Linux 系统概论

上机指南

伊现富　编著

天津医科大学

2014 年 12 月 11 日

目　录

实验 1　在虚拟机中安装并体验 Linux

安装 *Linux* 对于从未接触过 *Linux* 的人而言存在一定的难度。如果在已安装有其他操作系统的计算机上安装 *Linux*，错误的设置有可能导致原有数据全部丢失，造成不可估量的损失。为熟悉 *Linux* 的安装过程，可先使用虚拟机来模拟整个安装过程。对于只想尝试 *Linux* 的用户而言，在虚拟机中安装 *Linux* 也是一个不错的选择。本实验以 *Ubuntu 14.10* 和 *CentOS 7* 为例学习 *Linux* 的安装。

虚拟机（*Virtual Machine*）不是一台真正的计算机，而是利用真正计算机的部分硬件资源，通过虚拟机软件模拟出一台计算机。虽然只是虚拟机，却拥有自己的 *CPU* 等外围设备。现在的虚拟机软件已经能让虚拟机的功能和真正的计算机没有什么区别。用户可以对虚拟机进行磁盘分区、格式化、安装操作系统等操作，而对本身的计算机没有任何影响。

目前，比较常用的虚拟机软件有 *VMware* 公司出品的相关产品、甲骨文公司出品的 *Oracle VirtualBox* 以及微软公司出品的 *Virtual PC* 等。本实验以 *VirtualBox* 为例说明虚拟机的使用。*VirtualBox* 是以 *GNU* 通用公共许可证（*GPL*）发布的自由软件，并提供有二进制版本及开放源代码版本的代码，可从其官方网站*https://www.virtualbox.org/*下载，可运行于 *Windows* 和 *Linux* 等操作系统环境中。

一、实验要求

1. 掌握在 VirtualBox 中安装 Linux（Ubuntu 14.10 和 CentOS 7）的步骤。
2. 启动 Linux（Ubuntu 14.10 和 CentOS 7）并进行初始化设置。
3. 登录不同 Linux 发行版的桌面环境，了解 X Window 图形化用户界面。
4. 掌握注销与关机的方法。

二、实验准备

1. 一台已安装有 Windows 操作系统和 VirtualBox 软件的计算机。
2. 一张 Ubuntu 14.10 的安装光盘或 ISO 映像文件。
3. 一张 CentOS 7 的安装光盘或 ISO 映像文件。

三、实验内容

（一）Ubuntu 14.10 的安装与启动

1. 新建虚拟电脑。
   (a) 启动 VirtualBox 软件，进入主界面。单击“新建”图标新建一个虚拟机。
   (b) 在“新建虚拟电脑”的“虚拟电脑名称和系统类型”设置中，“名称”中输入虚拟机的名称，如“Ubuntu14.10”，“操作系统”选择“Linux”，“版本”选择“Ubuntu”。
   (c) 在“内存大小”的设置中，调整虚拟电脑的内存大小，不少于建议的内存大小。一般使用建议的内存大小即可。
   (d) 在“虚拟硬盘”的设置中，选中“现在创建虚拟硬盘”。

(e) 在“虚拟硬盘文件类型”的设置中，使用默认的“VDI”即可。

(f) 在“存储在物理硬盘上”的设置中，选中“固定大小”单选按钮。

(g) 在“文件位置和大小”的设置中，根据需要修改虚拟电脑的存储位置和虚拟硬盘的大小，虚拟硬盘要不小于建议的硬盘大小。单击“创建”按钮，稍等片刻即可完成虚拟电脑的创建。

2. 在虚拟机中安装 Ubuntu 14.10。

在 VirtualBox 虚拟机上安装操作系统时，既可以使用光盘进行安装，也可以利用 ISO 映像文件进行安装。此处通过 ISO 映像文件进行安装。

(a) 回到 VirtualBox 的主界面后，选中刚才创建的虚拟电脑。

(b) 点击主界面中的“设置”图标，可单击左侧的“存储”进行设置，为 IDE 控制器分配光驱，选择 Ubuntu 14.10 的虚拟光盘，即 ISO 映像文件。

(c) 回到 VirtualBox 主界面后，点击“启动”图标启动虚拟电脑。

(d) 虚拟机自动从安装源引导后进入 Ubuntu 14.10 的启动界面。在“Welcome”界面中，选择安装语言“中文（简体）”，界面随之刷新，选择“安装 Ubuntu”进行安装。

(e) 在“准备安装 Ubuntu”界面中进行相应的设置。一般默认即可。

(f) 在“安装类型”界面中，选择磁盘分区类型。对于初学者来说，使用默认的“清除整个磁盘并安装 Ubuntu”即可。

(g) 在“您在什么地方？”界面中，点选“Shanghai”设置时区。

(h) 在“键盘布局”界面中，一般默认即可。

(i) 在“您是谁？”界面中，设置个人信息。依次填写个人姓名、计算机名、用户名和密码。

(j) 点击“继续”安装系统。系统安装完成后，单击“现在重启”按钮，启动 Ubuntu 14.10。

3. 启动 Ubuntu 14.10。

(a) 启动 Ubuntu 14.10 后，将出现用户登录列表。选择相应的用户，输入对应的密码，回车确认输入。

(b) 用户名和密码验证通过后，进入 Unity 桌面环境。

4. 注销用户。

(a) 单击右上角类似齿轮的图标，从中选择“注销…”。

(b) 在弹出的对话框中单击“注销”按钮，将退出 Unity 桌面环境，屏幕再次显示登录界面，等待新用户登录系统。

5. 关机。

(a) 单击登录界面右上角类似齿轮的图标，选择“关机…”。

(b) 在弹出的对话框中单击“关机”按钮，系统将依次停止系统的相关服务，直至完全关闭计算机。

（二）CentOS 7 的安装与启动

参考 Ubuntu 14.10 的安装步骤在 VirtualBox 中安装 CentOS 7。

（三）体验其他 Linux 发行版。

在 VirtualBox 虚拟机中已经安装好了其他多个 Linux 发行版（Linux Mint，ElementaryOS，Fedora，openSUSE，Debian，Deepin 等），依次启动这些操作系统，体验不同的桌面环境。

实验 2　Linux 图形界面的基本操作

目前，*Linux* 操作系统上最常用的桌面环境主要是 *Unity*、*GNOME* 和 *KDE*，此外，还有 *Xfce*、*LXDE* 等多种桌面环境。*Ubuntu 14.10* 以 *Unity* 作为默认的桌面环境，*CentOS 7* 以 *GNOME* 作为默认的桌面环境。本实验以 *CentOS 7* 为基础，来练习 *GNOME* 桌面环境的基本操作。

一、实验要求

1. 了解 GNOME 系统面板的设置方法。
2. 掌握 GNOME 桌面环境的设置方法。
3. 掌握 GNOME 启动项的新建方法。
4. 掌握输入法的设置方法。
5. 掌握文件浏览器的使用方法。
6. 掌握桌面环境下管理用户与组的方法。
7. 了解其他的桌面环境。

二、实验准备

1. 安装有 CentOS 7 的计算机。

三、实验内容

（一）设置面板

1. 设置底部面板可隐藏。
   (a) 以普通用户身份登录 CentOS 7，进入 GNOME 桌面环境。
   (b) 右击底部面板的空白处，弹出快捷菜单，选择“属性”命令，弹出“面板属性”对话框。
   (c) 在“常规”选项卡中选中“自动隐藏”复选框，并选中“显示隐藏按钮”复选框。最后单击“关闭”按钮，底部面板将自动隐藏。
   (d) 移动光标到桌面的下边缘，出现底部面板。此时，底部面板的左右两端出线两个隐藏的按钮。单击左隐藏按钮，面板向左侧收缩。再单击左隐藏按钮，系统面板复原。
   (e) 再次设置底部面板，恢复其默认状态。
2. 在顶部面板上添加、移动和删除对象。
   (a) 右击顶部面板的空白处，弹出快捷菜单，选择“添加到面板”命令，弹出“添加到面板”对话框。
   (b) 选择“系统监视器”选项，单击“添加”按钮，则顶部面板出现黑底的系统监视器图标。单击这一图标，将打开“系统监视器”窗口。
   (c) 继续从“添加到面板”对话框中选择“抽屉”选项，单击“添加”按钮，可在面板上添加一个抽屉。单击“添加到面板”对话框中的“关闭”按钮，关闭“添加到面板”对话框。
   (d) 单击抽屉图标，可打开抽屉。

(e) 右击顶部面板的系统监视器图标，弹出快捷菜单，选择“移动”命令，此时系统监视器图标便会跟随鼠标移动。将其移动到抽屉中，单击鼠标左键，系统监视器图标将固定到抽屉中。单击向上箭头，可关闭抽屉。

(f) 右击抽屉图标，弹出快捷菜单，选择“从面板上删除”命令，出现提示信息。需要注意的是：删除抽屉将删除抽屉中的所有内容。单击“删除”按钮，连同系统监视器图标一起删除。

（二）设置桌面

1. 将桌面背景设置为瓢虫图片。

(a) 右击桌面空白处，弹出快捷菜单，选择“更改桌面背景”命令，弹出“外观首选项”对话框，选择“背景”选项卡。

(b) 选择有小瓢虫的图片，则所有工作区的桌面背景都发生变化。最后单击“关闭”按钮，关闭“外观选项卡”对话框。

2. 设置屏幕保护程序。

(a) 依次选择“系统”→“首选项”→“屏幕保护程序”命令，弹出“屏幕保护程序首选项”对话框。

(b) 选择“宇宙”主题，右侧预览框将显示屏幕保护时随机出现的与宇宙相关的图片。

(c) 拖动时间滑块将“于此事件后视计算机为空闲”设置为 1 分钟，并保持“计算机空闲时激活屏幕保护程序”复选框和“屏幕保护程序激活时锁定屏幕”复选框处于选中状态，即设置为 1 分钟后锁定屏幕，最后单击“关闭”按钮。

(d) 静置计算机，1 分钟后可观察到屏幕保护程序的效果，此时对计算机进行任何操作都将弹出对话框，要求输入用户的密码，密码验证成功才能回到桌面环境。

（三）设置桌面图标

1. 新建“我的文档”文件夹图标。

(a) 右击桌面空白处，弹出快捷菜单，选择“创建文件夹”命令，桌面出现一个新的文件夹，其名称默认为“未命名文件夹”。

(b) 按【Ctrl+Space】组合键，启动中文输入法，将文件夹名修改为“我的文档”，并按【Enter】键确认。

2. 新建文本编辑器 gedit 的启动器（图标）。

(a) 右击桌面空白处，弹出快捷菜单，选择“创建启动器”命令，弹出“创建启动器”对话框。

(b) 在“名称”文本框中输入应用程序快捷图标的名字“gedit”，在“命令”文本框中输入文本编辑器程序的路径“/usr/bin/gedit”。

(c) 单击图标按钮，弹出“选择图标”对话框，选中/usr/share/pixmaps 目录中的 apple-red.png 文件作为图标。单击“打开”按钮，回到“创建启动器”对话框。

(d) 单击“确定”按钮，桌面多出一个应用程序快捷图标，双击即可打开 gedit 文本编辑器。

（四）设置主题

1. 创建新主题。

(a) 依次选择“系统”→“首选项”→“外观”命令，弹出“外观首选项”对话框，CentOS 7 默认采用 System 主题。

(b) 单击“自定义”按钮，出现所有可用的主题细节信息，从“窗口边框”选项卡中选中 Crux 选项，注意边框样式的变化。

(c) 从“图标”选项卡中选择“十字架”选项，注意桌面上文件夹图标的变化。单击“关闭”按钮，回到“主题”选项卡。

(d) 此时所有主题列表的最前面出现一个新的“自定义主题”。单击“另存为”按钮，弹出“主题另存为”对话框，在“名称”文本框中输入主题名称，如“我的主题”，在“描述”文本框中输入对此主题的描述信息。最后单击“保存”按钮，回到“主题”选项卡。

