

前言：任何版本不是越新越好，而是越合适越好！

【理论准备】

1、部分常用的终端命令（已用红色和蓝色标注）

cd /绝对路径名：根据绝对路径名进入对应的目录，无视当前目录的位置

cd 相对路径名：根据相对路径名进入当前目录中的相应目录

pwd：查看当前目录的路径名

ls：查看当前目录的所有文件和目录

ls -l：查看当前目录的所有文件和目录及其详细信息（如最后编辑日期、大小）

cp 文件 A 的路径名 目录 B 的路径名：把文件 A 拷贝到目录 B 处

mv 文件 A 的路径名 目录 B 的路径名：把文件 A 移动到目录 B 处

tar 解压缩指令 待解压文件 A 的路径名：解压文件 A 到 A 所在的目录（这里解压缩指令取决于压缩包的类型，比如.tar.xz 类型的压缩包可以用 xfv 指令）

uname -a：查看当前系统名、节点名称、操作系统的发行版号、版本、运行系统 ID 号

uname -r：查看当前系统版本

sudo passwd root：为 root 用户（权限最高）设置密码

su root：进入 root 用户模式，需要输入设置的密码

sudo chmod 666 文本 A 的路径名：把文本文件 A 设置为可读可写文件（666 表示可读写）

sudo apt-get install XXX：下载并安装文件 XXX（XXX 为待下载的文件名，可以多个文件名以空格间隔，来同时安装多个文件）

sudo apt-get update：下载所有可更新的软件安装包

sudo apt-get upgrade：安装所有已下载且可更新的软件安装包

sudo reboot：重启

2、关于路径名

以“/”开头的路径名是绝对路径名，是从根目录开始读取的；不以“/”开头的路径名是相对路径名，是从当前目录开始读取的。上面 1 中提到的所有“路径名”，既可以是绝对路径，也可以是相对路径，取决于自己的选择。

路径名中一般没有空格，如果有空格，要把带空格的文件名用双引号括起来，比如 **cd “VMware Tools”**；或者在空格前加一个\，比如 **cd My\ Document**。

3、关于终端里输入密码的问题

终端有时需要输入密码，键盘键入密码时不会有任何显示，光标仍在原地跳动，这并不是你的键盘有问题，而是 Linux 的安全机制，实际密码已经被输入，输入完成后按回车即可，如果输错，有两次重新输入的机会。

4、关于权限

根目录文件以及一些特殊文件，都是需要 root 用户的权限才能进行访问和修改的，因此大部分与之有关的操作，都必须给予 root 权限，否则会报错 Permission Denied。这里有两

种方式给予权限：

- ①在每一个相关的命令前加上 **sudo**，比如上面给出的最后 5 个命令。此外，在终端第一次使用 **sudo** 需要验证密码，输入当前用户的密码即可，此后就不再需要。
- ②事先通 **su root** 进入 root 用户模式。

注意，如果你进入了 root 用户模式，那么本文提到的所有 **sudo** 都不再需要了，比如重启的命令此时是 **reboot**，而不再是 **sudo reboot**。

【实验环境】

系统版本：Ubuntu-18.04 LTS 版本

系统自带的内核版本：linux-5.4.0-73-generic

选择安装的内核版本：linux-5.4.115

虚拟机版本：VMware Workstation 15.5 Pro

【实验准备】

- 1、下载 Ubuntu 系统的镜像文件，本文用的是 18.04 LTS（64 位）版本：

<https://ubuntu.com/download/desktop>

- 2、下载内核源代码：

<https://www.kernel.org/> 或 <https://mirrors.edge.kernel.org/pub/linux/kernel/>

注 1：可用 `uname -r` 或 `cat /proc/version` 命令查看当前内核版本

注 2：尽量选用版本相同或接近的源码，不要太新或太旧的版本

注 3：下载的源代码为形如 `linux-5.4.115.tar.xz` 的压缩包

或者可以手动下载，这个我没用过，不确定是否可行。需在终端键入命令：

apt-cache search linux-source // 搜索内核版本

apt-get install linux-source-5.4.115 // 下载特定版本内核源码

（警告注意）Linux 之父 Linus Torvalds 警告：请勿使用 Linux 5.12-Rc1 内核

<https://baijiahao.baidu.com/s?id=1693497919240019327&wfr=spider&for=pc>

（补充学习）Linux 内核版本号命名的规则

https://blog.csdn.net/dj0379/article/details/50853114/?utm_term=linux%E5%86%85%E6%A0%B8%E7%89%88%E6%9C%AC%E9%80%89%E6%8B%A9&utm_medium=distribute.pc_aggpage_search_result.none-task-blog-2~all~sobaiduweb~default-3-50853114&spm=3001.4430

