Linux 系统概论

上机指南

伊现富　编著

天津医科大学

2019 年 4 月 27 日

目　录

实验 0　Markdown 标记语言　1

实验 1　远程登录 Linux 服务器　5

实验 2　体验（虚拟机中的）Linux　7

实验 3　Linux 图形界面的基本操作　9

实验 4　Linux 命令行界面的基本操作　10

实验 5　Linux 常用命令的操作　16

实验 6　Linux 高级命令的操作　22

实验 7　Linux 中的软件管理　30

实验 8　Vim 编辑器的基本操作　34

实验 9　shell 脚本编程　38

实验 0　Markdown 标记语言

Markdown 是一种轻量级标记语言，创始人为约翰·格鲁伯 *(John Gruber)*。它允许人们“使用易读易写的纯文本格式编写文档，然后转换成有效的 *XHTML*（或者 *HTML*）文档”。

Markdown 的目标是实现“易读易写”，成为一种适用于网络的书写语言。一份使用 *Markdown* 格式撰写的文件应该可以直接以纯文本发布，并且看起来不会像是由许多标签或是格式指令所构成。

Markdown 的语法全由一些符号所组成，这些符号经过精挑细选，其作用一目了然。比如：在文字两旁加上星号，看起来就像强调。*Markdown* 的列表看起来，嗯，就是列表。*Markdown* 的区块引用看起来就真的像是引用一段文字，就像你曾在电子邮件中见过的那样。

一、实验要求

1. 了解 Markdown 标记语言。
2. 掌握 Markdown 基本语法。
3. 使用 Markdown 撰写文档。

二、实验准备

1. 一台已安装有 Windows 操作系统或联网的计算机。
2. 一个 Markdown 编辑器。

三、实验内容

（一）Markdown 编辑器

1. 在线 Markdown 编辑器
    - Cmd Markdown（作业部落）
    - 马克飞象
    - 小书匠
    - StackEdit
2. 单机 Markdown 编辑器
    - Typora
    - Cmd Markdown 客户端
    - REMARKABLE
    - ghostwriter

（二）Markdown 基本语法

（三）Markdown 实例

（四）Markdown 实践

使用 Markdown 撰写实验报告（日记、作业……）。

Markdown Cheat Sheet

Format Text

Headers

```
# This is an <h1> tag
## This is an <h2> tag
###### This is an <h6> tag
```

Text styles

```
*This text will be italic*
_This will also be italic_
**This text will be bold**
__This will also be bold__

*You **can** combine them*
```

Lists

Unordered

```
* Item 1
* Item 2
  * Item 2a
  * Item 2b
```

Ordered

```
1. Item 1
2. Item 2
3. Item 3
   * Item 3a
   * Item 3b
```

Miscellaneous

Images

```
![GitHub Logo](/images/logo.png)
Format: ![Alt Text](url)
```

Links

```
http://github.com - automatic!
[GitHub](http://github.com)
```

Blockquotes

```
As Kanye West said:

> We're living the future so
> the present is our past.
```

Code Examples in Markdown

Syntax highlighting with GFM

```
```javascript
function fancyAlert(arg) {
  if(arg) {
    $.facebox({div:'#foo'})
  }
}
```
```

Or, indent your code 4 spaces

```
Here is a Python code example
without syntax highlighting:

    def foo:
      if not bar:
        return true
```

Inline code for comments

```
I think you should use an
`<addr>` element here instead.
```

Paragraphs and Line Breaks

- Para = one or more consecutive lines
- Empty lines = end of paragraph
- > 2 spaces at EoL = HTML line break

Headers

- # This is an H1
- ## This is an H2
- ###### This is an H6

Alternative: ==s for H1, --s for H2

Emphasis

- *This is an <em>*
  and _so is this_
- **This is a <strong>**
  and __so is this__
- Use same closing marker as opening
- Emphasis possible in middle of word
- * or _ for a literal * or _

Blockquotes

- Prefix > for each blockquote line...
  ... or even hard-wrapped paragraph
- Additional > for nested blockquotes
- Blockquotes can contain Markdown

Images

- ![Alt text](URL "Title")
- ![Alt text][id]
  [id]: URL "Title"
- See syntax for Links