(e) 此时在“主题”选项卡中可查看到刚定义的主题“我的主题”，并按照字母顺序排列在主题列表中。

（五）增加启动项

1. 实现登录桌面环境就自动启动文本编辑器（/usr/bin/gedit）。

(a) 依次选择“系统”→“首选项”→“启动应用程序”命令，弹出“启动应用程序首选项”对话框。

(b) 单击“添加”按钮，弹出“添加启动程序”对话框。在“名称”文本框中输入启动程序名，如“文本编辑器”，在“命令”文本框中输入文本编辑器的路径“/usr/bin/gedit”。单击“添加”按钮，回到“启动应用程序首选项”对话框，此时文本编辑器命令行将出现在“额外的启动程序”列表中。最后单击“关闭”按钮。

(c) 选择“系统”菜单中的“注销”命令，并单击“注销”按钮，当前用户退出。重新登录，可检查系统是否自动启动文本编辑器。

（六）设置输入法

GNOME 桌面环境在文本编辑器的编辑区域内，默认使用键盘输入的是英文字母和字符。按【Ctrl+Space】组合键，将切换到中文输入法。

1. 仅保留拼音输入法。

(a) 右击桌面空白处，弹出快捷菜单，从中选择“创建文档”→“空文件”命令创建一个空白文档，并将此文件命名为 f1。

(b) 双击 f1 文件，系统自动打开 gedit 文本编辑器。按【Ctrl+Space】组合键，切换至中文输入法。此时顶部面板出现“拼”字图标，表示当前采用“汉语-Pinyin”输入法。单击“拼”字图标显示所有可用的输入法。在 gedit 文本编辑器中输入任何英文字符，将显示出中文输入条。

(c) 对于大多数用户而言，只需要保留一种自己操作最熟练的输入法即可。依次选择“系统”→“首选项”→“输入法”命令，弹出“IM Chooser-输入法配置工具”对话框。单击“首选输入法”按钮，弹出“IBUS 设置”对话框。

(d) 选择“输入法”选项卡。选中不需要的输入法，单击“删除”按钮，最后单击“关闭”按钮。此时，单击顶部面板的“拼”字图标，显示只有“汉语-Pinyin”输入法可用。

2. 设置拼音输入法，启动模糊音。

(a) 单击顶部面板的“拼”字图标，从展开的菜单中选择“拼音首选项”命令，显示拼音输入法的初始状态和外观特征。

(b) 单击“模糊音”选项卡，选中“启动模糊音”复选框，可进一步选择允许哪些模糊音。

(c) 设置完成后，单击“关闭”按钮，并移动 f1 文件至用户主文件夹，为后续操作做准备。

（七）使用文件浏览器

1. 基本文件操作。

(a) 双击桌面上的用户主文件夹图标，启动文件浏览器，观察窗口的各组成部分，并可发现

新建的 f1 文件。

(b) 右击 f1 文件，弹出快捷菜单，选择“复制”命令，然后在窗口空白处再次右击，从弹出的快捷菜单中选择“粘贴”命令，文件浏览器中多出一个文件，名为“f1（复件）”。

(c) 右击 f1 文件，弹出快捷菜单，选择“创建链接”命令，窗口中多出一个链接文件，名为“到 f1 的链接”。

(d) 右击“f1（复件）”文件，弹出快捷菜单，选择“重命名”命令，文件名变为可编辑的文本框，输入新的文件名 f2，然后单击窗口空白处。

(e) 在窗口的空白处右击，弹出快捷菜单，选择“创建文件夹”命令，出现一个文件夹，默认名为“未命名文件夹”，文件名处于可编辑状态，输入新文件夹名 backup。

(f) 拖动 f2 文件至 backup 文件夹中实现文件的移动。

(g) 右击 f1 文件，弹出快捷菜单，选择“属性”命令。弹出“f1 属性”对话框，选中“徽标”选项卡中的 urgent 徽标。此时。f1 文件图标上出现紧急徽标。

2. 查看隐藏文件。

(a) 选择“查看”菜单中的“显示隐藏文件”命令。

(b) 文件浏览器窗口中多出一些目录和文件。这些文件的文件名都以“.”开头，是 Linux 中的隐藏文件。

（八）桌面环境下管理用户与组。

1. 新建两个用户账号，其用户名为 xuser1 和 xuser2，密码为 e12ut59er 和 wfu1t28er。

(a) 以超级用户身份登录 X Window 图形化用户界面，依次选择“系统”→“管理”→“用户和组群”命令，打开“用户管理者”窗口。

(b) 单击工具栏中的“添加用户”按钮，打开“添加新用户”窗口。在“用户名”文本框中输入用户名 xuser1，在“密码”文本框中输入密码 e12ut59er，在“确认密码”文本框中再次输入密码，然后单击“确定”按钮，返回“用户管理者”窗口。

(c) 用同样的方法新建用户 xuser2。

(d) 依次选择“应用程序”→“附件”→“gedit 文本编辑器”命令，启动文本编辑器，打开/etc/passwd 和/etc/shadow 文件，将发现文件末尾均出现表示 xuser1 和 xuser2 用户账号的信息。打开/etc/group 和/etc/gshadow 文件，将发现文件末尾均出现表示 xuser1 和 xuser2 私人组的信息。

(e) 按【Ctrl+Alt+F2】组合键切换到第二个虚拟终端，输入用户名 xuser2 和相应的密码可登录系统，说明新建用户操作已成功。

(f) 输入命令 pwd，屏幕显示用户登录系统后，自动进入用户主目录“/home/xuser2”。

(g) 输入命令 exit，xuser2 用户退出登录。

(h) 按【Ctrl+Alt+F1】组合键返回 GNOME 桌面环境。

2. 锁定 xuser2 用户账号。

(a) 在“用户管理者”窗口选中 xuser2 用户账户，单击工具栏中的“属性”按钮，打开“用户属性”窗口。

(b) 选中“账户信息”选项卡，选中“本地密码被锁”复选框。单击“确定”按钮，返回“用户管理者”窗口。

(c) 按【Ctrl+Alt+F2】组合键，再次切换到第二个虚拟终端，输入用户名 xuser2 和相应的密码，发现 xuser2 用户无法登录系统，说明 xuser2 用户账号已被锁定。

(d) 再次返回 GNOME 桌面环境。

3. 删除 xuser2 用户。
   (a) 在“用户管理者”窗口中，选择“编辑”→“首选项”命令，弹出“首选项”对话框，不选中“隐藏系统用户和组”复选框，最后单击“关闭”按钮。此时“用户”选项卡中显示包括超级用户和系统用户在内的所有用户的账号信息。
   (b) 在“搜索过滤器”文本框中输入“x*”并按【Enter】键，则仅显示用户名以 x 为首字母的用户。
   (c) 选中 xuser2 用户，单击工具栏中的“删除”按钮，单击“是”按钮，返回“用户管理者”窗口，发现 xuser2 用户已被删除。
   (d) 在“搜索过滤器”文本框中输入“*”并按【Enter】键，则显示所有用户。
4. 新建两个组，分别是 myusers 和 temp。
   (a) 在“用户管理者”窗口选中“组群”选项卡，显示出所有组。
   (b) 单击工具栏中的“添加组群”按钮，打开“添加新组群”窗口。在“组群名”文本框中输入 musers，单击“确定”按钮，返回“用户管理者”窗口。
   (c) 用相同的方法新建 temp 组。
5. 修改 myusers 组属性，将 xuser1 用户加入 myusers 组。
   (a) 从“组群”选项卡中选择 myusers 组，单击工具栏中的“属性”按钮，打开“组群属性”窗口。
   (b) 选择“组群用户”选项卡，选中 xuser1 复选框，设置 xuser1 用户为 myusers 组的成员。单击“确定”按钮，返回“用户管理者”窗口。
6. 删除 temp 组。
   (a) 从“组群”选项卡中选择 temp 组，单击工具栏中的“删除”按钮，出现确认删除对话框，单击“是”按钮即可。

（九）体验其他桌面环境。

在 VirtualBox 虚拟机中已经安装好了其他多个 Linux 发行版（Ubuntu，Kubuntu，Xubuntu，Lubuntu 等），涉及多个桌面环境（Unity，KDE，Xfce，LXDE 等），依次启动这些操作系统，体验不同的桌面环境。

实验 3　Linux 命令行界面的基本操作

图形化用户界面下用户操作非常简单而直观，但到目前为止图形化用户界面还不能完成所有的操作任务。字符界面占用资源少，启动迅速，对于有经验的管理员而言，字符界面下使用 *shell* 命令更为直接高效。

shell 命令是 *Linux* 操作系统的灵魂，灵活运用 *shell* 命令可完成操作系统所有的工作。并且，类 *Unix* 的操作系统在 *shell* 命令方面具有高度的相似性。熟练掌握 *shell* 命令，不仅有助于掌握 *CentOS 7*，而且几乎有助于掌握各种发行版本的 *Linux*，甚至 *Unix*。

包括 *CentOS 7* 在内的 *Linux* 系统都具有虚拟终端。虚拟终端为用户提供多个互不干扰、独立工作的工作界面，并且在不同的工作界面可用不同的用户身份登录。也就是说，虽然用户只面对一个显示器，但可以切换到多个虚拟终端，好像在使用多个显示器。

CentOS 7 中不仅可在字符界面下使用 *shell* 命令，还可借助于桌面环境下的终端工具使用 *shell* 命令。桌面环境下的终端工具中使用 *shell* 命令可显示中文，而字符界面下默认仅显示英文。

一、实验要求

1. 掌握图形化用户界面和字符界面下使用 shell 命令的方法。
2. 掌握 ls、cd 等 shell 命令的功能。
3. 掌握 shell 命令挂载和卸载移动存储介质的方法。

二、实验准备

1. 安装有 CentOS 7 的计算机。

三、实验内容

（一）图形化用户界面下的 shell 命令操作。

1. 显示系统时间，并将系统时间修改为 2015 年 2 月 18 日零时。
   (a) 启动计算机，以超级用户身份登录图形化用户界面。
   (b) 依次选择“应用程序”→“系统工具”→“终端”命令，打开桌面环境下的终端工具。
   (c) 输入命令`date`，显示系统的当前日期和时间。
   (d) 输入命令`date 021800002015`，屏幕显示新修改的系统时间。在桌面环境的终端执行时显示中文提示信息。
2. 切换为普通用户，查看 2015 年 3 月 5 日是星期几。
   (a) 前一操作是以超级用户身份进行的，但通常情况下只有在必须使用超级用户权限时，才以超级用户身份登录系统执行操作。为提高操作安全性，输入命令 su - USER 切换为普通用户 USER。
   (b) 输入命令`cal 2015`，屏幕上显示出 2015 年的日历，由此可知 2015 年 3 月 5 日是星期四。

3. 查看 ls 命令的-s 选项的帮助信息。
   - 方法一:
     (a) 输入命令*man ls*，屏幕显示出手册页中 ls 命令相关帮助信息的第一页，介绍 ls 命令的含义、语法结构以及-a、-A、-b 和-B 等选项的含义。
     (b) 使用【PgDn】键、【PgUp】键以及上、下方向键找到-s 选项的说明信息。
     (c) 由此可知，ls 命令的-s 选项等同于*--size* 选项，以文件块为单位显示文件和目录的大小。
     (d) 在屏幕上的“:”后输入 q，退出 ls 命令的手册页帮助信息。
   - 方法二:
     (a) 输入命令*ls --help*，屏幕显示 ls 命令的中文帮助信息。
     (b) 拖动滚动条，找到-s 选项的说明信息。
     (c) 在屏幕上的“:”后输入 q，退出 ls 命令的手册页帮助信息。
4. 查看/etc 目录下所有文件和子目录的详细信息。
   (a) 输入命令*cd /etc*，切换到 /etc 目录。
   (b) 输入命令*ls -al*，显示 /etc 目录下所有文件和子目录的详细信息。