- 3、下载实时补丁（可选）：

<https://mirrors.tuna.tsinghua.edu.cn/kernel/projects/rt/>

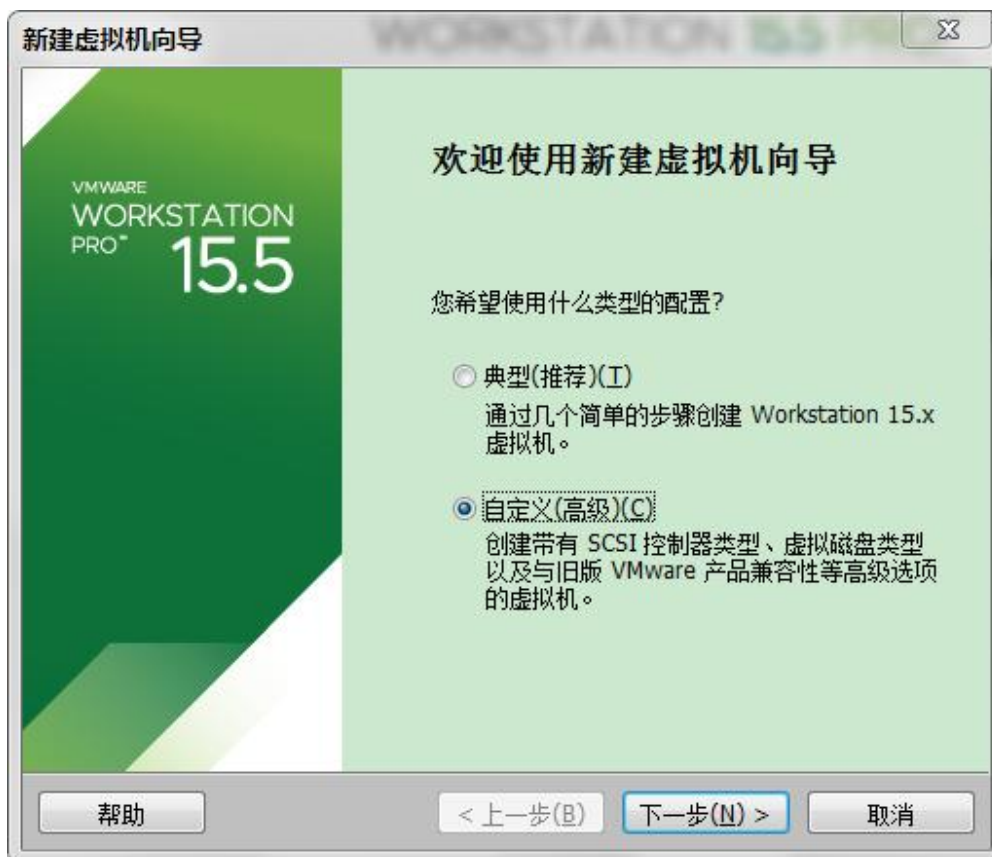
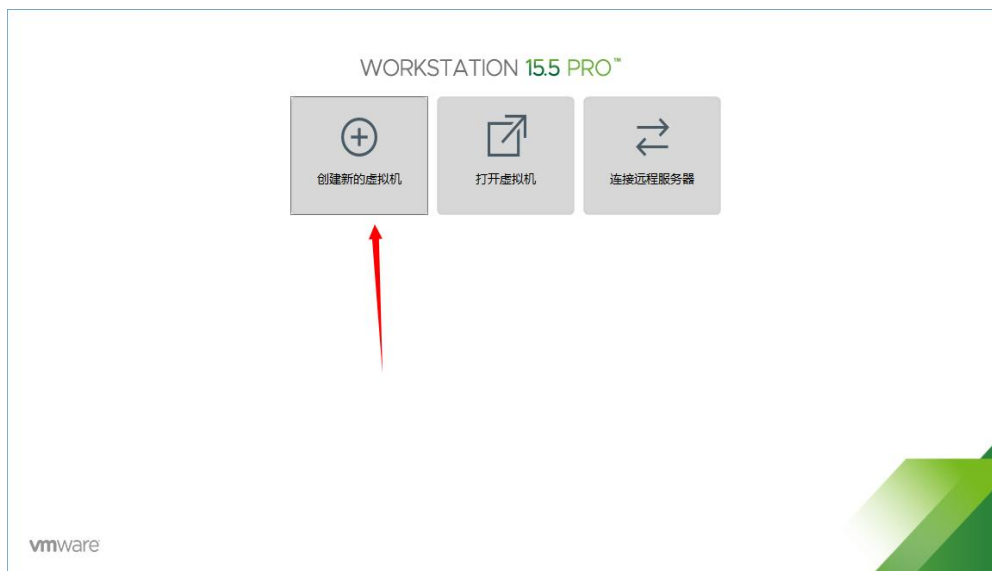
注 1：实时补丁的作用是可以实时更新内核，在启动这个补丁之后，用户就不需要每次修改内核后再重启了。这个补丁其实并不必要，如果想要安装，要选取和内核版本一致的补丁。

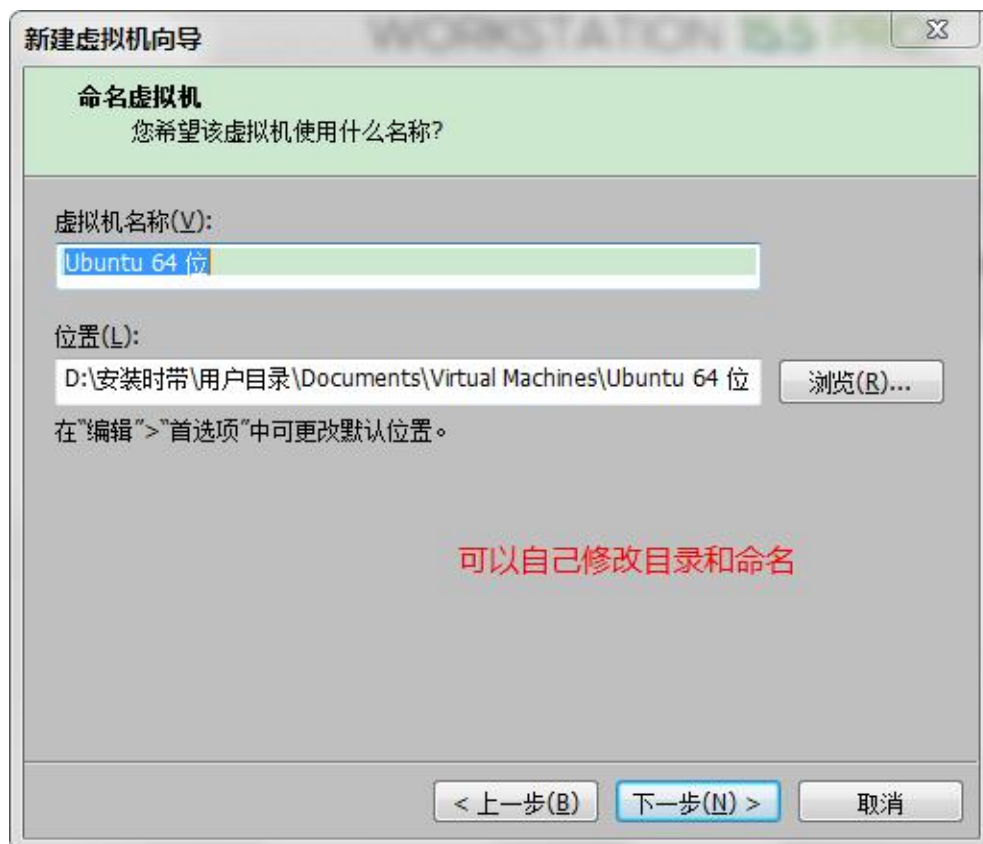
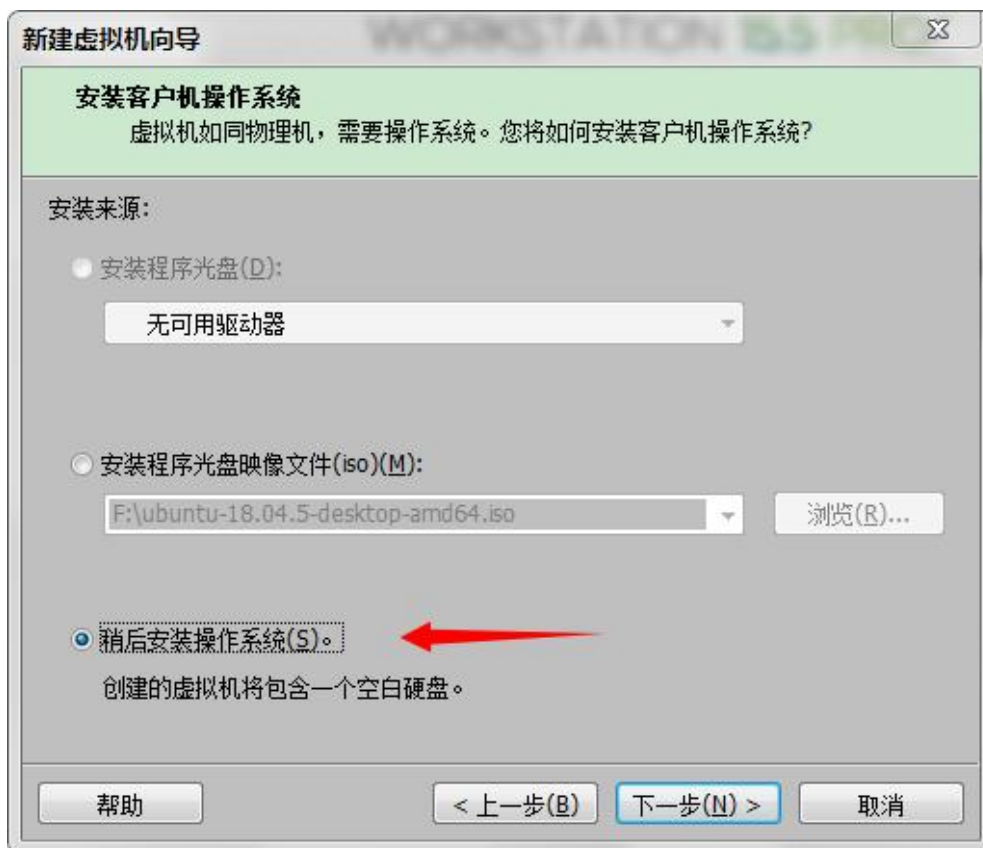
注 2：下载的补丁为形如 `patch-5.4.115-rt57.patch.xz` 的压缩包。