Links

Inline	★ This is [an example](http://example.com/) link ★ This is [example](http://example.com/ "Title") with a title
Reference	★ This is a [reference link][id] with an id [id]: http://example.com/ "Title" ★ This is a [reference link][] without an id [reference link]: http://example.com/ (Title)

- Link id can be letters, numbers, spaces, and punctuation
- Link id is case-insensitive
- In ref links, link definition can be indented up to 3 spaces
- In ref links, use parentheses, single or double quotes for title
- In ref links, link URL can be surrounded by < >
- In ref links, title can be on separate line with indentation
- In ref links, link definitions can appear anywhere in document

Lists

Unordered	Asterisks, pluses & hyphens interchangeably	+ Red * Green - Blue
Ordered	Numbers, not necessarily sequential. Always start with 1.	1. Red 1. Green 1. Blue

- List marker must be followed by 1/more spaces or a tab
- List marker can be indented upto 3 spaces
- Hanging indent supported but not required
- Blank lines between list items implies paragraph
- Paragraph start must be indented by 4 spaces or a tab
- To avoid unintended list numbering (e.g 1986. What a great season) backslash the period (1986\. What a great season).

Markdown

➔ http://daringfireball.net/projects/markdown/
Cheat sheet from Ahren Code ➔ http://ahren.org/code/

Code Block

- Indent code by 4 spaces or 1 tab
- One level indentation will be removed
- & and <, > are automatically HTML'ised
- Markdown not usable in code blocks

Code Spans

- Use backticks around code: `printf()`
- Use multiple backticks for literal backtick: ``ls `date+%h` | wc -l``
  - Use space if literal backtick is at start of code span: `` ` ``

Miscellaneous

Horizontal Rule / Line
3 or more asterisks, hyphens or underscores

List with embedded Blockquote
Indent > delimiters:

```
* A list item with a blockquote:
  > This is a blockquote
```

List with embedded Code block
Indent code block 8 spaces or two tabs:

```
* A list item with a code block:
    ... some code ....
```

Automatic Links
- Use <, > for auto-linking URLs
- Use <, > for auto-linking email addrs
- Email text is auto obscured (somewhat)

For all else use HTML markup
- blank lines around block level elements
- no indentation for tags
- no Markdown inside HTML blocks

syntax

```
Plain text
End a line with two spaces to start a new paragraph.
*italics* and _italics_
**bold** and __bold__
superscript^2^
~~strikethrough~~
[link](www.rstudio.com)

# Header 1

## Header 2

### Header 3

#### Header 4

##### Header 5

###### Header 6

endash: --
emdash: ---
ellipsis: ...
inline equation: $A = \pi*r^{2}$
image: ![](path/to/smallorb.png)

horizontal rule (or slide break):

***

> block quote

* unordered list
* item 2
    + sub-item 1
    + sub-item 2

1. ordered list
2. item 2
    + sub-item 1
    + sub-item 2

Table Header  | Second Header
------------- | -------------
Table Cell    | Cell 2
Cell 3        | Cell 4
```

becomes

Plain text
End a line with two spaces to start a new paragraph.
italics and *italics*
bold and **bold**
superscript^2
~~strikethrough~~
link

Header 1

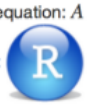
Header 2

Header 3

Header 4

Header 5

Header 6

endash: –
emdash: —
ellipsis: …
inline equation: $A = \pi * r^2$
image: 
horizontal rule (or slide break):

> block quote

- unordered list
- item 2
  - sub-item 1
  - sub-item 2

1. ordered list
2. item 2
   - sub-item 1
   - sub-item 2

Table Header	Second Header
Table Cell	Cell 2
Cell 3	Cell 4

```
#The Properties of Water

## Special considerations
The grade level of this lesson plan is prospective and
practicing **K-8 teachers**
> May be adapted for use in elementary classes.

### To ponder

Water is everywhere. It's in the air we breathe. It's in our
sink faucets, and it's in every cell of our body. Water is an
unusual substance with special properties. Just think about the
wonder of water:

1.  How does water rise from the roots of a redwood tree to the
very top?
2.  How do insects walk on water?
3.  Why does ice float rather than sink?
4.  Why do people become seriously ill, or die, if they go
without liquid for a week or so?


**In this first lab, we will investigate the properties of
water in an attempt to understand how water behaves in relation
to both our bodies and the environment. Through a concise set
of experiments, the unique properties of water and its
consequent importance to living things will become apparent.**

### Supplies

* chromatography paper strips
* detergent
* vis-a-vis black ink pens


###Objectives
```

The Properties of Water

Special considerations

The grade level of this lesson plan is prospective and practicing **K-8 teachers**

> May be adapted for use in elementary classes.