（二）字符界面下的 shell 命令操作。

1. 查看当前目录。
   (a) 启动计算机后，默认进入 CentOS 7 的图形化用户界面，此时按【Ctrl+Alt+F2】组合键切换到第二个虚拟终端。（也可使用图形化界面中的终端）
   (b) 输入一个普通用户的用户名和密码，登录系统。在字符界面下输入密码时，屏幕上不会出现类似“*”的提示信息，提高了密码的安全性。
   (c) 输入命令*pwd*，显示当前目录。
2. 用 cat 命令在用户主目录下创建一个名为 f1 的文本文件，内容为:

```
Linux is useful for us all.
You can never imagine how great it is.
```

   (a) 输入命令*cat >f1*，屏幕上输入点光标闪烁，依次输入上述内容。使用 cat 命令进行输入时，不能使用上、下、左、右方向键，只能用【Backspace】键来删除光标前一位置的字符。并且一旦按【Enter】键，该行输入的字符就不可修改了。
   (b) 上述内容输入后，按【Enter】键，让光标处于输入内容的下一行，按【Ctrl+D】组合键结束输入。
   (c) 要查看文件是否生成，输入命令*ls* 即可。
   (d) 输入命令*cat f1*，查看 f1 文件的内容。
3. 向 f1 文件增加以下内容：Why not have a try?
   shell 命令中可使用重定向来改变命令的执行。此处使用“>>”符号可向文件结尾处追加内容，而如果使用“>”符号则将覆盖已有的内容。shell 命令中常用的重定向符号共 3 个，如下所示:
   - >: 输出重定向，将前一命令执行的结果保存到某个文件。如果这个文件不存在，则创建此文件；如果这个文件已存在，则将覆盖原有内容。
   - >>: 附加输出重定向，将前一命令执行的结果追加到某个文件。
   - <: 将某个文件交由命令处理。

   (a) 输入命令*cat >>f1*，屏幕上输入点光标闪烁。

(b) 输入上述内容后，按【Enter】键，让光标处于输入内容的下一行，按【Ctrl+D】组合键结束输入。

(c) 输入命令`cat f1`，查看 f1 文件的内容，会发现 f1 文件增加了一行。

4. 统计 f1 文件的行数、单词数和字符数，并将统计结果存放到 countf1 文件中。

(a) 输入命令`wc <f1 >countf1`，屏幕上不显示任何信息。

(b) 输入命令`cat countf1`，查看 countf1 文件的内容，其内容是 f1 文件的行数、单词数和字符数信息，即 f1 文件共有 3 行、19 个单词和 87 个字符。

5. 将 f1 和 countf1 文件合并为 f 文件。

(a) 输入命令`cat f1 countf1 >f`，将两个文件合并为一个文件。

(b) 输入命令`cat f`，查看 f 文件的内容。

6. 分页显示 /etc 目录中所有文件和子目录的信息。

管道符号“|”用于连接多个命令，前一命令的输出结果是后一命令的输入内容。

(a) 输入命令`ls /etc | more`，屏幕显示出`ls /etc`命令输出结果的第一页，屏幕的最后一行还出现“–More–”或“–更多–”字样，按【Space】键可查看下一页信息，按【Enter】键可查看下一行信息。

(b) 浏览过程中按【Q】键，可结束分页显示。

7. 仅显示 /etc 目录中的前 5 个文件和子目录。

(a) 输入命令`ls /etc | head -n 5`，屏幕显示出`ls /etc`命令输出结果的前面 5 行。

8. 清除屏幕内容。

(a) 输入命令`clear`，则屏幕内容完全被清除，命令提示符定位在屏幕左上角。

（三）创建链接。

1. 创建原文件。

(a) 使用`cd ~`命令定位到主目录。

(b) 使用`touch original_file`命令创建一个名为 original_file 的文件。

(c) 运行`ls -l`命令来查看刚才创建的文件。注意该文件的链接数目。

2. 创建硬链接。

(a) 使用`ln original_file hard_link`命令对 original_file 创建一个硬链接，该链接命名为 hard_link。

(b) 再次运行`ls -l`。注意文件的链接数目和上一次的修改时间。

(c) 运行`ls -i`命令来显示文件的 inode 值。将会看到这两个文件的 inode 值是一样的。

3. 创建软链接。

(a) 使用`ln -s original_file soft_link`命令为 original_file 创建一个软链接，该链接命名为 soft_link。

(b) 再次使用`ls -l`命令显示文件。注意文件的链接数目和修改时间。

(c) 使用`ls -i`命令来查看所有文件的 inode 值。可以看出 soft_link 的 inode 与 original_file 和 hard_link 的 inode 不同。

4. 原文件对链接文件的影响。

(a) 使用`cat`命令来查看每个文件的内容，确认这些文件中没有包含任何文本或数据。

(b) 要了解原文件的改变是如何影响链接文件的，可以使用

```
echo "This text goes to the original_file" >>original_file
```

命令给 original_file 中添加一行“This text goes to the original_file”。

(c) 运行*ls -l* 命令来查看原文件和链接文件在文件大小方面的变化。虽然我们只给 original_file 文件添加了数据，但 hard_link 文件的大小也会发生变化，而 soft_link 文件的大小没有变化。

(d) 使用*cat* 命令来查看文件的内容，可以发现三个文件具有完全相同的内容。

5. 硬链接对原文件的影响。

(a) 要了解硬链接文件的变化对原始文件的影响，可以使用

echo "This text goes to the hard_link file" >>hard_link

命令给 hard_link 中添加一行文本“This text goes to the hrad_link file”。

(b) 运行*ls -l* 并观察输出。original_file 和 hard_link 的修改时间和大小都发生了改变，但是 soft_link 没有变化。

(c) 现在再次使用*cat* 命令来显示文件的内容，注意每个文件的变化。

6. 软链接对原文件的影响。

(a) 如果使用*echo* 命令给 soft_link 文件添加一行文本，将会修改原始文件，从而也更新了 hard_link 文件。使用

echo "This text goes to the soft_link file" >>soft_link

命令给 soft_link 文件中添加一行“This text goes to the soft_link file”。

(b) 使用*cat* 命令来显示文件的内容。

（四）字符界面下使用移动存储介质。

1. 将光盘中的任意一个文件复制到用户主目录，最后卸载光盘。

(a) 登录系统，使用命令*ls /mnt/cdrom* 查看光盘的挂载点是否有内容，/mnt/cdrom 目录应为空。

(b) 利用 mount 命令，手动挂载光盘。手动挂载光盘时，不仅可以使用*mount /dev/cdrom* 命令，也可以使用*mount /mnt/cdrom* 命令。也就是说，mount 命令的参数既可以是设备名，也可以是挂载点。

(c) 查看挂载点的内容，也就是查看光盘的内容。只能通过查看挂载点目录（例如，*ls /mnt/cdrom*）来查看移动存储介质的内容，而*ls /dev/cdrom* 命令查看到的只是光盘的设备信息。

(d) 使用 cp 命令复制光盘中任意一个文件到用户主目录。

(e) 最后利用 umount 命令，卸载光盘。卸载光盘时，不仅可以使用*umount /dev/cdrom* 命令，也可以使用*umount /mnt/cdrom* 命令。

2. 将 U 盘上的任意一个文件复制到用户主目录，并查看所有磁盘的使用情况，最后卸载 U 盘。

(a) 登录系统，插入 U 盘。CentOS 7 自动显示 U 盘的相关信息，按【Enter】键，出现命令提示符。

(b) 利用*mount /mnt/usb* 命令，手动挂载 U 盘。

(c) 使用*ls /mnt/usb* 命令查看 U 盘中的文件内容。

(d) 使用*df* 命令查看已挂载的文件系统，并可了解 U 盘的使用率。此时，不仅能查看到硬盘上分区的使用率，还能查看到 U 盘的挂载信息和使用率。

(e) 使用 cp 命令复制 U 盘中任意一个文件到用户主目录。

(f) 最后利用*umount /mnt/usb* 命令，卸载 U 盘。

（五）常见文件与文件夹操作的命令实现。

回忆、总结图形界面下文件与文件夹的常见操作（如：新建、删除、重命名、移动等），在命令行界面中用命令将其实现。

实验 4　Linux 常用命令的操作

一、实验要求

1. 掌握联机帮助页的使用方法。
2. 掌握命令选项的使用方法。
3. 熟练掌握目录和文件管理的相关方法。
4. 掌握文件权限的修改方法。
5. 掌握文件归档和压缩的方法。
6. 了解进行系统性能监视的基本方法。
7. 掌握重定向、管道、通配符、历史记录等的使用方法。
8. 掌握对文件进行排序的方法。
9. 掌握利用 shell 命令管理用户和组群的方法。
10. 理解 /etc/passwd 和 /etc/group 文件的含义。
11. 了解批量新建用户账号的步骤和方法。

二、实验准备

1. 安装有 CentOS 7 的计算机。

三、实验内容

（一）使用联机帮助页。

1. 使用联机帮助页，搜索指定的关键字，查看显示了哪些命令。如果一个关键字都想不起来，可以使用`man -k shell`。
2. 从搜索结果列表中选择一个命令，阅读它的联机帮助页。

（二）对 ls 命令使用选项。

尝试这个练习以了解 Linux 命令语法的灵活性。使用 ls 命令，添加-a 选项以便在目录列表中包含隐藏文件；隐藏文件是那些文件名以点号开头的文件，例如.bashrc。

1. 执行命令`ls -l -a /etc`。
2. 执行命令`ls -la /etc`。
3. 比较两个命令的输出。它们的结果是一样的。

（三）文件管理。

1. 创建两个新目录 dir1 和 dir2，然后将 dir2 目录移到 dir1 目录中，最后删除 dir2 目录。
   (a) 登录计算机，打开终端，当前目录为用户的主目录。
   (b) 输入命令`ls -l`，查看当前目录中的所有文件。
   (c) 创建两个新目录，输入命令`mkdir dir{1,2}`。使用 mkdir 命令创建多个目录时，如果目录名的开头都相同，可利用“{}”符号。

(d) 再次输入命令`ls -l`，确认两个目录是否成功创建。
(e) 输入命令`mv dir2 dir1`，将 dir2 目录移动到 dir1 目录。
(f) 输入命令`cd dir1`，切换到 dir1 目录，再输入 ls 命令，会查看到 dir2 目录。
(g) 输入命令`rm -rf dir2`，删除 dir2 目录。删除目录时，当前目录不能为被删除的目录或者其子目录。
(h) 输入命令`ls`，发现 dir2 目录确实已被删除。
(i) 输入命令`cd ~`，回到用户主目录。

2. 查找 profile 文件。
(a) 由于普通用户只对部分目录具有权限，不能从所有的目录查找文件。因此，先输入命令`su -`，并输入超级用户的密码，验证成功后，从普通用户切换到超级用户。
(b) 使用`find / -name profile`命令进行查找，屏幕显示已找到 /etc/profile 文件。
(c) 使用`exit`命令，退出超级用户身份。

3. 将 /etc/profile 文件中所有包含 HOSTNAME 的行存入 f4 文件，并修改 f4 文件的权限，让所有的用户都可以读写。
(a) 使用命令`grep -n "HOSTNAME" /etc/profile > f4`，查找 /etc/profile 文件中所有包含 HOSTNAME 的行，并存入 f4 文件。grep 命令中使用“-n”选项可显示出行号。
(b) 输入命令`cat f4`，查看 f4 文件的内容。
(c) 输入命令`ls -l`，查看 f4 文件的详细信息。
(d) 使用`chmod 666 f4`命令，修改 f4 文件的权限。