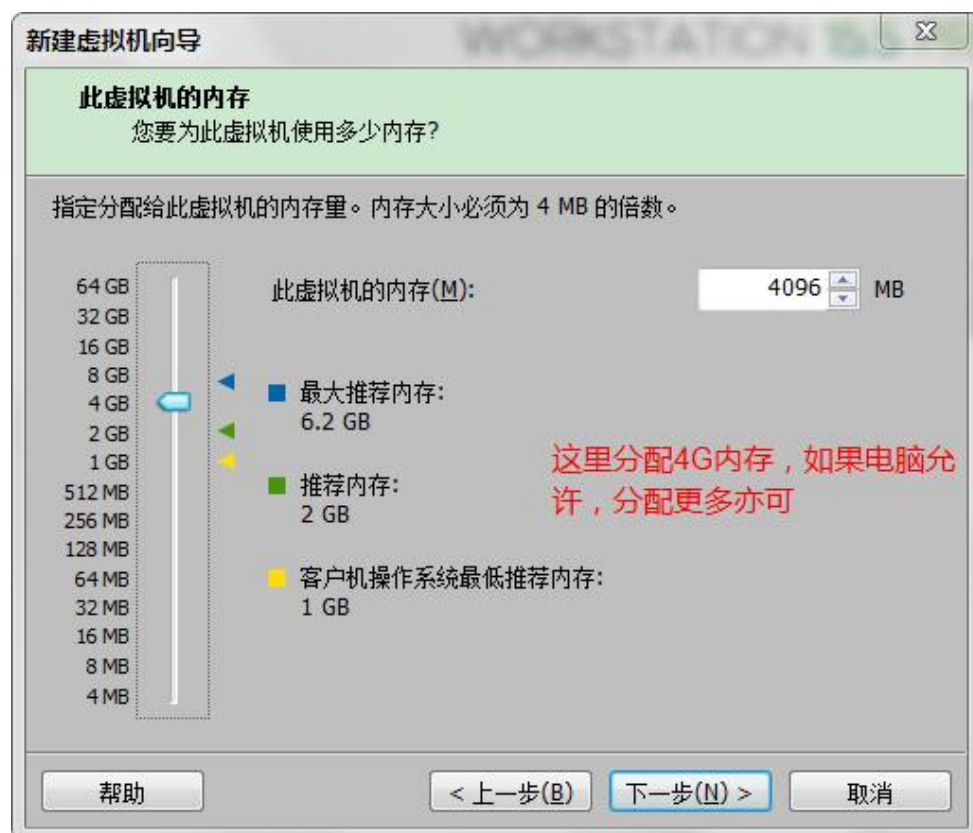
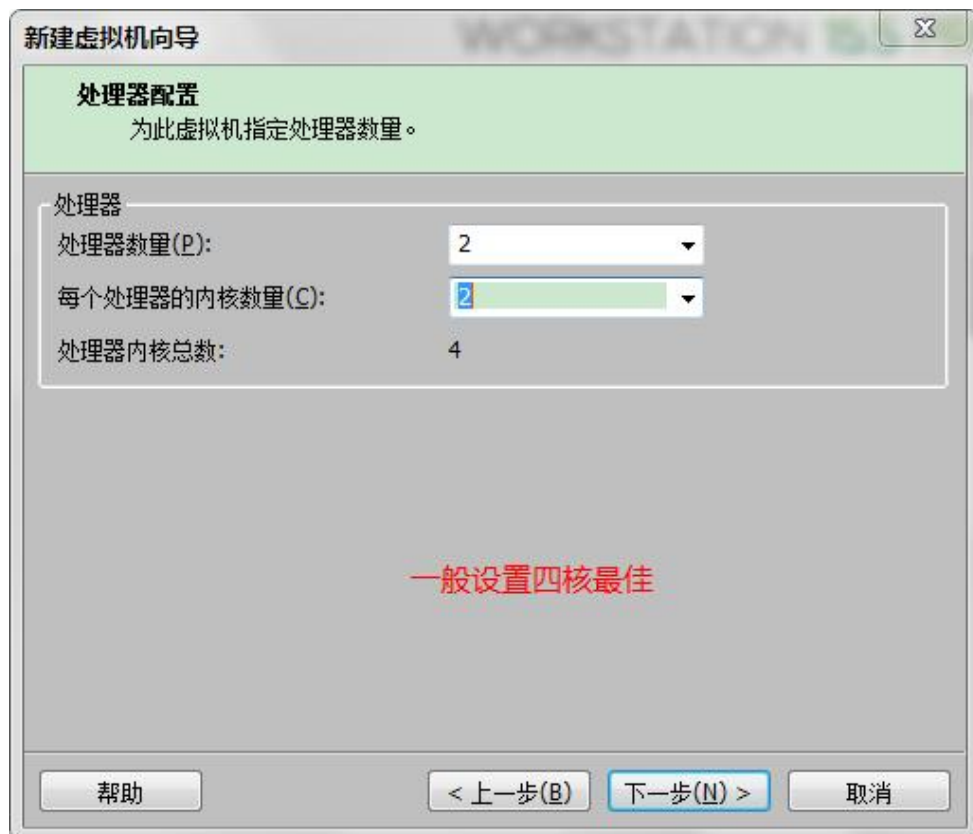
【实验过程】