To ponder

Water is everywhere. It's in the air we breathe. It's in our sink faucets, and it's in every cell of our body. Water is an unusual substance with special properties. Just think about the wonder of water:

1. How does water rise from the roots of a redwood tree to the very top?
2. How do insects walk on water?
3. Why does ice float rather than sink?
4. Why do people become seriously ill, or die, if they go without liquid for a week or so?

In this first lab, we will investigate the properties of water in an attempt to understand how water behaves in relation to both our bodies and the environment. Through a concise set of experiments, the unique properties of water and its consequent importance to living things will become apparent.

Supplies

- chromatography paper strips

Word Count: 400　　Template: Expresso

实验 1　远程登录 Linux 服务器

通常我们可以将服务器看做一台配置功能强大的超级电脑，它也有自己独立的操作系统，其中核心系统以 *Linux* 系统为主的服务器，我们都可以称为 *Linux* 服务器。

一、实验要求

1. 熟悉远程登录 Linux 服务器的工具。
2. 掌握远程登录 Linux 服务器的方法。
3. 掌握登录和退出 Linux 的基本命令。

二、实验准备

1. 一台已安装有 Linux 操作系统的（远程）服务器。
2. 一台安装有 Windows（或/和 Linux）操作系统的台式机。

三、实验内容

（一）使用 Windows 操作系统远程登录 Linux 服务器

1. 检索可以使用的远程登录工具。
2. 检索远程登录工具的使用方法并进行练习与使用。
3. 尝试对远程登录工具进行个性化配置。
4. 尝试与远程服务器进行文件交互。

为了快速、顺利完成检索与学习，请参看以下提示：

- 远程登录：PuTTY；MobaXterm；等。
- 文件交互：FileZilla；等。
- 工具选择：首选开源工具，次选收费工具，**不要使用破解软件**。
- 工具下载：**一定从官网下载软件！**
- 检索工具：引擎首选**谷歌/必应**，次选百度等；百科首选**维基百科**，次选百度百科等。
- 检索策略：关键词首选**英文**，次选中文。

（二）使用 Linux 操作系统远程登录 Linux 服务器

在本地 Linux 操作系统中找到终端（Terminal），之后尝试登陆、退出远程 Linux 服务器。

1. 登录命令：ssh
2. 退出命令：logout, exit

登录远程 Linux 服务器时，思考以下问题（提示：从安全的角度进行思考）：

- 登录前会有用户名或者密码的提示吗？
- 输入密码时与常规的登录网站有何不同？为什么这么“反人类”？
- 如果登录失败会有失败信息的提示吗？为什么这么“反人类”？

- 登录后看到了哪些基本信息？这些信息有什么作用？

实验 2　体验（虚拟机中的）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. 了解在 VirtualBox/VMware 中安装 Linux 的步骤。
2. 启动 Linux 并进行初始化设置。
3. 登录不同 Linux 发行版的桌面环境，了解不同发行版的图形化用户界面。
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/VMware 软件的计算机。
2. VirtualBox/VMware 中已安装多个 Linux 发行版。

三、实验内容

（一）了解虚拟机和 Linux 发行版

1. 通过检索了解虚拟机的基本概念及常用虚拟机软件。
2. 通过检索了解常见的 Linux 发行版。
3. 通过检索进入各个 Linux 发行版的官网，熟悉其界面。
4. 通过检索了解 Linux 发行版的排名。

为了快速、顺利完成检索，请参看以下提示：

- 检索工具：引擎首选**谷歌/必应**，次选百度等；百科首选**维基百科**，次选百度百科等。
- 检索策略：关键词首选**英文**，次选中文。
- 发行版排名：**DistroWatch** 网站。

（二）Linux 发行版（在虚拟机中）的安装与启动

1. 在虚拟机中安装 Linux（Ubuntu/CentOS 等）。

   检索 Virtuabox/VMware 的使用方法，以及在其中安装 Linux 的图文教程，详细阅读，理解主要的安装步骤。

2. 启动 Linux。

   (a) 启动 Linux 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 Linux 桌面环境。

3. 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 Linux 桌面环境，屏幕再次显示登录界面，等待新用户登录系统。