4. 将 f4 文件复制到 dir1 目录，并在 dir1 目录中创建 /etc/fstab 文件的符号链接文件 fstab-link。
(a) 输入命令`cp f4 dir1`，将 f4 文件复制到 dir1 目录。
(b) 输入命令`ln -s /etc/fstab fstab-link`，创建 /etc/fstab 文件的符号链接文件。ln 命令使用-s 选项建立符号链接文件，一旦源文件被删除，符号链接文件就失效。
(c) 输入命令`ls -l`，可发现淡蓝色的符号链接文件 fstab-link，“->”符号后的内容为链接文件所指向的源文件。

5. 查看用户目录占用磁盘的情况。
(a) 输入命令`du -h`，显示当前目录和每个子目录的磁盘使用情况。
(b) 输入命令`du -sh`，显示当前目录总共使用的磁盘大小。

（四）文件归档与压缩。

1. 将 /etc/X11 目录归档为 X.tar 文件，并将 X.tar 文件压缩为.gz 文件。
(a) 方法一。
i. 输入命令`tar -czvf X.tar.gz /etc/X11`，将 /etc/X11 目录中的所有文件归档并压缩为 X.tar.gz 文件。
ii. 输入命令`tar -tf X.tar.gz`，可查看 X.tar.gz 所包含的所有文件。
(b) 方法二。
i. 输入命令`tar -cvf X.tar /etc/X11`，将 /etc/X11 目录中的所有文件归档为 X.tar 文件，屏幕将显示命令的执行过程。
ii. 输入命令`ls -l *.tar`，可发现新生成一个红色的 X.tar 文件。
iii. 压缩 X.tar 文件，输入命令`gzip X.tar`。
iv. 再次输入命令`ls -l *.tar.gz`，可发现 X.tar 文件已被 X.tar.gz 文件所取代，其字节数也有所减少。

v. 输入命令`tar -tf X.tar.gz`，查看 X.tar.gz 所包含的所有文件。

vi. 为方便下一步操作，输入命令`tar -tf X.tar.gz | grep applnk`，查看 X.tar.gz 是否打包和压缩了/etc/X11/applnk 目录。

2. 将 /etc/X11 目录归档压缩为 X11.tar.gz 文件，但跳过 /etc/X11/applnk 目录。
   (a) 输入命令`tar --exclude /etc/X11/applnk -czvf X11.tar.gz /etc/X11`，创建新的打包压缩文件 X11.tar.gz，但不包括 /etc/X11/applnk 目录。打包压缩生成.tar.gz 文件时，可利用“`--exclude`”选项，排除不需要打包的目录或文件。
   (b) 查看 X11.tar.gz 中是否打包和压缩了 /etc/X11/applnk 目录。
3. 将 /etc 目录中所有 2015 年 4 月 1 日以后有过更新的文件，打包压缩到 1504new.tar.gz 文件。
   (a) 输入命令`tar -N "2015/04/01" -czvf 1504new.tar.gz /etc`，显示大量的信息，如果在 2015 年 4 月 1 日以后没有更新的文件就会被跳过。打包压缩生成.tar.gz 文件时，可利用“-N 时间”选项，选定指定时间以后更新的文件进行打包压缩。
4. 将 X.tar.gz 中的 /etc/X11/Xresources 文件解压缩到 dir1 目录。
   (a) 首先切换到 dir1 目录，也就是解压缩的目标目录。
   (b) 执行`tar -xzvf ~/X.tar.gz etc/X11/Xresources`命令。
   (c) 查看解压缩的效果。
5. 使用 gzip。
   (a) 使用`cd /tmp;touch test-file`命令在 /tmp 目录中创建一个名为 test-file 的文件。
   (b) 使用 gedit 或 Vim 在这个文件中输入一些文本——10 行左右的随机语句——然后保存这个文件。
   (c) 使用`ls -l`命令显示文件的大小。
   (d) 为了证明可以查看这个文件，因为它是一个普通的文本文件，使用 cat 命令。
   (e) 使用 gzip 压缩这个刚才创建的文件。
   (f) 现在在 /tmp 目录中产生了一个名为 test-file.gz 的文件。
   (g) 使用`ls -l`命令查看压缩文件的大小。这个文件应该比未压缩的版本占用较少的空间，因为 gzip 压缩了它的大小。
   (h) 使用 gzip 或 gunzip 解压文件：`gzip -d test-file.gz`或`gunzip -d test-file.gz`。
   (i) 现在/tmp 目录中有一个名为 test-file 的文件（test-file.gz 文件已不存在）。使用 cat 命令查看该文件的内容。
   (j) 看到的输出应该和压缩该文件前使用 cat 命令看到的一样。再次使用`ls -l`命令，将看到这个文件和原来的文件大小相同。

（五）利用 shell 监视系统性能。

1. 输入命令 top，屏幕动态显示 CPU 使用率、内存使用率和进程状态等相关信息，且默认以 CPU 使用率进行排列。
2. 按【M】键，所有进程按照内存使用率排列。
3. 按【T】键，所有进程按照执行时间排列。
4. 按【P】键，恢复按照 CPU 使用率排列所有进程。
5. 按【Ctrl+C】组合键结束 top 命令。

（六）通配符的使用。

shell 命令的通配符包括`*`、`?`、`[]`、-和`!`，灵活使用通配符可同时引用多个文件，方便操作。

- *：匹配任意长度的任何字符。
- ?：匹配一个字符。
- []：表示范围。
- -：通常与[]配合使用，起始字符-终止字符构成范围。
- !：表示不在范围，通常也与[]配合使用。

1. 显示 /bin 目录中所有以 c 为首字母的文件和目录。
   (a) 输入命令`ls /bin/c*`，屏幕将显示 /bin 目录中以 c 开头的所有文件和目录。
2. 显示 /bin 目录中所有以 c 为首字母、文件名只有 3 个字符的文件和目录。
   shell 可以记录一定数量的已执行过的命令，当用户需要再次执行时，不用再次输入，可以直接调用。使用上、下方向键，【PaUp】或【PgDn】键，在 shell 命令提示符后将出现已执行过的命令。直接按【Enter】键就可以再次执行这一命令，也可以对出现的命令行进行编辑，修改为用户所需要的命令后再执行。
   (a) 按向上方向键，shell 命令提示符后出现上一步操作时输入的命令`ls /bin/c*`。
   (b) 将其修改为`ls /bin/c??`并执行，屏幕显示 /bin 目录中以 c 为首字母、文件名只有 3 个字符的文件和目录。
3. 显示 /bin 目录中所有的首字母为 c 或 s 或 h 的文件和目录。
   (a) 输入命令`ls /bin/[csh]*`，屏幕显示 /bin 目录中首字母为 c 或 s 或 h 的文件和目录。`[csh]*`并非表示所有以 csh 开头的文件，而是表示以 c 或 s 或 h 为首字母的文件。为避免误解，也可以使用`[c,s,h]*`，达到相同的效果。
4. 显示 /bin 目录中所有首字母是 v、w、x、y、z 的文件和目录。
   (a) 输入命令`ls /bin/[!a-u]*`，屏幕显示 /bin 目录中首字母是 v ～ z 的文件和目录。
5. 重复上一步操作。
   用户不仅可利用上、下方向键来显示执行过的命令，还可以使用 history 命令查看或调用执行过的命令。history 命令可以查看到已执行命令在历史记录列表中的序号，使用“! 序号”命令即可进行调用，而“`!!`”命令则执行最后执行过的那个命令。
   (a) 输入命令`!!`，自动执行上一步操作中使用过的`ls /bin/[!a-u]*`命令。
6. 查看刚执行过的 5 个命令。
   (a) 输入命令`history 5`，显示最近执行过的 5 个命令。

（七）对文件进行排序。

1. 用下面的文本创建文件 /tmp/outoforder：

```
Zebra
Quebec
hosts
Alpha
Romeo
juliet
unix
XRay
Xray
Sierra
Charlie
```

```
horse
horse
horse
Bravo
1
11
2
23
```

2. 以字典的顺序排列文件：*sort -d /tmp/outoforder*。注意，以大写字母开头的字符串位于所有小写单词的前面。
3. 单词 horse 在文件中出现了 3 次。要去冗余，即删除排序中额外的实例，可以使用如下命令：*sort -du /tmp/outoforder*。

（八）利用 shell 命令管理用户与组。

1. 新建一名为 duser 的用户，其密码是 tdd63u2，主要组群为 myusers。
   (a) 按【Ctrl+Alt+F3】组合键，切换到第三个虚拟终端，以超级用户身份登录。
   (b) 输入命令*useradd -g myusers duser*，建立新用户 duser，其主要组群是 myusers。
   (c) 为新用户设置密码，输入命令*passwd duser*，根据屏幕提示输入两次密码，最后屏幕提示密码成功设置信息。设置用户密码时，输入的密码在屏幕上并不显示出来，而输入两次的目的在于确保密码没有输错。
   (d) 输入命令*cat /etc/passwd*，查看 /etc/passwd 文件的内容，发现文件的末尾增加了 duser 用户的信息。
   (e) 输入命令*cat /etc/group*，查看 /etc/group 文件的内容，发现文件内容并未增加。
   (f) 按【Ctrl+Alt+F4】组合键，切换到第四个虚拟终端，输入 duser 用户名和密码可登录系统。
   (g) 输入命令*exit*，duser 用户退出登录。
2. 将 duser 用户设置为不需要密码就能登录。
   (a) 按【Ctrl+Alt+F3】组合键，切换到正被超级用户使用的第三个虚拟终端。
   (b) 输入命令*passwd -d duser*。
   (c) 按【Ctrl+Alt+F4】组合键，再次切换到第四个虚拟终端，在“Login:”后输入用户名 duser，按【Enter】键直接出现 shell 命令提示符，说明 duser 用户不需要密码即可登录。
3. 查看 duser 用户的相关信息。
   (a) 在第三个虚拟终端输入命令*id duser*，显示 duser 用户的用户 ID（UID）、主要组群的名称和 ID（GID）。
4. 从普通用户 duser 切换为超级用户。
   (a) 第四个虚拟终端当前的 shell 命令提示符为“*$*”，表明当前用户是普通用户。
   (b) 输入命令*ls /root*，屏幕上未列出 /root 目录中的文件和子目录，而是出现提示信息，提示当前用户没有查看 /root 目录的权限。
   (c) 输入命令*su -*或者是*su - root*，屏幕提示输入密码，此时输入超级用户的密码，验证成功后 shell 提示符从“*$*”变为“*#*”，说明已从普通用户转换为超级用户。
   (d) 再次输入命令*ls /root*，可查看 /root 目录中的文件和子目录。
   (e) 输入命令*exit*，回到普通用户的工作状态。