一、在 VMware 虚拟机上安装对应的 Ubuntu 系统

1、创建新的虚拟机（按下图。如果下图没有显示的步骤，直接按照默认即可）

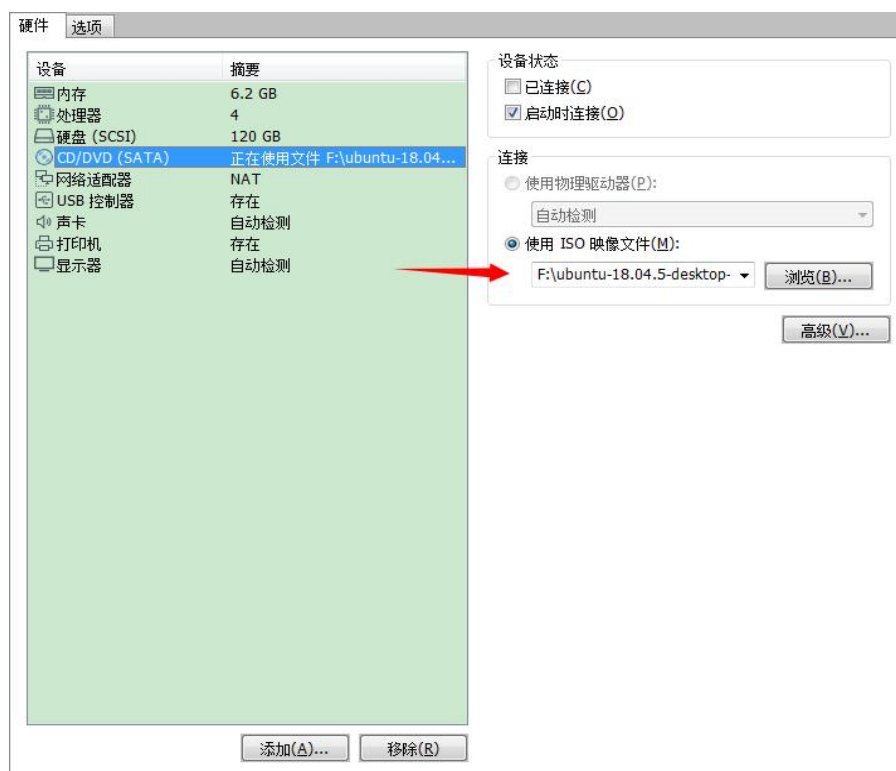




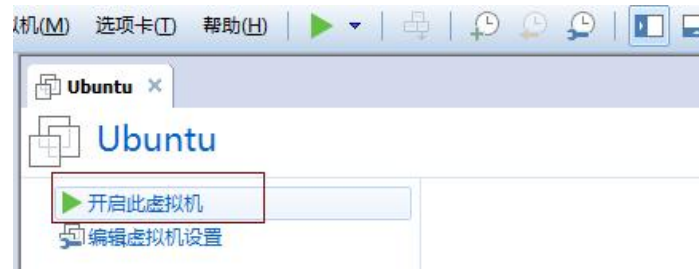




完成之后，点击“编辑虚拟机设置”，在 CD/DVD 中设置使用 ISO 映像文件，读取之前下载的 Ubuntu 系统的 iso 文件（下图是我自己的虚拟机，和上面的教程有所差异），如下图：



2、开启运行虚拟机。



二、更新必要的软件、工具

1、更新软件（我并不确定这一步是否必要，但做了肯定没错）

开机后进入安装流程，进行 Install。可以设置语言为中文，安装的过程比较简单，这里略。安装完成后重启，重启后会提示更新软件，这里尽量选择下载更新。更新完成后先别急着重启，顺便做完第 2 步，安装完 VMware Tools 再重启，这样就不必重启三次了。

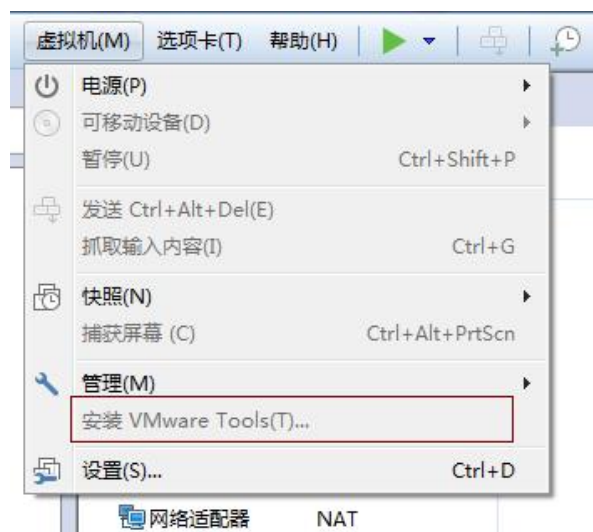
2、安装 VMware Tools

VMware Tools 是 VMware 提供的工具包，具有非常实用的功能。对于我们的实验，最大的两个用处在于：

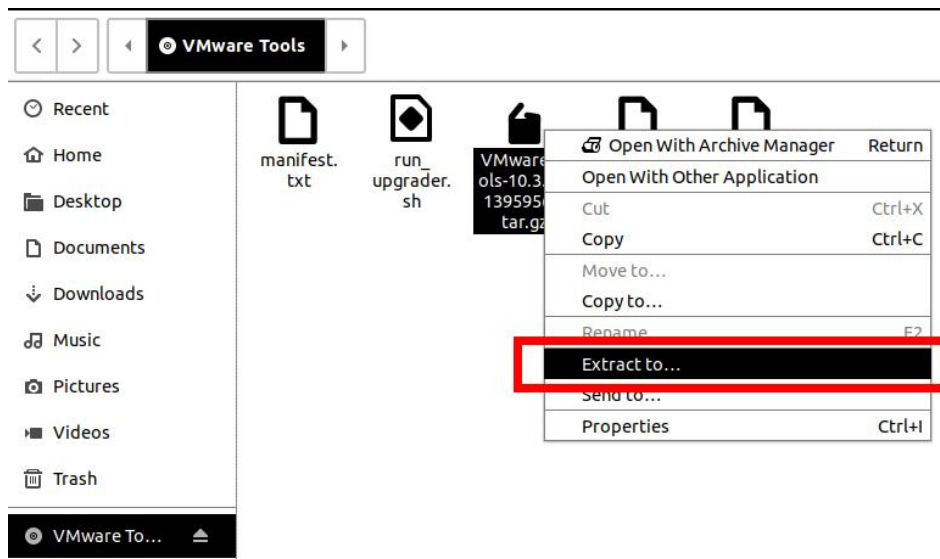
- ①增加了拖拽实现文件交换的功能（即可以把主机中的文件通过拖拽，添加到虚拟机中）；
- ②增加了共享粘贴板的功能（即在主机复制或剪切的文字、文件可以粘贴到虚拟机里，反之亦然。非常方便在报错的时候百度一下错误原因）。

