源
 - 示例：**deb http://ftp.cn.debian.org/debian/ buster main contrib non-free**
 - 其中，<http://ftp.cn.debian.org/debian/>指出了发布源的网站域名、路径和访问所用网络协议；**stretch**指出**Debian**发行版本，**main**、**contrib**、**non-free**指出了不同的软件包集合类别
 - **Debian**官方的软件源/镜像站点列表可从下列地址获得
<https://www.debian.org/mirror/list>
 - 修改了上述**APT**软件包发布源配置文件**sources.list**后，需以管理员权限运行命令“**apt-get update**”，以从新设置的软件包发布源中下载、更新本地的软件包列表

使用Linux虚拟机

安装软件——使用Debian软件包管理系统APT

- 使用**APT**工具安装软件包
 - 命令: **apt-get install <软件包名>**
 - 例如: **apt-get install sudo**
 - **Synaptic Package Manager**工具: 图形化软件包管理工具
 - **aptitude**工具: 全屏文本界面软件包管理工具

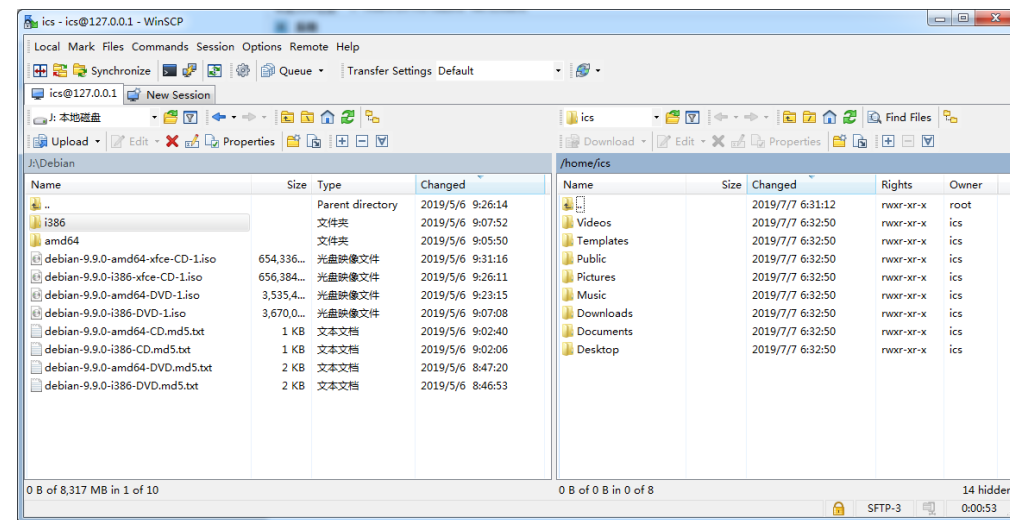
安装实验相关软件包

- 使用**APT**命令集安装实验所需的下列软件包
 - **build-essential**: 包含**gcc**, **make**, **ld**等开发工具
 - **vim** 或 **gedit**: 文本和源代码编辑器
 - **gdb**: 交互式程序调试工具
 - **hexedit**: 二进制文件编辑器
 - **git**: 源代码版本控制工具
- 安装方法:
 - ✓ 在命令行窗口中使用“**su**”命令切换为**root**用户（或）
apt-get install <软件包名>
例如: **apt-get install build-essential**
 - ✓ 或者，在当前普通用户（需具有**sudo**权限——具体介绍请查阅相关文档）登录环境中，通过“**sudo**”命令以使用**root**权限：
sudo apt-get install <软件包名>

使用Linux虚拟机

在Linux虚拟机与Host操作系统之间拷贝文件

- 使用SSH软件中的SCP/SFTP组件
 - 在Linux虚拟机中安装openssh-server、openssh-sftp-server软件包
 - 在Host系统中安装、使用SCP/SFTP客户端
 - 命令行客户端程序
 - 例如：PuTTY中的PSCP
 - 图形界面客户端程序
 - 例如：Windows上运行的WinSCP

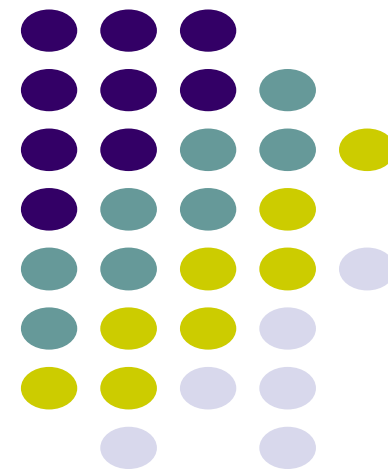


- 使用VirtualBox提供的Shared Folder功能
 - 需安装VirtualBox的” Guest Additions”增强功能（需编译内核模块，参见VirtualBox用户手册中相应说明）

结束

《计算机系统基础（四）：编程与调试实践》

基本实验工具的使用



基本实验工具的使用

基本gcc命令的使用

基本objdump命令的使用

基本gdb命令的使用

基本gcc命令的使用

基本gcc命令的使用

GCC是一套由GNU项目开发的编程语言编译器，可处理C语言、C++、Fortran、Pascal、Objective-C、Java等等。GCC通常是跨平台软件的编译器首选。gcc是GCC套件中的编译驱动程序名。