(f) 输入命令*exit*，duser 用户退出登录。

5. 一次性删除 duser 用户及其工作目录。
   (a) 按【Ctrl+Alt+F3】组合键，切换到正被超级用户使用的第三个虚拟终端。
   (b) 输入命令*userdel -r duser*，删除 duser 用户。处于登录状态的用户不能删除。如果在新建这个用户时还创建了私人组，而该私人组当前又没有其他用户，那么在删除用户的同时也将一并删除这一私人组。
   (c) 输入命令*cat /etc/passwd*，查看 /etc/passwd 文件的内容，发现 duser 用户的相关信息已消失。
   (d) 输入命令*ls /home*，发现 duser 用户的主目录 /home/duser 已不复存在。
6. 新建组 mygroup。
   (a) 在超级用户的 shell 提示符后输入命令*groupadd mygroup*，建立 mygroup 组。
   (b) 输入命令*cat /etc/group*，发现 group 文件的末尾出现 mygroup 组的信息。
   (c) 输入命令*cat /etc/gshadow*，发现 gshadow 文件的末尾也出现 mygroup 组的信息。
7. 将 mygroup 组改名为 newgroup。
   (a) 在超级用户的 shell 提示符后输入命令*groupmod -n newgroup mygroup*，其中-n 选项表示更改组名称。
   (b) 输入命令*cat /etc/group*，查看组信息，发现原来 mygroup 所在行的第一项变为 newgroup。
8. 删除 newgroup 组。
   (a) 超级用户输入命令*groupdel newgroup*，删除 newgroup 组。

（九）批量新建多个用户账号。

1. 为全班 30 位同学创建用户账号，用户名为“s”+学号的组合，其中班级名册中第一位同学的学号为 140101。所有同学都属于 class1401 组。所有同学的初始密码都为 123456。
   (a) 以超级用户身份登录系统，输入命令*groupadd -g 600 class1401*（假设值为 600 的 GID 未被使用），新建全班同学的组 class1401。
   (b) 输入命令 vim students，新建用户信息文件。
   (c) 按【i】键，切换为 Vim 的文本编辑模式，输入第一行信息“s140101:x:601:600::/home/s140101:/bin/bash”，完成后按【Esc】键，切换到命令模式。
   (d) 按【:】键，并输入*1,1co1*。此时将本身的一行复制为两行，继续输入*1,2co2* 从两行复制为四行，依此类推，直到出现 30 行信息。
   (e) 按【i】键，切换为文本编辑模式，修改每位同学用户信息不同的部分，编辑完成的文件如下所示。最后保存并退出 Vim。

```
s140101:x:601:600::/home/s140101:/bin/bash
s140102:x:602:600::/home/s140102:/bin/bash
s140103:x:603:600::/home/s140103:/bin/bash
...
```

   (f) 输入命令 vim stu-passwd，新建用户密码文件。
   (g) 按【i】键，切换为 Vim 的文本编辑模式，输入第一行信息“s140101:123456”，即所有同学的初始密码为 123456，然后进行复制，直至复制出 30 行信息，然后依次修改用户名。完成的文件如下所示，保存并退出 Vim。

```
s140101:123456
s140102:123456
s140103:123456
...
```

(h) 输入命令*newusers < students*，批量新建用户账号。

(i) 输入命令*pwunconv*，暂时取消 shadow 加密。

(j) 输入命令*chpasswd < stu-passwd*，批量新建用户的密码。

(k) 输入命令*pwconv*，进行 shadow 加密，完成批量创建用户账号工作。

(l) 输入命令*cat /etc/passwd*，查看/etc/passwd 文件将发现所有的用户账号均已建立。

(m) 可尝试以新建的用户名登录，并应该及时修改用户的密码。

(n) 使用此方法批量建立的用户登录系统时，其命令提示符不是默认的格式：*[用户名@localhost ~]$*，而是*-bash-4.1$*。如果希望使用标准的命令提示符，可复制用户主目录中的配置文件.bash_profile 和.bahsrc 到那些批量创建的用户的主目录中，再次登录，恢复标准的 shell 命令提示符。

实验 5　Linux 高级命令的操作

一、实验要求

1. 掌握 find 的使用方法。
2. 掌握 grep 的使用方法。
3. 掌握 sed 的使用方法。
4. 掌握 AWK 的使用方法。
5. 了解 datamash 的使用方法。
6. 了解 Linux 命令在生物信息学文本处理中的应用。

二、实验准备

1. 安装有 CentOS 7 的计算机。

三、实验内容

（一）使用 find。

1. 使用以下命令，创建测试 find 命令的文件。

```
# 切换至主目录
cd ~

# 创建测试find命令的根目录
mkdir find_test

# 切换至根目录
cd find_test

# 创建空文件
touch MybashProgram.sh
touch mycprogram.c
touch MyCProgram.c
touch Program.c

# 创建文件夹和文件
mkdir backup
cd backup
touch MybashProgram.sh
```

```
touch mycprogram.c
touch MyCProgram.c
touch Program.c

# 切换回根目录
cd ..

# 检查文件夹和文件的创建结果
ls -R
```

2. 用文件名查找文件。
   (a) 用 MyCProgram.c 作为查找名在当前目录及其子目录中查找文件：`find -name "MyCProgram.c"`。
   (b) 用 MyCProgram.c 作为查找名在当前目录及其子目录中查找文件，忽略大小写：`find -iname "MyCProgram.c"`。
3. 限定搜索指定目录的深度。
   (a) 在 root 目录及其子目录下查找 passwd 文件：`find / -name passwd`。
   (b) 在 root 目录及其 1 层深的子目录中查找 passwd：`find / -maxdepth 2 -name passwd`。
   (c) 在 root 目录下及其最大两层深度的子目录中查找 passwd 文件：`find / -maxdepth 3 -name passwd`。
   (d) 在第二层子目录和第四层子目录之间查找 passwd 文件：`find / -mindepth 3 -maxdepth 5 -name passwd`。
4. 在 find 命令查找到的文件上执行命令。
   (a) 计算所有不区分大小写的文件名为“MyCProgram.c”的文件的 MD5 验证和：`find -iname "MyCProgram.c" -exec md5sum {} \;`。`{}` 将会被当前文件名取代。
5. 相反匹配。
   (a) 显示所有名字不是 MyCProgram.c 的文件或者目录：`find -maxdepth 1 -not -iname "MyCProgram.c"`。由于 maxdepth 是 1，所以只会显示当前目录下的文件和目录。
6. 使用 inode 编号查找文件。
   (a) 创建两个名字相似的文件，例如一个有空格结尾，一个没有：`touch "test-file-name"`，`touch "test-file-name "`。
   (b) `ls -1 test*`。从 ls 的输出不能区分哪个文件是以空格结尾的。使用选项-i，可以看到文件的 inode 编号，借此可以区分这两个文件：`ls -i1 test*`。
   (c) 用 inode 编号重命名文件名中有特殊符号的文件：
   `find -inum INODE -exec mv {} new-test-file-name \;`，
   或者删除它：`find -inum INODE -exec rm {} \;`。
7. 根据文件权限查找文件。
   (a) 找到当前目录下对同组用户具有读权限的文件，忽略该文件的其他权限：`find . -perm -g=r -type f -exec ls -l {} \;`。
   (b) 找到对组用户具有只读权限的文件：`find . -perm g=r -type f -exec ls -l {} \;`。

(c) 找到对组用户具有只读权限的文件（使用八进制权限形式）：`find . -perm 040 -type f -exec ls -l {} \;`。

8. 查找空文件。
   (a) 找到 home 目录及子目录下所有的空文件（0 字节文件）：`find ~ -empty`。
   (b) 只列出 home 目录里的空文件：`find . -maxdepth 1 -empty`。
   (c) 只列出当前目录下的非隐藏空文件：
   `find . -maxdepth 1 -empty -not -name ".*"`。
9. 查找最大最小的文件。
   (a) 列出当前目录及子目录下 5 个最大的文件：
   `find . -type f -exec ls -s {} \; | sort -n -r | head -5`。
   (b) 查找 5 个最小的文件：
   `find . -type f -exec ls -s {} \; | sort -n  | head -5`。
   (c) 列出最小的文件，而不是 0 字节文件：
   `find . -not -empty -type f -exec ls -s {} \; | sort -n  | head -5`。
10. 查找指定文件类型的文件。
    (a) 查找 socket 文件：`find . -type s`。
    (b) 查找所有的目录：`find . -type d`。
    (c) 查找所有的一般文件：`find . -type f`。
    (d) 查找所有的隐藏文件：`find . -type f -name ".*"`。
    (e) 查找所有的隐藏目录：`find . -type d -name ".*"`。
11. 通过文件大小查找文件。+指比给定尺寸大，–指比给定尺寸小，没有符号代表和给定尺寸完全一样大。
    (a) 查找比指定文件大的文件：`find ~ -size +100M`。
    (b) 查找比指定文件小的文件：`find ~ -size -100M`。
    (c) 查找符合给定大小的文件：`find ~ -size 100M`。
12. 删除大型打包文件【谨慎操作，请勿尝试】。
    (a) 删除大于 100M 的 *.zip 文件：`find / -type f -name *.zip -size +100M -exec rm -i {} \;"`。
    (b) 删除所有大于 100M 的 *.tar 文件：`find / -type f -name *.tar -size +100M -exec rm -i {} \;"`。
13. 基于访问/修改/更改时间查找文件。
    (a) 找到当前目录及其子目录下，最近一次修改时间在 1 个小时（60 分钟）之内的文件或目录：`find . -mmin -60`。
    (b) 找到 24 小时（1 天）内被修改过的文件（文件系统根目录/下）：`find / -mtime -1`。
    (c) 找到当前目录及其子目录下，最近一次访问时间在 1 个小时（60 分钟）之内的文件或目录：`find . -amin -60`。
    (d) 找到 24 小时（1 天）内被访问过的文件（文件系统根目录/下）：`find / -atime -1`。
    (e) 在当前目录及其子目录下，查找 1 个小时（60 分钟）内文件状态发生改变的文件：`find . -cmin -60`。
    (f) 在根目录/及其子目录下，查找 1 天（24 小时）内文件状态发生改变的文件：`find / -ctime -1`。

(g) 显示 30 分钟内被修改过的文件，但文件夹不显示：*find /etc/sysconfig -mmin -30 -type f*。

(h) 显示当前目录及其子目录下，15 分钟内文件内容被修改过的文件，并且只列出非隐藏文件：*find . -mmin -15 \(! -regex ".*/\..*" \)*。

14. 基于文件比较的查找。

(a) 显示所有在 ordinary_file 之后创建修改的文件：*find -newer ordinary_file*。

(b) 显示在 /etc/passwd 修改之后被修改过的文件：*find -newer /etc/passwd*。

(c) 显示所有在 /etc/hosts 文件被修改之后被访问过的文件：*find -anewer /etc/hosts*。

(d) 显示在修改文件 /etc/fstab 之后所有文件状态发生改变的文件：*find -cnewer /etc/fstab*。

15. 在查找到的文件列表结果上直接执行命令。

(a) 在 find 命令输出上使用 ls -l，列举出 1 小时内被编辑过的文件的详细信息：*find -mmin -60 -exec ls -l {} \;*。

(b) 在同一个命令中使用多个：*find -name "*.txt" cp {} {}.bak \;*。

(c) 将所有 mp3 文件的文件名中的空格替换成下划线：*find . -type f -iname "*.mp3" -exec rename "s/ /_/g" {} \;*。

16. 将错误重定向到 /dev/null。

(a) 将错误消息重定向到 /dev/null 中去：*find / -name "*.conf" 2>>/dev/null*。

（二）使用 grep。

1. 搜索和寻找文件：*sudo dpkg -l | grep -i python*。
2. 搜索和过滤文件，除掉所有的注释行：*grep -v "#" /etc/apache2/sites-available/default-ssl*。
3. 找出艺术家 JayZ 的所有 mp3 格式的音乐文件，里面也不要有任何混合音轨：*find . -name ".mp3" | grep -i JayZ | grep -vi "remix"*。
4. 显示搜索字符串后面、前面或前后的几行：*ifconfig | grep -A 4 UP*，*ifconfig | grep -B 2 UP*|，*ifconfig | grep -C 2 lo*。
5. 计算匹配项的数目：*ifconfig | grep -c inet6*。
6. 按给定字符串搜索文件中匹配的行号：*grep -n "main" setup.py*。
7. 递归搜索：*grep -r "function" **。
8. 进行精确匹配搜索（包含要搜索的单词，而不是通配）：*ifconfig | grep -w RUNNING*。
9. 在 Gzip 压缩文件中搜索：*zgrep -i error /var/log/syslog.2.gz*。
10. 在一个文件或文件列表中搜索固定样式的字符串（读取保存在文件中的匹配模式），功能与 grep -F 相同：*fgrep -f file_full_of_patterns.txt file_to_search.txt*。