另外，在最新的 Ubuntu 20.04 版本中，VMware Tools 似乎已经包括在第 1 步中的软件更新里了，所以不必再重复安装了，可以跳过这步直接重启即可。但我们所用的 Ubuntu 18.04 版本并不是这样，所以还是要老老实实安装。

下面，我们安装 VMware Tools，如果你在桌面没有找到这个文件夹，就在 VMware 的菜单栏里找“虚拟机”里的“安装 VMware Tools”，加载安装包。如下图：



打开桌面的“VMware Tools”，对里面唯一的压缩包，即 VMwareTools-XXX.tar.gz（XXX 是版本号，不用担心版本出错，虚拟机提供的一定是可以用的），解压到桌面（右键菜单）。



下面打开终端（快捷键：Ctrl+Alt+T），用 `cd` 指令进入桌面解压出来的 `vmware-tools-distrib` 文件夹，然后切换到 `root` 用户（`su root`，见最上面理论准备部分），输入命令：`./vmware-install.pl`，按下回车后，会出现很多选项，可以全部按回车表示默认，提示 **Enjoy, –the VMware team** 后即为安装完成。如果出错了，自己不会百度吗魂淡！

3、更新软件

重启系统，我们安装的 VMware Tools 就已经正式生效了，在开始更新软件之前，可以顺便把下载好的源代码 `linux-5.4.115.tar.xz` 和实时补丁 `patch-5.4.115-rt57.patch.xz` 拖拽进虚拟机里的桌面了（这是我用的版本，你不一定要跟我一样）。此外你也可以解压完之后再拖进来，这样效率更高。

打开终端，依次执行以下三个命令（你也可以复制粘贴）：

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

```
sudo apt-get install gcc vim make libncurses5-dev openssl libssl-dev build-essential pkg-config  
libc6-dev bison flex libelf-dev zlibc minizip libidn11-dev libidn11
```

前两个更新软件源里的软件，最后一个查漏补缺，中间可能要你选择 `y/n`，输入 `y` 按回车即可。等到全部更新完成之后，再次重启。

重启之后，准备工作基本结束，我建议这里拍个快照，快照可以保存系统状态，便于事后恢复，但比较占空间，不宜拍太多。



三、添加系统调用、加载实时补丁

1、移动源码，添加系统调用

在开始之前，你可以在终端输入 `uname -a` 查看当前系统的信息，可以截图留作对照。

在桌面解压内核源码的压缩包，得到一个文件夹 `linux-5.4.115`。用 `mv` 命令把桌面的文件夹 `linux-5.4.115` 移动到绝对路径 `/usr/src/` 下，并进入该目录，即在终端输入：

```
cd ~/Desktop
```

```
sudo mv linux-5.4.115 /usr/src
```

```
cd /usr/src/linux-5.4.115
```

然后开始添加系统调用，我们会用到 `linux-5.4.115` 中的以下三个文件：

`arch/x86/entry/syscalls/syscall_64.tbl` //设置系统调用号

`include/linux/syscalls.h` //系统调用的头文件

`kernel/sys.c` //定义系统调用

由于它们都是只读的，我们需要更改权限。有很多种方法可以修改权限，这里我们用 `chmod` 命令实现，即在终端依次输入：

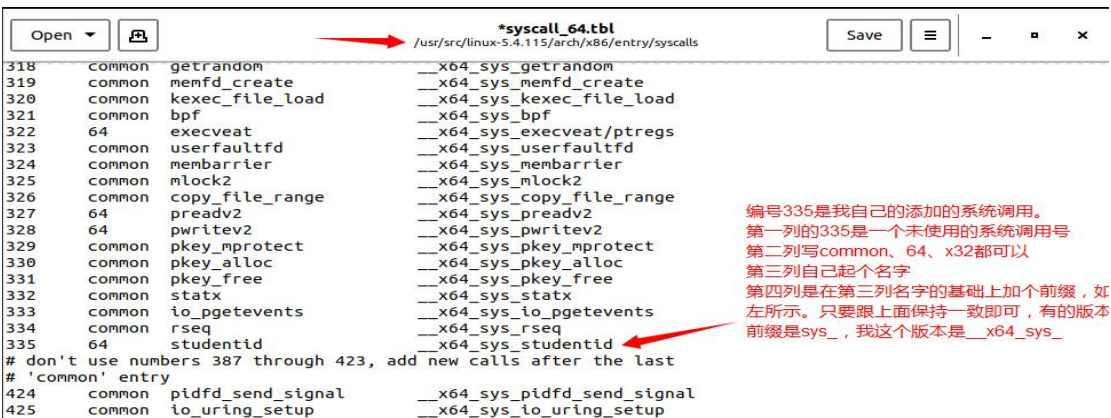
```
sudo chmod 666 arch/x86/entry/syscalls/syscall_64.tbl
```

```
sudo chmod 666 include/linux/syscalls.h
```

```
sudo chmod 666 kernel/sys.c
```