4. 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。

（三）体验其他 Linux 发行版

在 VirtualBox/VMware 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，根据自己的喜好对各个发行版进行排名。

实验 3　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* 以 *Unity* 作为默认的桌面环境，*CentOS* 以 *GNOME* 作为默认的桌面环境。本实验以 *Ubunut/CentOS* 为基础，来练习 *Unity/GNOME* 等常见桌面环境的基本操作。

一、实验要求

1. 掌握 Linux 桌面环境的设置方法。
2. 掌握文件浏览器的使用方法。
3. 了解其他的 Linux 桌面环境。

二、实验准备

1. 安装有 Ubuntu/CentOS 等 Linux 发行版的计算机。

三、实验内容

（一）总结常见的桌面环境操作和文件浏览器操作

回顾使用 Windows 操作系统的过程，总结常见的桌面环境操作和文件浏览器操作：

- 桌面环境操作举例：修改桌面背景、设置屏幕保护程序、设置主题、修改桌面图标、……
- 文件浏览器操作举例：新建/复制/移动/删除文件/文件夹、修改文件徽标、创建快捷方式/链接、查看隐藏文件、……

（二）Linux 中的桌面环境操作

尝试在 Linux 的桌面环境下重现 Windows 操作系统中的常见桌面环境操作。

（三）Linux 中的文件浏览器操作

尝试在 Linux 的文件浏览器中重现 Windows 操作系统中的常见文件浏览器操作。

（四）体验其他桌面环境

在 VirtualBox/VMware 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 4　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 *Ubuntu/CentOS* 等单个的 *Linux* 发行版，而且几乎有助于掌握各种发行版本的 *Linux*，甚至 *Unix*。

包括 *Ubuntu/CentOS* 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。

Ubuntu/CentOS 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. 掌握软链接和硬链接的使用与区别。
4. 了解可读、可写和可执行权限对文件和目录的区别。

二、实验准备

1. 安装有 Ubuntu/CentOS 的计算机。

三、实验内容

(一) 图形化用户界面下的 shell 命令操作

启动计算机，登录图形化用户界面。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。

1. 显示系统时间。
   (a) 输入命令 `date`，显示系统的当前日期和时间。
2. 查看日历。
   (a) 输入命令 `cal 2018`，屏幕上显示出 2018 年的日历。
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

启动计算机后，默认进入 Linux 的图形化用户界面，此时按【Ctrl+Alt+F2】组合键切换到第一（CentOS）/二（Ubuntu）个虚拟终端（也可使用图形化界面中的终端）。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。