（三）使用 sed 进行操作。

1. 使用 vim 创建两个文件，每个文件含有一组名字：

```
#names1.txt
Paul
Craig
Debra
Joe
```

```
Jeremy

#names2.txt
Paul
Katie
Mike
Tom
Pat
```

2. 在命令行中输入并运行下面的命令：
   `sed -e s/Paul/Pablo/g names1.txt names2.txt > names3.txt`。
3. 使用`cat names3.txt`命令显示第三个文件的输出，以观察得到的名字列表。

（四）使用带有多个命令的 sed。

1. 定位先前示例中创建的两个含有一组名字的文本文件。
2. 利用 vim 命令创建一个名为 edits.sedscr 的新文件，并为 sed 列出一组编辑指令：

```
s/Pat/Patricia/
s/Tom/Thomas/
s/Joe/Joseph/
1d
```

3. 调用 sed：`sed -f edits.sedscr names1.txt names2.txt > names3.txt`。查看 names3.txt 文件中的内容。

（五）使用 AWK。

1. 在命令行中输入如下的 awk 命令：`awk '{print $0}' /etc/passwd`。

（六）使用 AWK 文件。

1. 使用 vim 命令输入如下内容，并将文件保存为 print.awk：

```
BEGIN {
  FS=":"
}
{ printf "username: " $1 "\t\t\t user id: " $3 }
```

2. 以如下方式执行 awk 命令：`awk -f print.awk /etc/passwd`。

（七）使用 datamash。

1. datamash 与 AWK、Perl 和 R 的比较（http://www.gnu.org/software/datamash/alternatives/）。
   (a) 计算总和、最小值、最大值、平均值：

```
# sum
seq 10 | datamash sum 1
seq 10 | awk '{sum+=$1} END {print sum}'

# minimum value
```

```
seq -5 1 7 | datamash min 1
seq -5 1 7 | awk 'NR==1 {min=$1} NR>1 && $1<min { min=$1  }
    END {print min}'

# maximum value
seq -5 -1 | datamash max 1
seq -5 -1 | awk 'NR==1 {max=$1} NR>1 && $1>max { max=$1  }
    END {print max}'

# mean
seq 10 | datamash mean 1
seq 10 | awk '{sum+=$1} END {print sum/NR}'
```

(b) 处理分组数据:

```
# data
DATA=$(printf "%s\t%d\n" a 1 b 2 a 3 b 4 a 3 a 6)

# First value of each group
echo "$DATA" | datamash -s -g 1 first 2
echo "$DATA" | awk '!($1 in a){a[$1]=$2} END {for(i in a) {
    print i, a[i] }}'

# Last value of each group:
echo "$DATA" | datamash -s -g 1 last 2
echo "$DATA" | awk '{a[$1]=$2} END {for(i in a) { print i, a
    [i]  }}'

# Number of values in each group:
echo "$DATA" | datamash -s -g 1 count 2
echo "$DATA" | awk '{a[$1]++} END {for(i in a) { print i, a[
    i]  }}'

# Collapse all values in each group:
echo "$DATA" | datamash -s -g 1 collapse 2
echo "$DATA" | perl -lane '{push @{$a{$F[0]}},$F[1]} END{
    print join("\n",map{"$_ ".join(",",@{$a{$_}})} sort keys
     %a);}'

# Collapse unique values in each group:
echo "$DATA" | datamash -s -g 1 unique 2
echo "$DATA" | perl -lane '{$a{$F[0]}{$F[1]}=1} END{print
```

```
    join("\n",map{"$_ ".join(",",sort keys %{$a{$_}})} sort
    keys %a);}'
23
24 # Print a random value from each group:
25 echo "$DATA" | datamash -s -g 1 rand 2
26 echo "$DATA" | perl -lane '{ push @{$a{$F[0]}},$F[1]  } END{
     print join("\n",map{"$_ ".$a{$_}->[rand(@{$a{$_}})] }
    sort keys %a ) ; }'
```

(c) 统计操作：

```
1  # A simple summary of the data, without grouping:
2  echo "$DATA" | datamash min 2 q1 2 median 2 mean 2 q3 2 max
       2
3  echo "$DATA" | Rscript -e 'summary(read.table("stdin"))
4
5  # A simple summary of the data, with grouping:
6  echo "$DATA" | datamash -s --header-out -g 1 min 2 q1 2
       median 2 mean 2 q3 2 max 2 | expand -t 18
7  echo "$DATA" | Rscript -e 'a=read.table("stdin")' -e '
       aggregate(a$V2,by=list(a$V1),summary)'
8
9  # Calculating mean and standard-deviation for each group:
10 echo "$DATA" | datamash -s -g 1 mean 2 sstdev 2
11 echo "$DATA" | Rscript -e 'a=read.table("stdin")' -e 'f=
       function(x){c(mean(x),sd(x))}' -e 'aggregate(a$V2,by=
       list(a$V1),f)'
```

(d) 反转、转置：

```
1  # reverse fields:
2  echo "$DATA" | datamash reverse
3  echo "$DATA" | perl -lane 'print join(" ", reverse @F)'
4
5  # transpose a file (swap rows and columns):
6  echo "$DATA" | datamash transpose
7  echo "$DATA" | Rscript -e 'write.table(t(read.table("stdin")
       ),quote=F,col.names=F,row.names=F)'
```

2. datamash 在生物信息学文本处理中的应用（http://www.gnu.org/software/datamash/examples/#example_genes）。

(a) 使用的数据：http://git.savannah.gnu.org/cgit/datamash.git/plain/examples/genes.txt。

(b) 数据中 16 列的含义：①bin，②name (isoform/transcript identifier)，③chromosome，④strand，

⑤txStart (transcription start site)，⑥txEnd (transcription end site)，⑦cdsStart (coding start site)，⑧cdsEnd (coding end site)，⑨exonCount n(umber of exons)，⑩exonStarts，⑪exonEnds，⑫score，⑬GeneName (gene identifier)，⑭cdsStartStat，⑮cdsEndStat，⑯exonFrames。

(c) 使用 datamash 处理 genes.txt 文件：

```
# Find Number of isoforms per gene
datamash -s -g 13 count 2 < genes.txt
# print all the isoforms for each gene
datamash -s -g 13 count 2 collapse 2 < genes.txt

# Find genes with more than 5 isoforms
cat genes.txt | datamash -s -g 13 count 2 collapse 2 | awk '
    $2>5'

# Which genes are transcribes from both strands?
cat genes.txt | datamash -s -g 13 countunique 4 | awk '$2>1'

# Which genes are transcribes from multiple chromosomes?
cat genes.txt | datamash -s -g 13 countunique 3 unique 3 |
    awk '$2>1'

# Examine Exon-count variability
cat genes.txt | datamash -s -g 13 count 9 min 9 max 9 mean 9
     pstdev 9 | awk '$2>1'

# How many transcripts are in each chromosome?
datamash -s -g 3 count 2 < genes.txt

# How many transcripts are in each chromsomse AND strand?
datamash -s -g 3,4 count 2 < genes.txt
```

（八）Linux 命令在生物信息学文本处理中的应用。

1. Useful bash one-liners useful for bioinformatics：
   https://github.com/stephenturner/oneliners。
2. Unix commands applied to bioinformatics：
   http://rous.mit.edu/index.php/Unix_commands_applied_to_bioinformatics。
3. Scott's list of linux one-liners：
   https://wikis.utexas.edu/display/bioiteam/Scott%27s+list+of+linux+one-liners。

实验 6　Linux 中的软件管理

一、实验要求

1. 掌握在命令行中下载文件的方法。
2. 掌握使用 APT 与 Yum 管理软件的方法。
3. 掌握使用 dpkg 与 RPM 管理软件的方法。
4. 掌握通过源代码安装软件的步骤。
5. 熟悉其他安装软件的方法。

二、实验准备

1. 安装有 CentOS 7 的计算机。
2. 安装有 Ubuntu 14.10 的计算机。

三、实验内容

（一）在命令行中下载安装包。

以 datamash 为例，使用 wget 和 curl 在命令行中下载其安装包。

```
# wget
wget -c http://files.housegordon.org/datamash/bin/datamash_1.0.6-1_amd64.deb
wget -c http://ftp.gnu.org/gnu/datamash/datamash-1.0.6.tar.gz
# curl
## 使用原始文件名
curl -O http://files.housegordon.org/datamash/bin/datamash-1.0.6-1.el6.x86_64.rpm
## 使用新的文件名
curl -o datamash.tar.gz http://ftp.gnu.org/gnu/datamash/datamash-1.0.6.tar.gz
```

（二）通过 APT 与 Yum 安装软件。

以 htop（或 Glances，dos2unix 等）为例，使用 APT 在 Ubuntu 中安装 htop，使用 Yum 在 CentOS 中安装 htop。

```
# 需要使用sudo
apt-get install htop
apt-get install glances
```

```
apt-get install dos2unix

# 需要切换为root用户
yum install htop
yum install glances
yum install dos2unix
```

（三）通过 dpkg 与 RPM 安装软件。

以 Webmin（或 datamash，TeamViewer，RStudio 等）为例，使用 dpkg 在 Ubuntu 中安装 Webmin，使用 RPM 在 CentOS 中安装 Webmin。

```
# 需要使用sudo
dpkg -i webmin_1.700_all.deb
dpkg -i datamash_1.0.6-1_amd64.deb
dpkg -i teamviewer_linux.deb
dpkg -i rstudio-0.98.1062-amd64.deb

# 需要切换为root用户
rpm -ivh webmin-1.700-1.noarch.rpm
rpm -ivh datamash-1.0.6-1.el6.x86_64.rpm
rpm -ivh teamviewer_linux.rpm
rpm -ivh rstudio-0.98.1062-x86_64.rpm
```

（四）通过源代码安装软件。

以 dos2unix（或 datamash，htop，parallel，FASTX-Toolkit，msort，SAMtools，BEDTools，seqtk 等）为例，使用源代码对其进行安装。

```
# dos2unix
tar -zxvf dos2unix-7.0.tar.gz
cd dos2unix-7.0/
vim INSTALL.txt
make
make check
#make strip
make install
#make clean
#make mostlyclean

# datamash
tar -zxvf datamash-1.0.6.tar.gz
cd datamash-1.0.6/
vim INSTALL
```

```
vim README
./configure
make
make check
sudo make install
#make installcheck
#make clean
#make uninstall

# htop
tar -zxvf htop-1.0.3.tar.gz
cd htop-1.0.3/
vim INSTALL
vim README
./configure
make
make install

# parallel
tar -xjvf parallel-20140822.tar.bz2
cd parallel-20140822/
vim README
./configure
make
make install

# FASTX-Toolkit
tar -xjvf fastx_toolkit-0.0.14.tar.bz2
cd fastx_toolkit-0.0.14/
vim INSTALL
vim README
./configure
make
sudo make install

# msort
tar -zxvf msort.tar.gz
cd msort/
vim README
./autogen.sh
./configure
```

```
make

# SAMtools
tar -xjvf samtools-1.0.tar.bz2
cd samtools-1.0/
vim INSTALL
make
make install

# BEDTools
tar -zxvf bedtools-2.21.0.tar.gz
cd bedtools2/
make
sudo cp ./bin/* /usr/local/bin

# seqtk
unzip seqtk-master.zip
cd seqtk-master/
make
```