此时，这三个文件都可读可写了。依次在文件夹里打开这三个文件进行修改并保存：

①注意，要记住自己写的系统调用的编号，后面会用到的。内容见下图：



*syscall_64.tbl			
/usr/src/linux-5.4.115/arch/x86/entry/syscalls			
318	common	getrandom	__x64_sys_getrandom
319	common	memfd_create	__x64_sys_memfd_create
320	common	kexec_file_load	__x64_sys_kexec_file_load
321	common	bpf	__x64_sys_bpf
322	64	execveat	__x64_sys_execveat/ptregs
323	common	userfaultfd	__x64_sys_userfaultfd
324	common	membarrier	__x64_sys_membarrier
325	common	mlock2	__x64_sys_mlock2
326	common	copy_file_range	__x64_sys_copy_file_range
327	64	preadv2	__x64_sys_preadv2
328	64	pwritev2	__x64_sys_pwritev2
329	common	pkey_mprotect	__x64_sys_pkey_mprotect
330	common	pkey_alloc	__x64_sys_pkey_alloc
331	common	pkey_free	__x64_sys_pkey_free
332	common	statx	__x64_sys_statx
333	common	io_pgetevents	__x64_sys_io_pgetevents
334	common	rseq	__x64_sys_rseq
335	64	studentid	__x64_sys_studentid
# don't use numbers 387 through 423, add new calls after the last			
# 'common' entry			
424	common	pidfd_send_signal	__x64_sys_pidfd_send_signal
425	common	io_uring_setup	__x64_sys_io_uring_setup

编号335是我自己的添加的系统调用。
第一列的335是一个未使用的系统调用号
第二列写common、64、x32都可以
第三列自己起个名字
第四列是在第三列名字的基础上加个前缀，如左所示。只要跟上面保持一致即可，有的版本前缀是sys_，我这个版本是__x64_sys_

②这里是系统调用的头文件，给出了系统调用函数的声明，包括返回值类型、参数类型及参数，而前面的 `asm linkage` 是一个系统调用常用的宏，不要随意更改。



```
static inline long ksys_truncate(const char __user *pathname, long length)
{
    return do_sys_truncate(pathname, length);
}

static inline unsigned int ksys_personality(unsigned int personality)
{
    unsigned int old = current->personality;

    if (personality != 0xffffffff)
        set_personality(personality);

    return old;
}

asm linkage long sys_studentid(unsigned long long sid);

/* for __ARCH_WANT_SYS_IPC */
long ksys_semtimedop(int semid, struct sembuf __user *tsops,
                    unsigned int nsops,
```

③这里是系统调用的定义，必须符合②中所给出的返回值类型、参数类型及参数。



```
    return 0;
}

SYSCALL_DEFINE1(sysinfo, struct sysinfo __user *, info)
{
    struct sysinfo val;

    do_sysinfo(&val);

    if (copy_to_user(info, &val, sizeof(struct sysinfo)))
        return -EFAULT;

    return 0;
}

SYSCALL_DEFINE1(studentid, unsigned long long, sid)
{
    if(sid%2==0) return sid%1000000;
    else return sid%100000;
}

#ifdef CONFIG_COMPAT
struct compat_sysinfo {
```

至此，我们已经添加完成添加系统调用。至于对系统调用的测试，会在后面写到。

2、加载实时补丁（可选）

虽然我很想让你自己去百度，但是本着认真负责的态度，我还是含着泪把后面说了吧。在桌面解压实时补丁的压缩包，得到一个文件 `patch-5.4.115-rt57.patch`。

用 `mv` 命令把桌面的文件 `patch-5.4.115-rt57.patch` 移动到 `linux-5.4.115` 文件夹里，即继续在终端输入：

```
cd ~/Desktop
```

```
sudo mv patch-5.4.115-rt57.patch /usr/src/linux-5.4.115
```

然后用 `cd` 命令进入源代码文件夹，即在终端输入：

```
cd /usr/src/linux-5.4.115
```

再输入以下这条命令，来加载实时补丁：

```
sudo patch -p1 < patch-5.4.115-rt57.patch
```

等待加载完成。

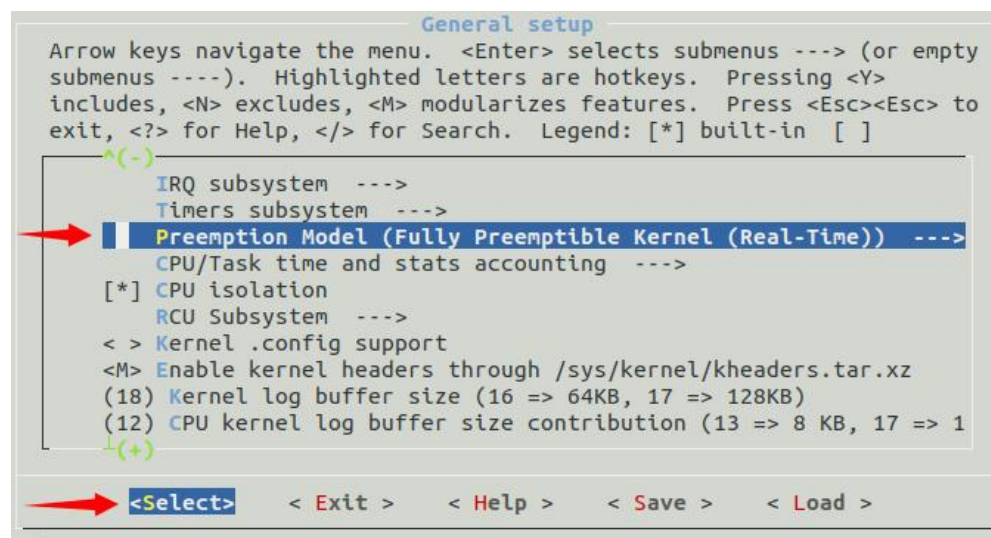
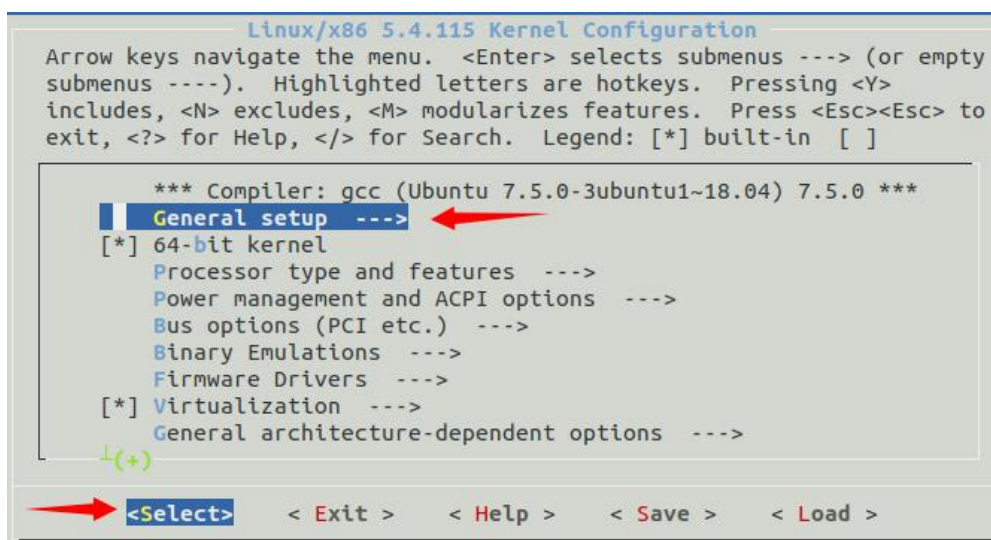
3、配置内核选项