准备工作

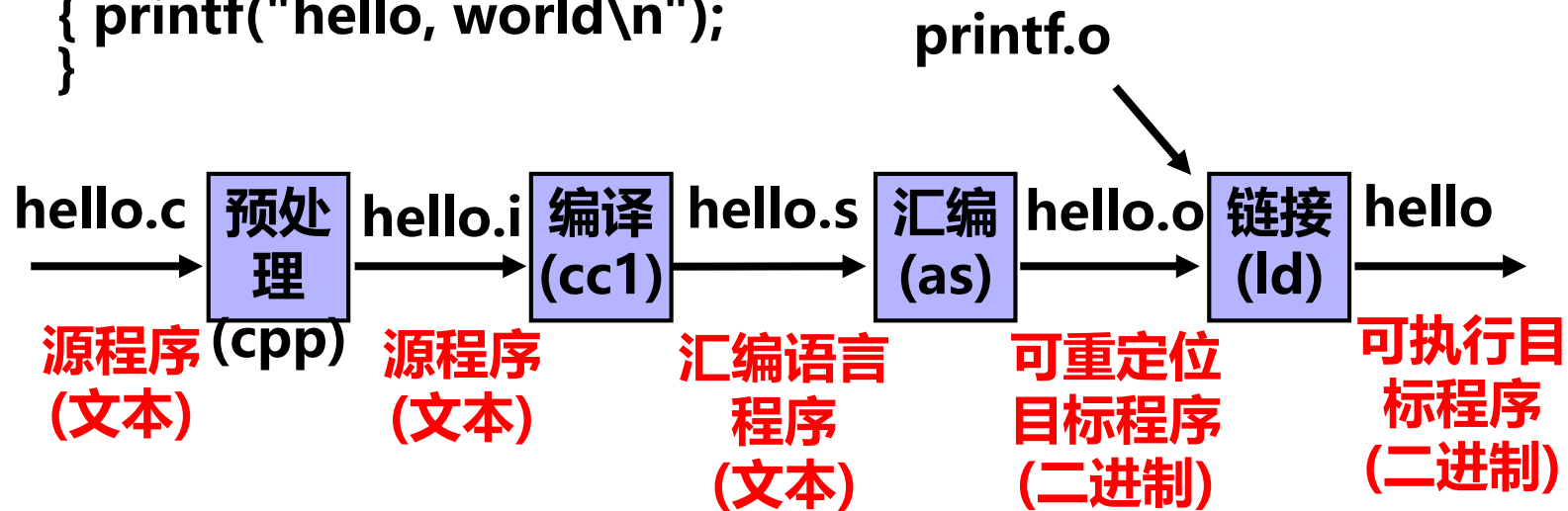
- 1. 具备对Linux系统的了解，掌握Linux的常用命令**
- 2. 若计算机是x86-64位系统，为了编译成IA-32指令集，
则请先运行下列命令：**

```
sudo apt-get install build-essential module-assistant
```

```
sudo apt-get install gcc-multilib g++-multilib
```

基本gcc命令的使用

```
#include "stdio.h"
main()
{ printf("hello, world\n");
}
```



`gcc -E hello.c -o hello.i`

`gcc -S hello.i -o hello.s`

`gcc -c hello.s -o hello.o`

`gcc hello.o -o hello`

`gcc hello.c -o hello //将hello.c直接编译成可执行目标文件hello`

基本objdump命令的使用

```
#include "stdio.h"
void main( )                                gdbtest.c
{ int x=3, y=5, z ;
  z=x+y;
  printf("z=%d\n",z);
  return;
}
```

```
gcc -E -g -m32 gdbtest.c -o gdbtest.i
gcc -S -g -m32 gdbtest.i -o gdbtest.s
gcc -c -g -m32 gdbtest.s -o gdbtest.o
gcc -O0 -m32 -g gdbtest.c -o gdbtest
```

//gdbtest.o可重定位目标文件、gdbtest可执行目标文件

```
objdump -S gdbtest.o>gdbtesto.txt
```

```
objdump -S gdbtest>gdbtest.txt
```