（五）通过脚本安装软件。

以 cheat（或 Glances，Webmin 等）为例，使用脚本对其进行安装。

```
# cheat
## First install the required python dependencies
sudo pip install docopt pygments
## Then
unzip cheat-master.zip
cd cheat-master
vim README.md
sudo python setup.py install

# Glances
unzip Glances-master.zip
cd Glances-master
vim README.rst
python setup.py install

# Webmin
tar -zxvf webmin-1.700.tar.gz
cd webmin-1.700/
vim README
```

```
./setup.sh
```

（六）通过其他方式安装软件。

以 Galaxy（或 cheat，Glances 等）为例，根据安装说明对其进行安装。

```
# Galaxy
cd ~
hg clone https://bitbucket.org/galaxy/galaxy-dist/
cd galaxy-dist
hg update stable

# cheat
## Using pip
sudo pip install cheat
## Using homebrew
brew install cheat

# Glances
pip install Glances
```

（七）不需要安装的软件。

以 CPU-G（或 FASTX-Toolkit，WebLogo，TeamViewer，IGV 等）为例，下载后可以直接使用。

```
# CPU-G
tar -zxvf cpu-g-0.9.0.tar.gz
cd cpu-g-0.9.0/
chmod 755 cpu-g
./cpu-g

# FASTX-Toolkit
tar -xjvf fastx_toolkit_0.0.13_binaries_Linux_2.6_amd64.tar.bz2
cd bin

# WebLogo
tar -zxvf weblogo-3.3.tar.gz
cd weblogo-3.3/
./weblogo -h

# TeamViewer
tar -zxvf teamviewer_linux.tar.gz
cd teamviewer9/
./teamviewer
```

```
20
21 # IGV
22 unzip IGV_2.3.34.zip
23 cd IGV_2.3.34/
24 vim readme.txt
25 ./igv.sh
```

（八）从源代码编译 openssl 程序。

1. 在主目录中，创建一个目录（例如 src），然后切换到这个目录，软件的编译将在这个目录中进行。
2. 获取 openssl 的源代码。
3. 提取源代码并切换到新目录中。
4. 使用 ls 命令查看包含在安装目录中的文件（有多个 README* 和 INSTALL* 文件）。
5. 使用 less 或 more 命令阅读这些找到的文件。
6. 根据 INSTALL 文件中的说明，运行带有`--prefix`开关的 config 脚本。
7. 开始编译：`make`。
8. 为了在安装之前测试已编译好的程序，运行带有 test 的 make 命令：`make test`。
9. 安装软件：`make install`。

（九）编译需要预装软件的代码。

1. 切换到源代码编译目录。
2. 找到 Lynx 的源代码。
3. 提取源代码并切换到新创建的目录中。
4. 查找 README 或 INSTALL 文件。
5. 运行`./configure --help`查看编译选项列表。
6. 必须执行一个特殊的步骤，这个步骤配置 Lynx 编译程序以便安装到主目录中，并使用前面已经安装的 SSL 库。
7. 准备安装：`make`。
8. 为了安装软件，执行命令：`make install`。
9. 如果希望安装其他 Lynx 文档，可以根据屏幕上给出的提示进行操作。

（十）使用 RPM。

1. 在命令行中输入以下命令：`rpm -qa | grep mysql`。根据个人的安装情况，要么会获得一些输出，显示在机器上安装了哪个版本的 MySQL，要么什么输出都没有。
2. 要安装新版的 MySQL，需要从 MySQL 的网站下载相关的 RPM 包并保存到所需要的文件夹中。
3. 执行命令：`rpm -Uvh MySQL.rpm`。

（十一）软件管理。

尝试使用 dpkg 与 APT、RPM 与 Yum 对软件（如：htop，Glances，dos2unix 等）进行管理（如：查询、安装、卸载等）。

实验 7　Vim 编辑器的基本操作

一、实验要求

1. 熟悉并掌握 Vim 三种工作模式之间的转换方法。
2. 掌握新建和保存文件的操作方法。
3. 掌握插入和删除文本的操作方法。
4. 掌握查找和替换字符串的操作方法。
5. 掌握 Vim 的常用操作，如移动定位、复制粘贴、修改删除和重做撤销等。

二、实验准备

1. 安装有 Vim 编辑器的计算机。

三、实验内容

（一）新建文本文件。

1. 利用 Vim 新建文件 f3，内容为：

```
How to Read Faster

When I was a schoolboy I must have read every comic book ever published.
But as I got older, my eyeballs must have slowed down or something I
mean, comic books started to pile up faster than I could read them!

It wasn't until much later, when I was studying at college, I realized
that it wasn't my eyeballs that had gone wrong. They're still moving as
well as ever. The problem is that there's too much to read these days,
and too little time to read every WORD of it.
```

(a) 启动计算机，以普通用户身份登录字符界面。

(b) 在 Shell 命令提示符后输入命令*vim*，启动 Vim 文本编辑器。此时，Vim 默认进入命令模式。

(c) 按【i】键，从命令模式转换为文本编辑模式，此时屏幕的最底边出现“–INSERT–”字样。

(d) 输入上述文本内容。输入过程中如果出错，可使用【Backspace】键或【Delete】键删除错误的字符。

(e) 输入完成后，按【Esc】键返回命令模式。

(f) 按【:】键进入最后行模式，输入*w f2*，将正在编辑的内容保存为 f2 文件。

(g) 屏幕底部显示“"f2" [New] 3L, 495C written”字样，表示此文件有 3 行、495 个字符。Vim 中行的概念与平时所说的行有所区别，在输入文字的过程中由于字符串长度超过屏幕宽

度而发生的自动换行，Vim 并不认为是新的一行，只有在 Vim 中按一次【Enter】键、另起一行的才算作新的一行。

(h) 按【:】键后输入*q*，退出 Vim。

（二）编辑文件。

1. 打开 f2 文件并显示行号。
   (a) 输入命令*vim f2*，启动 Vim 文本编辑器并打开 f2 文件。
   (b) 按【:】键切换到最后行模式，输入*set nu*，每一行前出现行号。
   (c) Vim 自动返回到命令模式，连续两次按下【Z】键（注意大写），保存文件并退出 Vim。
2. 在 f2 文件的第二行后插入如下内容：“With the development of society, the ability of reading becomes more and more important.”，并在最后一行的后面再添加一行，内容为：“We must know some methods to read faster.”。
   (a) 再次输入命令*vim f2*，启动 Vim 文本编辑器并打开 f2 文件。
   (b) 移动光标到“When I was a schoolboy…”所在行，按【O】键，进入文本编辑模式，屏幕底部出现“–INSERT–”字样，Vim 直接在第二行下新起一行。
   (c) 输入“With the development of society, the ability of reading becomes more and more important.”
   (d) 将光标移动到最后一行的末尾，按【Enter】键，另起一行，输入“We must know some methods to read faster.”。
3. 将文本中所有的 eyeballs 替换为 eye-balls。
   (a) 按【Esc】键后输入“:”，进入最后行模式。因为当前 f2 文件中共有 5 行，所以输入*1,5s/eyeballs/eye-balls/g*，并按【Enter】键，将文件中所有的 eyeballs 替换为 eye-balls。
   (b) 进入最后行模式，输入*wq*，保存对文件的修改，并且退出 Vim。
4. 把第二行移动到文件末尾，删除第一行和第二行，随后撤销删除，最后不保存修改。
   (a) 再次输入命令*vim f2*，启动 Vim 文本编辑器并打开 f2。
   (b) 按【:】键，再次进入最后行模式，输入*2m5*，将第二行移动到第五行的后面。
   (c) 按【:】键，输入*1,2d*，删除第一行和第二行。
   (d) 按【u】键，恢复被删除的部分。
   (e) 按【:】键，进入最后行模式，输入*q!*，退出 Vim，不保存对文件的修改。
5. 复制第二行，并添加到文件的末尾，然后删除第二行，保存修改后退出 Vim。
   (a) 再次输入命令*vim f2*，启动 Vim 文本编辑器并打开 f2 文件。
   (b) 按【:】键，进入最后行模式，输入*2co5*，将第二行的内容复制到第五行的后面。
   (c) 移动光标到第二行，按下两次【d】键，删除第二行。
   (d) 按【:】键，输入*wq*，存盘并退出 Vim。
6. 新建 userlist 文件，要求从 /etc/passwd 文件中取出用户名，并在文件头添加注释信息“This is a userlist generated by /etc/passwd.”，注释信息的前后各空一行，并添加“#”符号设置这三行为注释信息。
   (a) 输入命令*vim userlist*，启动 Vim 文本编辑器并新建 userlist 文件。
   (b) 按【:】键，进入最后行模式，输入*r /etc/passwd*，在光标所在处读入 /etc/passwd 文件的内容。
   (c) 按【:】键，进入最后行模式，输入*%s/:.*//g*，其中*%*表示整个文档，而*:.**表示以*:*开始的部分。最末一行显示进行了多少替换。在 Vim 中进行字符串替换操作时，可用*%*

表示整个文档；^ 表示行首。

(d) 按【i】键，切换到文本编辑模式，并移动光标至文件的第一行，输入注释信息“This is a userlist generated by /etc/passwd.”，并按【Enter】键添加两行空行。

(e) 按【Esc】键后输入*:1,3s/^/#/g*，其中 ^ 表示行首。

(f) 最后按【:】键，输入*x*，存盘并退出 Vim。

7. 使用 Vim 中的命令。

(a) 使用*vim /tmp/beginning_unix_testfile* 在 /tmp 目录下创建一个新文件，命名为 beginning_unix_testfile。

(b) 目前处在命令模式下，通过输入【i】切换到插入模式。

(c) 输入下面的内容，在每行的末尾都必须按【Enter】键：

```
The quick brown fox jumps over the lazy dog.
Sentence two.
Sentence three.
Sentence four.
Vi will never become vii.
Sentence six.
```

(d) 按下【Esc】键返回到命令模式。

(e) 输入*1G* 使游标移动到第一行，然后输入*4l*，从而使游标放在 quick 中的 q 上。

(f) 输入*cw*，该命令将删除单词 quick 并切换到插入模式。

(g) 输入单词 slow，然后按下【Esc】键。文件现在的内容是：

```
The slow brown fox jumps over the lazy dog.
Sentence two.
Sentence three.
Sentence four.
Vi will never become vii.
Sentence six.
```

(h) 输入*2j* 使游标向下移动两行。游标将位于第 3 行上的 Sentence 中的最后一个 e 上。

(i) 输入*r*，然后输入*E*。该句将变成：

```
SentencE three.
```

(j) 输入*k* 使游标上移一行。输入 2yy 复制两行。

(k) 输入*4j* 使游标移动到最后一行，然后输入*p* 粘贴缓冲区内的文本。得到的结果是：

```
The slow brown fox jumps over the lazy dog.
Sentence two.
SentencE three.
Sentence four.
Vi will never become vii.
Sentence six.
Sentence two.
SentencE three.
```

(l) 按下【Esc】键，然后输入*:q!* 退出该文件（没有保存输入）。

（三）Vim 的常用操作。

自由练习 Vim 的常用操作（如：启动，保存，退出，移动定位，修改删除，复制粘贴，搜索替换等）。

实验 8　shell 脚本编程

一、实验要求

1. 掌握 shell 脚本的结构和运行方法。
2. 掌握 shell 函数的使用方法。
3. 掌握 shell 脚本中的流程控制。
4. 掌握 shell 脚本的调试方法。