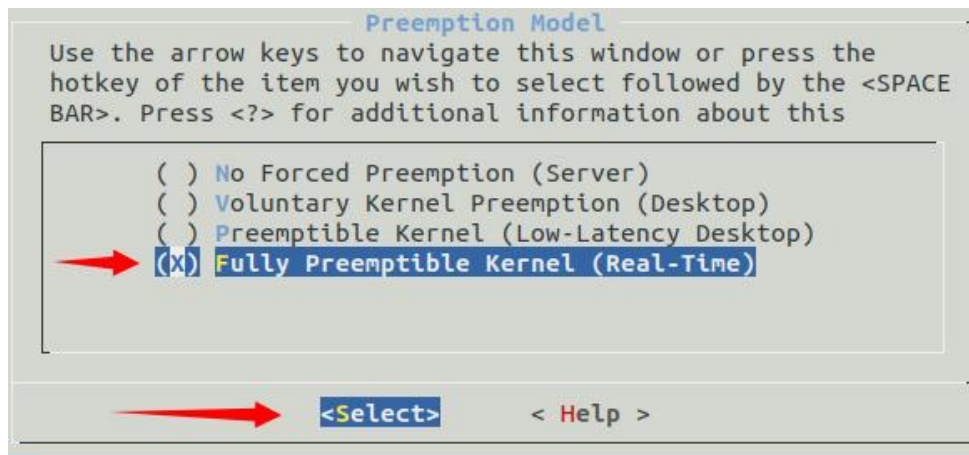
如果你做了第 2 步，且加载完成后，补丁其实还没有生效，需要在 menuconfig 中勾选相关选项，即在 **linux-5.4.115** 目录下，在终端输入：

```
sudo make mrproper // 这个命令是用来净化之前编译产生的文件，如果你是第一次编译，则不需要这个命令
```

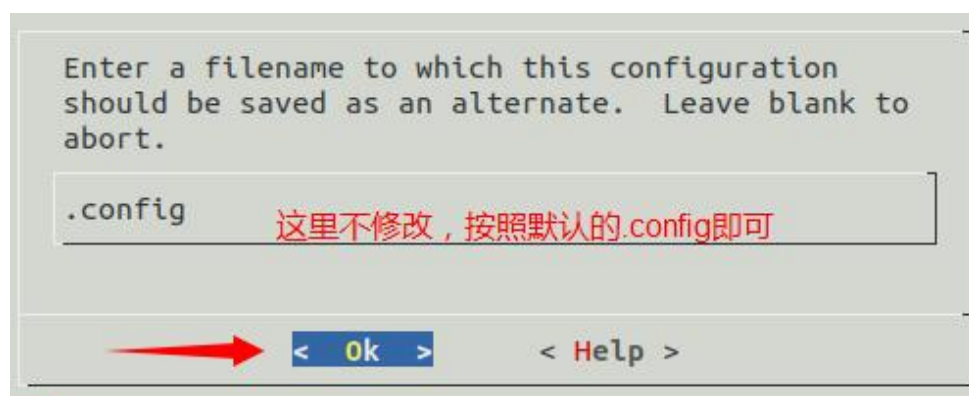
```
sudo make menuconfig // 对内核选项进行配置
```

加载后会弹出内核选项，依次进行如下操作：



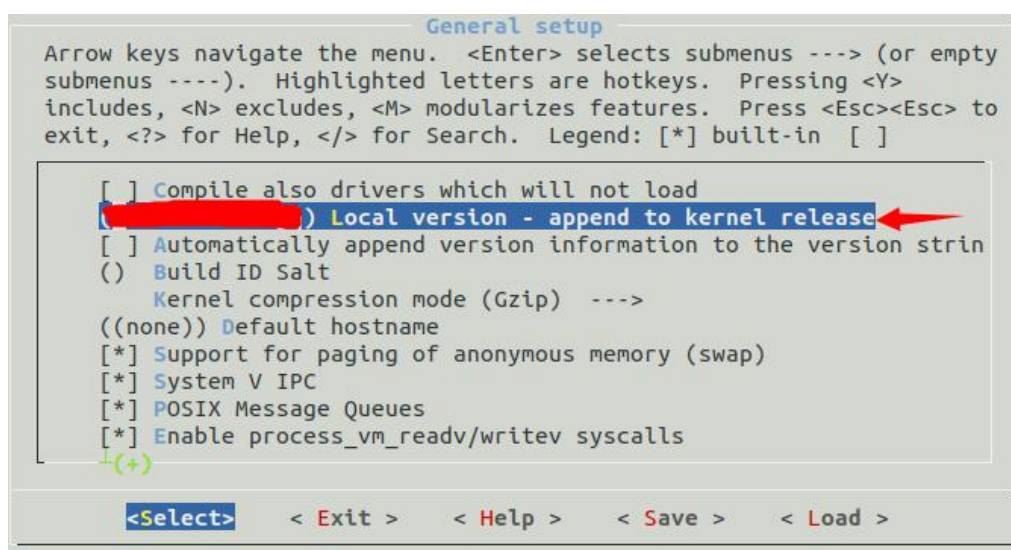


从这一步开始，即使没有打实时补丁也要做，目的是生成.config 文件。

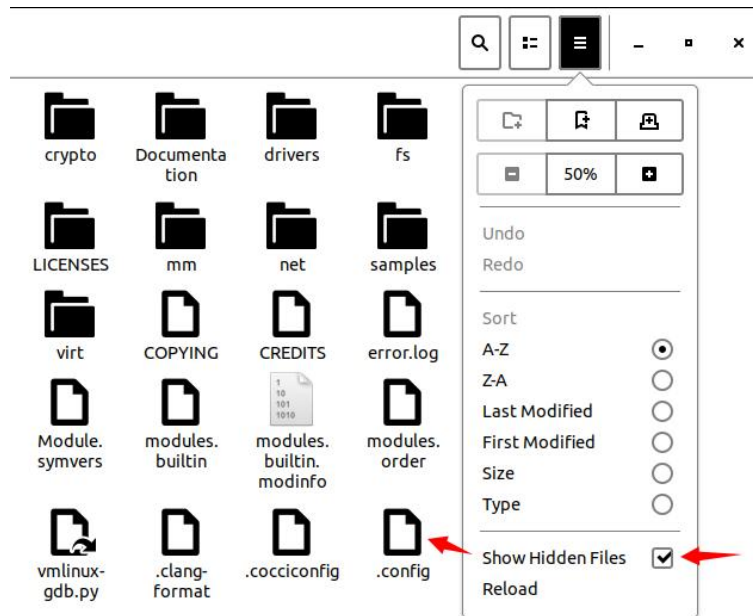


保存完成后，点 Exit 退出即可。

如果你愿意的话，也可以在退出之前，先进入 General setup >> Local version - append to kernel release 中向发行版本号内添加内容，比如我就把自己的学号添加进了内核版本号。