1. 查看当前目录。

(a) 输入命令 `pwd`，显示当前目录。

2. 操作文本文件。

(a) 创建文本文件。

首先，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.
```

具体操作步骤如下:

i. 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。

ii. 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。

iii. 要查看文件是否生成，输入命令 `ls` 即可。

iv. 输入命令 `cat f1`，查看 f1 文件的内容。

(b) 添加内容。

然后，向 f1 文件增加以下内容：Why not have a try？具体操作步骤如下:

i. 输入命令 `cat >>f1`，屏幕上输入点光标闪烁。

ii. 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。

iii. 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。

知识点解析： 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。

(c) 统计文本文件信息。

之后，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。具体操作步骤如下：

i. 输入命令 `wc <f1 >countf1`，屏幕上不显示任何信息。

ii. 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。

(d) 合并文件。

最后，将 f1 和 countf1 文件合并为 f 文件。具体操作步骤如下：

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

ii. 输入命令 `cat f`，查看 f 文件的内容。

3. 查看目录。

(a) 分页显示 /etc 目录中所有文件和子目录的信息。

i. 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。

ii. 浏览过程中按【Q】键，可结束分页显示。

知识点解析：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。

(b) 仅显示 /etc 目录中的前 5 个文件和子目录。

i. 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。

4. 清除屏幕内容。

(a) 输入命令 `clear`，或者使用快捷键【Ctrl+L】，屏幕内容将完全被清除，命令提示符定位在屏幕左上角。

（三）链接操作

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）链接的本质

1. 创建原文件、硬链接和软链接。

(a) 使用 cd ~ 命令切换到主目录。

(b) 分别使用 touch fn、ln fn hl 和 ln -s fn sl 命令创建名为 fn 的原文件、名为 hl 的硬链接、名为 sl 的软链接。

(c) 使用 ls -li fn hl sl 查看各个文件的属性。（注意：fn、hl、sl 的 inode、链接数、文件大小；sl 的指向【除了 ls 的输出外，还可以使用 readlink sl 查看 sl 的指向】）。

(d) 使用 echo 123 > fn 向原文件添加内容。

(e) 使用 ls -li fn hl sl 查看各个文件的属性。（注意：fn、hl、sl 的 inode、链接数、文件大小的变化）。

(f) 使用 cat fn hl sl 查看各个文件的内容。

2.“不小心”删除原文件。

(a) 使用 rm fn 删除原文件

(b) 使用 ls -li hl sl 查看各个文件的属性。（注意：hl、sl 的 inode、链接数、文件大小的变化；sl 指向的有效性）。

(c) 使用 cat hl sl 查看各个文件的内容。

3.“碰巧”创建同名文件

(a) 使用 touch fn 创建一个和被删除文件同名的文件。

(b) 使用 ls -li fn hl sl 查看各个文件的属性。（注意：fn、hl、sl 的 inode、链接数、文件大小的变化；sl 指向的有效性）。

(c) 使用 cat fn、cat hl 和 cat sl 分别查看各个文件的内容。

(d) 使用 `echo abcdef > fn` 向新的文件添加内容。

(e) 使用 `ls -li fn hl sl` 查看各个文件的属性。（注意：fn、hl、sl 的 inode、链接数、文件大小的变化；sl 指向的有效性）。

(f) 使用 `cat fn hl sl` 查看各个文件的内容。

知识点解析：硬链接指向的是原文件所在的数据块，除了名字不同以外，本质上和原文件没有区别。软链接存储的仅仅是原文件的路径，并非实际的数据块。

（五）目录的可读、可写、可执行权限

1. 创建目录。

(a) 使用 `cd ~` 命令切换到主目录。

(b) 使用 `mkdir dir` 创建目录，使用 `touch dir/file1` 在目录中创建文件。

(c) 使用 `ls -ld dir` 查看所有者对目录的权限。

(d) 分别尝试 `ls dir`、`touch dir/file2` 和 `cd dir; cd ..`（注意命令的提示或者最终状态）。

2. 目录的可读权限。

(a) 使用 `chmod u-r dir` 删除所有者对目录的可读权限。

(b) 使用 `ls -ld dir` 查看所有者对目录的权限。

(c) 分别尝试 `ls dir`、`touch dir/file3` 和 `cd dir; cd ..`（注意命令的提示或者最终状态）。

(d) 使用 `chmod u=rwx dir` 恢复所有者对目录的原有权限。

(e) 使用 `ls -ld dir` 查看所有者对目录的权限。

3. 目录的可写权限。

(a) 使用 `chmod u-w dir` 删除所有者对目录的可读权限。

(b) 使用 `ls -ld dir` 查看所有者对目录的权限。

(c) 分别尝试 `ls dir`、`touch dir/file4` 和 `cd dir; cd ..`（注意命令的提示或者最终状态）。

(d) 使用 `chmod u=rwx dir` 恢复所有者对目录的原有权限。

(e) 使用 `ls -ld dir` 查看所有者对目录的权限。

4. 目录的可执行权限。

(a) 使用 `chmod u-x dir` 删除所有者对目录的可读权限。

(b) 使用 `ls -ld dir` 查看所有者对目录的权限。

(c) 分别尝试 `ls dir`、`touch dir/file5` 和 `cd dir`（注意命令的提示或者最终状态）。

(d) 使用 `chmod u=rwx dir` 恢复所有者对目录的原有权限。

(e) 使用 `ls -ld dir` 查看所有者对目录的权限。

知识点解析：目录的可读权限意味着可以使用 ls 列出目录的内容（就像文件的可读权限意味着可以读取文件的内容一样）。目录的可写权限意味着可以在目录中使用 touch 创建文件（也就是修改目录的内容，就像文件的可写权限意味着可以修改文件的内容的一样）。目录的可执行权限意味着可以使用 cd 切换到目录中去（同时也会对 ls 和 touch 等命令有所影响）。所以，目录一定会有可执行权限，绝大多数情况下也会同时有可读权限，换言之，目录应该至少同时有可读和可执行权限。

（六）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 5　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. 掌握重定向、管道、通配符、历史记录等的使用方法。
7. 掌握对文件进行排序的方法。
8. 掌握利用 shell 命令管理用户和组群的方法。
9. 理解 /etc/passwd 和 /etc/group 文件的含义。

二、实验准备

1. 安装有 Ubuntu/CentOS 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) 使用 `find /etc -name profile`命令进行查找，屏幕显示已找到 /etc/profile 文件。

知识点解析：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，可以先输入命令 `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。要切换回普通用户，使用 `exit` 命令，即可退出超级用户身份。

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 2017 年 1 月 1 日以后有过更新的文件，打包压缩到 1701new.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。（如果不想手动输入内容，也可以通过命令 `man ls > test-file` 把 ls 的联机帮助页保存到文件中。）
   (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）通配符的使用

知识点解析：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。

知识点解析：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。

知识点解析：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。

（六）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`。

（七）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. 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）。
3. 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。
4. 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。
5. 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。
6. 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。

7. 删除 newgroup 组。

(a) 超级用户输入命令 `groupdel newgroup`，删除 newgroup 组。

实验 6　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 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 Ubuntu/CentOS 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命令的目录
mkdir find_test

# 切换至该目录
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. 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 校验和：
   `find -iname "MyCProgram.c" -exec md5sum {} \;`。`{}`将会被当前文件名取代。
4. 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。
5. 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`。
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 -l 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 -il 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 {} \;`。（仔细比较该命令和上一个命令的区别）

(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" -exec 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 "\n"}
```