二、实验准备

1. 安装有 CentOS 7 的计算机。

三、实验内容

（一）创建一个简单的 shell 脚本。

1. 在主目录中创建 bin 目录，并进入其中。
2. 创建一个空白文档，向其中输入下面几行内容：

```
# ~/bin/dirinfo.sh
# 20150501
# A really stupid script to give some basic info about the
    current directory

#!/bin/bash
pwd
ls
```

3. 把文件保存为 dirinfo.sh
4. 利用下面的命令使文件可执行：`chmod a+x dirinfo.sh`。
5. 运行该脚本：`./dirinfo.sh`。

（二）传递参数给函数。

1. 在主目录的 bin 目录中，创建一个空白文档，输入以下代码，并将其保存为 func.sh。

```
#!/bin/bash

# func

# A simple function
```

```
6
7  repeat ( ) {
8    echo -n "I don't know $1 $2"
9  }
10
11 repeat Your Name
```

2. 使脚本变成可执行文件：*chmod 755 ~/bin/func.sh*。
3. 在命令行运行脚本：*~/bin/func.sh*。

（三）使用嵌套函数。

1. 在主目录的 bin 目录中，创建一个空白文档，输入以下代码，并将其保存为 nested.sh。

```
1  #!/bin/bash
2
3  # nested
4  # Calling one function from another
5
6  number_one ( ) {
7    echo "This is the first function speaking..."
8    number_two
9  }
10
11 number_two ( ) {
12   echo "This is now the second function speaking..."
13 }
14
15 number_one
```

2. 使脚本变成可执行文件。
3. 在命令行中运行该脚本。

（四）处理作用域。

1. 在主目录的 bin 目录中，创建一个空白文档，输入以下代码，并将其保存为 scope.sh。

```
1  #!/bin/bash
2
3  # scope
4  # dealing with local and global variables
5
6  scope ( ) {
7    local lclVariable=1
8    gblVariable=2
9    echo "lclVariable in function = $lclVariable"
```

```
10    echo "gblVariable in function = $gblVariable"
11  }
12
13  scope
14
15  # We now test the two variables outside the function block to
        see what happens
16
17  echo "lclVariable outside function = $lclVariable"
18  echo "gblVariable outside function = $gblVariable"
19
20  exit 0
```

2. 使脚本变成可执行文件。
3. 在命令行中运行该脚本。

（五）使用 getopts。

1. 在主目录的 bin 目录中，创建一个空白文档，输入以下代码，并将其保存为 get.sh。

```
1  #!/bin/bash
2
3  # get
4  # A script for demonstrating getopts
5
6  while getopts "xy:z:" name
7  do
8    echo "$name" $OPTIND $OPTARG
9  done
```

2. 使脚本变成可执行文件。
3. 在命令行调用脚本，传递一些参数，如：`~/bin/get.sh -xy "one" -z "two"`。

（六）用 trap 命令清除临时文件。

1. 在主目录的 bin 目录中，创建一个空白文档，输入以下代码，并将其保存为 sigtrap.sh。

```
1  #!/bin/bash
2
3  # sigtrap
4  # A small script to demonstrate signal trapping
5
6  tmpFile=/tmp/sigtrap$$
7  cat > $tmpFile
8
9  function removeTemp( ) {
```

```
  if [ -f "$tmpFile" ]
  then
    echo "Sorting out the temp file..."
    rm -f "$tmpFile"
  fi
}

trap removeTemp 1 2

exit 0
```

2. 使脚本变成可执行文件。
3. 在命令行中运行该脚本。
4. 在命令行输入一些文本然后按下【Ctrl+D】组合键。一旦按下这个组合键，就会立即检查在 tmp 目录中创建的文件（它的名字可能和 sigtrap(procid) 差不多）。
5. 现在再次运行代码，但在按下【Ctrl+D】组合键之前，先按下【Ctrl+C】组合键。这一次会收到一条消息 “Sorting out the temp file...”。如果此时去查看名称类似于 sigtrap(procid) 的文件，将发现它并不存在。

（七）在数组中使用文本文件中的数据。

1. 在主目录的 tmp 目录（如果不存在，就先创建该目录）中，创建一个空白文档，输入以下文本，并将其保存为 sports.txt。

```
rugby hockey swimming
polo cricket squash
basketball baseball football
```

2. 在主目录的 bin 目录中，创建一个空白文档，输入以下代码，并将其保存为 array.sh。

```
#!/bin/bash

# array
# A script for polulating an array from a file

populated=('cat ~/tmp/sports.txt | tr '\n' ' '')
echo ${populated[@]}

exit 0
```

3. 使脚本变成可执行文件。
4. 在命令行中运行该脚本。

（八）shell 脚本实例。

```
# 模拟Linux登录shell
```

```
#!/bin/bash
echo -n "login:"
read name
echo -n "password:"
read passwd
if [ $name = "cht" -a $passwd = "abc"  ]\tauhen
  echo "the host and password is right!"
else
  echo "input is error!"
fi

# 比较两个数的大小
#!/bin/bash
echo "please enter two number"
read a
read b
if test $a -eq $b
  then echo "NO.1 = NO.2"
elif test $a -gt $b
  then echo "NO.1 > NO.2"
else echo "NO.1 < NO.2"
fi

# 查找/root/目录下是否存在该文件
#!/bin/bash
echo "enter a file name:"
read a
if test  -e /root/$a
  then echo "the file is exist!"
else echo "the file is not exist!"
fi

# for循环的使用
#!/bin/bash
clear
for num in 1 2 3 4 5 6 7 8 9 10
do
  echo "$num"
done

# 查看是否是当前用户
```

```
#!/bin/bash
echo "Please enter a user:"
read a
b=$(whoami)
if test $a = $b
  then echo "the user is running."
else echo "the user is not running."
fi

# 删除当前目录下大小为0的文件
#!/bin/bash
for filename in ls
do
  if test -d $filename
    then b=0
  else
    a=$(ls -l $filename | awk '{ print $5 }')
    if test $a -eq 0
      then rm $filename
    fi
  fi
done

# 普通无参数函数
#!/bin/bash
p ( ) {
  echo "hello"
}
p

# 给函数传递参数
#!/bin/bash
p_num ( ) {
  num=$1
  echo $num
}
for n in $@
do
  p_num $n
done

```

```
84 # 创建文件夹
85 #!/bin/bash
86 while :
87 do
88   echo "please input file's name:"
89   read a
90   if test -e /root/$a
91   then
92     echo "the file is existing Please input new file name:"
93   else
94     mkdir $a
95     echo "mkdir sussesful!"
96     break
97   fi
98 done
99
100 # 获取本机IP地址
101 #!/bin/bash
102 ifconfig | grep "inet addr:" | awk '{ print $2  }'| sed 's/addr://g'
103
104 # 查找最大文件
105 #!/bin/bash
106 a=0
107 for name in *.*
108 do
109   b=$(ls -l $name | awk '{print $5}')
110   if test $b -gt $a
111   then a=$b
112     namemax=$name
113   fi
114 done
115 echo "the max file is $namemax"
116
117 # case语句练习
118 #!/bin/bash
119 clear
120 echo "enter a number from 1 to 5:"
121 read num
122 case $num in
123   1) echo "you enter 1"
124   ;;
```

```
  2) echo "you enter 2"
  ;;
  3) echo "you enter 3"
  ;;
  4) echo "you enter 4"
  ;;
  5) echo "you enter 5"
  ;;
  *) echo "error"
  ;;
esac

# yes/no返回不同的结构
#!/bin/bash
clear
echo "enter [y/n]:"
read a
case $a in
  y|Y|Yes|YES) echo "you enter $a"
  ;;
  n|N|NO|no) echo "you enter $a"
  ;;
  *) echo "error"
  ;;
esac

# 内置命令的使用
#!/bin/bash
clear
echo "Hello, $USER"
echo "Today's date id date"
echo "the user is :"
who
echo "this is uname -s"
echo "that's all folks! "

# 输入分数显示好坏
#!/bin/bash
echo "Please enter score: "
read score
if [ $score -lt 80  ]
```

```
then
  echo "bad!!"
elif [ $score -ge 80 -a -lt 90  ]
then
  echo "good!!"
else
  echo "very good!!"
fi

# 编辑服务列
#!/bin/bash
echo "Services: "
echo -n "1) ls "
echo -n "2) ls -l"
echo -n :3) exit"
echo "Please choice [1-3]"
read choice
case $choice in
  1) ls
  ;;
  2) ls -l
  ;;
  3)exit
  ;;
  *) echo "wrong choice"
  ;;
esac
```

（九）shell 脚本的调试。

练习理论课中演示的 shell 脚本，并对所有脚本进行调试。

实验 9　Perl 脚本编程

一、实验要求

1. 掌握 Perl 脚本的语法结构与运行方法。
2. 掌握 Perl 读取写入文件的方法。
3. 了解 Perl 中获取标准输入的方法。
4. 熟练使用 Perl 的基本函数。
5. 掌握 Perl 中的判断语句和循环语句。

二、实验准备

1. 安装有 CentOS 7 的计算机。

三、实验内容

（一）用 Perl 编写的 “Hello World”。

1. 用文本编辑器创建一个名为 hello.pl 的文件，内容如下：

```
#!/usr/bin/perl -w
# Classic "Hello World" as done with Perl

print "Hello World!\n";
```

2. 保存文件、退出编辑器之后，修改文件的权限并运行。

（二）在 Perl 中打开文件和目录。

1. 在主目录中用文本编辑器创建一个名为 file.pl 的文件，内容如下：

```
#!/usr/bin/perl -w
# Create a file that will have a directory listing of user's
    home dir

print "About to read user's home directory\n";
# open the home directory and read it
opendir(HOMEDIR, ".");
@ls = readdir HOMEDIR;
closedir(HOMEDIR);

print "About to creat file dirlist.txt with a directory listing
```

```
    of user's home dir\n";
# open a file and write the directory listing to he file
open(FILE, ">dirlist.txt");
foreach $item (@ls) {
  print FILE $item . "\n";
}
close(FILE);
print "All done\n\n";
```

2. 保存文本、退出编辑器之后，修改文件的权限并运行。
3. 一旦脚本打印出“All done”，请找到 dirlist.txt 文件并用 cat 命令显示它的内容。

（三）检查来自标准输入的输入。

1. 在文本编辑器中，输入以下脚本，把它保存为 check.pl，并修改权限使得该脚本可执行。

```
#!/usr/bin/perl -T
# check.pl
# A Perl script that checks the input to determine if it
   contains numeric information

# First set up your Perl environment and import some useful
   Perl packages
use warings;
use diagnostics;
use strict;
my $result;

# Next check for any input and then call the proper subroutine
   based on the test
if (@ARGV) {
  # If the test condition for data from standard in is true run
     the test subroutine on the input data
  $result = &test;
} else {
  # else the test condition is false and you should run an
    error routine.
  &error;
}

# Now print the results
print $ARGV[0] . " has " . $result . " elements\n\n";
exit(0);

```

```
24 sub test{
25   # Get the first array element form the global array ARGV
26   # assign it to a local scalar and then test the local
       variable with a regular expression
27   my $cmd_arg = $ARGV[0];
28   if ($cmd_arg =~ m/[0-9]) {
29     return ('numeric');
30   } else {
31     # There was a error in the input; generate an error message
32     warn "Unknown input";
33   }
34 }
35
36 sub error {
37   # There was no input; generate an error message and quit
38   die "Usage: check.pl, (<INPUT TO CHECK>)";
39 }
```

2. 在命令行使用不同的参数或者不带任何参数运行该脚本。注意脚本输出的不同文本行取决于提供了什么样的参数：

```
1 ./check.pl
2 ./check.pl 42
3 ./check.pl h2g242
4 ./check.pl hg
```

（四）Perl 的函数与脚本练习。

练习理论课中演示的 Perl 脚本，并尝试对脚本进行调试。