退出来之后，你会发现 **linux-5.4.115** 目录中多出了一个 **.config**。如果你没有找到这个文件，可能是因为这是个隐藏文件，在文件夹选项中可以勾选“显示隐藏文件”。



下面我们对这个 **.config** 文件进行修改，注意它也是只读的，因此需要仿照之前的操作，用 **chmod 666** 命令将其更改为可读可写的，具体操作参考[上文](#)。之后打开 **.config** 文件，搜索查找“**CONFIG_SYSTEM_TRUSTED_KEYS**”所在行，将后面字符串中的 **debian/canonical-certs.pem** 去掉，即该行内容修改为：**CONFIG_SYSTEM_TRUSTED_KEYS=""**

注：其实这里直接用 **vi** 或 **vim** 修改也可以，但是个人感觉不如修改权限后打开文件修改，更方便新手操作。

修改完成后，记得保存。

4、编译（引导）内核！！

终于到了最激（sang）动（xin）人（bing）心（kuang）的时刻，温馨提示，本环节耗时最长，且最有可能出错，如果你担心出错，可以在这之前再拍个快照，但一定要保证空间足够用，至少 **50GB**。本菜共反复重装 5——6 次，重复编译不下 **10** 次，耗时一周时间，最后一次成功编译约用 **5** 个小时。

我们依然是在 **linux-5.4.115** 目录中使用终端命令：

```
sudo make -j4 // 编译内核
```

这里的“-j4”表示 **4** 个线程（因为我们在安装的时候设置了 **4** 个内核，所以最多开 **4** 个）同时进行编译，可以在一定程度上加快速度。如果直接输入 **sudo make**，则默认单核。

经过漫长的等待，如果你有幸最终没有报错，那么继续使用命令：

sudo make modules_install // 安装模块

再次等待一段时间后，如果又幸运地没有报错，那么随我继续作死，使用命令：

sudo make install // 安装内核

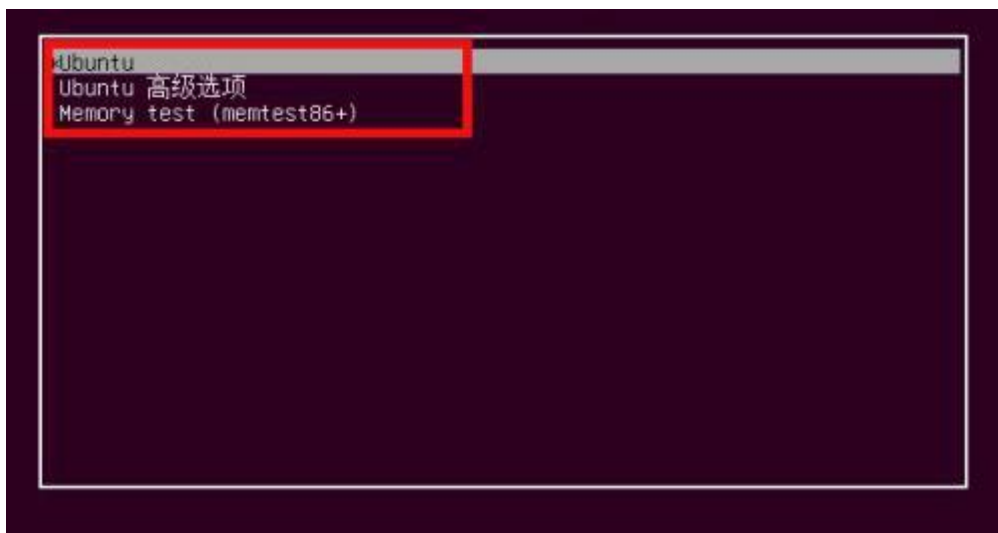
进行完这一步，如果依然没有错误，那么恭喜，本实验已经基本进入尾声了。

为了以防万一，可以再用一个命令：

sudo update-grub // 更新 grub

此时/lib/module 目录下应该就有安装好的内核。

重启后再次进入系统（重启可能会用较长时间，不要慌，耐心等待），如果重启时弹出 GRUB 引导界面，可以在 Ubuntu 高级（Advanced）选项中，选择自己安装的系统。



重启成功后，可以用 **uname -a** 命令查看当前系统信息，与实验前的截图对比不同。

5、测试系统调用

打开终端，输入命令：

cd ~/Desktop // 进入桌面

gedit test.c // 打开当前目录的 test.c 文件，如果没有，就创建一个再打开

然后会打开一个空白文本文件，自行输入测试系统调用的 C 语言源代码，保存到桌面。
以下是我的源代码，仅供参考！


```

#include <unistd.h>
#include <sys/syscall.h>
#include <stdio.h>
#define _SYSCALL_MYSETNICE_ 335 // 这是我自定义的系统调用的编号
int main()
{
    unsigned long long sid = 19030500000; // 学号
    long result = syscall(_SYSCALL_MYSETNICE_, sid); // 单数返回后 5 位，偶数返回后 6 位
    printf("The result is ");
    // 为了保证前面的 0 不被去掉，增加一步判断
    int num=6, i=0, k=1;
    // num 为学号后 5 (6) 位中第 1 个非 0 位之前 0 的个数，其初值为保留的位数 (5 或 6)
    num -= (sid%2);
    for(i=num; i>0; i--)
    {
        if(result/k==0) break;
        k *= 10;
        num--;
    }
    for(i=0; i<num; i++) printf("0"); // 根据 num 的值返回对应个数的 0
    if(num!=6) printf("%ld\n", result); // 特殊情况，即后 6 位全都是 0，需单独讨论
    else printf("\n");
    return 0;
}

```

我们继续在终端使用命令，依次进行编译和运行：

gcc -o test test.c // 编译 C 语言文件，生成编译文件

./test // 运行 C 语言程序

运行成功，实验结束！

```

File Edit View Search Terminal Help
renguiqi@renguiqi-virtual-machine:~/Desktop$ gcc -o test test.c
renguiqi@renguiqi-virtual-machine:~/Desktop$ ./test
The result is 00141
renguiqi@renguiqi-virtual-machine:~/Desktop$ █

```