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

（七）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。

（八）【课外扩展】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{$_}})} sort keys %a);}'
19
20 # Collapse unique values in each group:
21 echo "$DATA" | datamash -s -g 1 unique 2
22 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/#exam-

ple_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)，⑩ exonStarts，⑪ exonEnds，⑫ score，⑬ GeneName (gene identifier)，⑭ cdsStartStat，⑮ cdsEndStat，⑯ exonFrames。

(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
```

实验 7　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 bioconda 的安装、配置与使用。
3. 了解使用 APT 与 Yum 管理软件的方法。
4. 了解使用 dpkg 与 RPM 管理软件的方法。
5. 掌握通过源代码安装软件的步骤。
6. 熟悉其他安装软件的方法。

二、实验准备

1. 安装有 Ubuntu/CentOS 的计算机。

三、实验内容

（一）在命令行中下载安装包

为便于后续软件的安装，使用 wget 和/或 curl 在命令行中下载 Miniconda、datamash、htop 和 Glances 的安装包。通过查阅 wget 和 curl 的帮助页，掌握它们的使用方法。（提示：主要是 wget 的 `-c` 选项和 curl 的 `-o` 和 `-O` 选项）

（二）使用 bioconda 管理生物信息学软件

0. 请先通读该文章：conda 管理生信软件一文就够！
1. 下载 Miniconda/Anaconda（此处以 Miniconda 为例）。主要参考网址为：https://mirrors.tuna.tsinghua.edu.cn/help/anaconda/。

```
# 从国外官网下载
#wget -c https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
# 从国内（清华）镜像下载
wget -c https://mirrors.tuna.tsinghua.edu.cn/anaconda/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

提示：从国外官网上下载 Miniconda 的速度很慢，建议通过国内镜像进行下载。另外，在真正使用 conda 之前，也强烈建议首先配置国内镜像，详细方法参看后文。

2. 安装 Miniconda。

```
bash Miniconda3-latest-Linux-x86_64.sh
```

提示：理解安装过程中交互式步骤的具体含义，尤其是最后配置 PATH 环境变量的步骤要选择 yes（如果选择了 no，请按照提示进行手动配置）。

3. 使更新后的 PATH 环境变量生效。下面两种方法任选其一即可：
   - `source ~/.bashrc`
   - 重开一个终端或者重登服务器
4. 设置镜像/添加 channels。

```
# free and main
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/
# conda-forge
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge/
# r
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r/
# bioconda
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/bioconda/
conda config --set show_channel_urls yes
```