//反汇编，-S 保留C语句，> 保存到文件

基本gdb命令的使用

GDB调试的基本步骤只有6步

步1: 启动GDB调式工具，加载要执行的目标文件

(1) **gdb 可执行目标文件** //启动GDB调试工具，并加载程序

(2) **gdb** //启动GDB调试工具

file 可执行目标文件 //加载程序

步2: 设置断点

break main //在main函数的入口处设置断点

break gdbtest.c:3 //源程序gdbtest.c的第3行处设置断点

步3: 启动程序运行

run //程序会在断点处停下

步4: 查看程序运行时的当前状态

步5: 继续执行下一条指令或语句

si //执行一条机器指令

s //执行一条c语句

步6: 退出调试

quit

步4和步5根据自己的需要不断地、交替地执行，达到对程序执行过程的跟踪。

基本gdb命令的使用

步4: 查看程序运行时的当前状态

程序的当前断点位置: **i r eip (或 i r)**

通用寄存器的内容: **i r eax ebx ecx edx (或 i r)**

存储器的单元内容: **x/8xb 0xffffd2bc**
x/2xw 0xffffd2bc

栈帧信息

当前栈帧范围: **i r esp ebp** //esp栈顶指针和ebp栈底指针

当前栈帧字节数: $y = R[ebp] - R[esp] + 4$ //不是命令, 是计算方法

显示当前栈帧内容: **x/yxb \$esp**
x/zxw \$esp //z=y/4

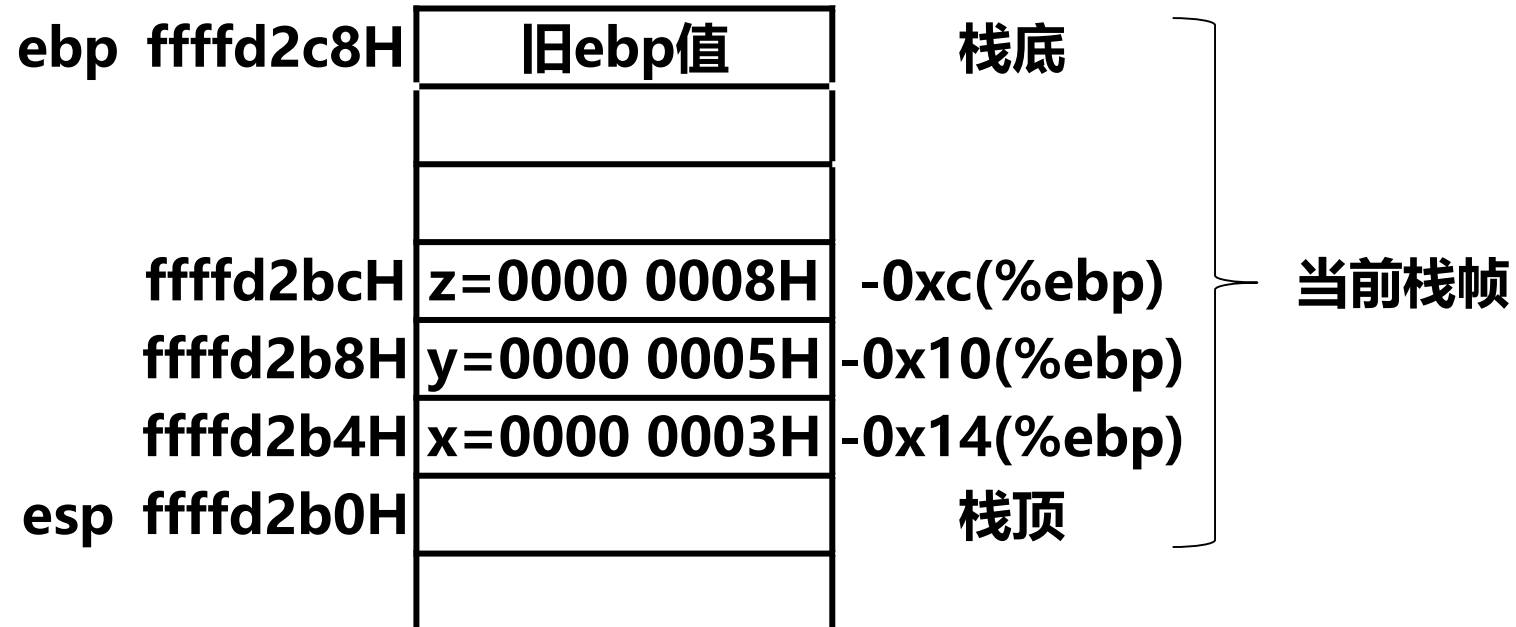
基本gdb命令的使用-举例

C源程序	可执行目标文件	反汇编文件
gdbtest.c	gdbtest	gdbtest.txt

步1: **gdb gdbtest** //启动GDB调式工具, 进入gdbtest的调试环境
步2: **break main** //在main函数处设置断点
 break gdbtest.c:3 //在gdbtest.c的第3行设置断点
步3: **run** //启动运行程序
步4: **si (或s)** //执行一条指令 (或c语句)
步5: **i r** //查看各寄存器的内容
步6: **i r eip** //查看eip寄存器的内容
步7: **i r esp ebp** //查看esp和ebp寄存器的内容
步8: **x/yxb \$esp** //按字节显示当前栈帧内容
 // $y = R[ebp] - R[esp] + 4$
 或 **x/zxw \$esp** //按4字节显示当前栈帧内容
 // $z = (R[ebp] - R[esp] + 4) / 4$
 重复执行步4~步8, 观察程序运行时的各种数据变化。
步9: **quit** //退出调试

基本gdb命令的使用

从2到8的自然数有几个? 2 3 4 5 6 7 8 答案:
8-2=6 8-2+1=7 7个



当前栈帧的字节数:

$$\begin{aligned} R[ebp] - R[esp] + 4 &= \text{fffd2c8H} - \text{fffd2b0H} + 4 \\ &= 28 \text{ 个字节} \end{aligned}$$



谢谢！