提示：conda 的配置文件是`$HOME/.condarc`，可以直接编辑该文件添加 channels 或者调整 channel 的顺序。（也可以使用 conda 命令间接修改该文件。）

5. 使用 conda 管理软件。

```
# 查看使用帮助
conda -h
# 安装软件
conda install ucsc-fasize
# 使用软件（查看软件使用帮助）
faSize
# 删除软件
conda remove ucsc-fasize
```

（三）通过 APT 与 Yum 安装软件

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

```
# 需要使用sudo
apt-get install dos2unix

```

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

（四）通过 dpkg 与 RPM 安装软件

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

```
# 需要预先使用wget或curl下载deb和/或rpm包
# 需要使用sudo
dpkg -i datamash_1.0.6-1_amd64.deb

# 需要切换为root用户
rpm -ivh datamash-1.0.6-1.el6.x86_64.rpm
```

（五）通过源代码安装软件

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

```
# 需要预先使用wget或者curl下载htop的tar.gz源代码包
# htop
tar -zxvf htop-1.0.3.tar.gz
cd htop-1.0.3/
#vim INSTALL
#vim README
./configure
make
make install
```

（六）通过脚本安装软件

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

```
# 需要预先下载Glances的zip源代码包
# Glances
unzip Glances-master.zip
cd Glances-master
#vim README.rst
python setup.py install
```

（七）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 ~
```

```
git clone https://github.com/galaxyproject/galaxy
cd galaxy
git checkout -b master orgin/master
```

（八）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。

```
# 需要预先下载CPU-G的tar.gz源代码包
# CPU-G
tar -zxvf cpu-g-0.9.0.tar.gz
cd cpu-g-0.9.0/
chmod 755 cpu-g
./cpu-g
```

（九）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 8　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）Vim 命令概览

1. 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。
2. 目前处在命令模式下，通过输入【i】切换到插入模式。
3. 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.
```

4. 按下【Esc】键返回到命令模式。
5. 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。
6. 输入 `cw`，该命令将删除单词 quick 并切换到插入模式。
7. 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.
```

8. 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。
9. 输入 r，然后输入 E。该句将变成:

```
SentencE three.
```

10. 输入 k 使游标上移一行。输入 2yy 复制两行。
11. 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.
```

12. 按下【Esc】键，然后输入 :q! 退出该文件（! 没有保存输入!）。

（二）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【:】键进入末行模式，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。**知识点解析：**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。

（三）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【:】键切换到末行模式，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“:”，进入末行模式。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) 进入末行模式，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，再次进入末行模式，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，进入末行模式，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，进入末行模式，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，进入末行模式，输入 `r /etc/passwd`，在光标所在处读入 /etc/passwd 文件的内容。
   (c) 按【:】键，进入末行模式，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。**知识点解析：**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。

（三）强化练习 Vim 的常用操作

1. 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）。
2. 在线交互式学习 Vim 的基本操作：http://www.openvim.com/tutorial.html。
3. 在线练习 Vim 的常用命令：http://www.vimgenius.com/。
4. 观看 Vim 操作视频：http://derekwyatt.org/。
5. 在游戏中学习 Vim：http://vim-adventures.com/。

实验 9　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 Ubuntu/CentOS 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/bash

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

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 "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）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。

```
1 #!/bin/bash
2
3 # array
4 # A script for polulating an array from a file
5
6 populated=($(cat ~/tmp/sports.txt | tr '\n' ' '))
7 echo ${populated[@]}
8
9 exit 0
```

3. 使脚本变成可执行文件。
4. 在命令行中运行该脚本。

（六）shell 脚本实例

```
1 # 模拟Linux登录shell
2 #!/bin/bash
3 echo -n "login:"
4 read name
```

```
5  echo -n "password:"
6  read passwd
7  if [ $name = "cht" -a $passwd = "abc" ]
8  then
9    echo "the host and password is right!"
10 else
11   echo "input is error!"
12 fi
13
14 # 比较两个数的大小
15 #!/bin/bash
16 echo "please enter two number"
17 read a
18 read b
19 if (test $a -eq $b)
20 then
21   echo "NO.1 = NO.2"
22 elif (test $a -gt $b)
23 then
24   echo "NO.1 > NO.2"
25 else
26   echo "NO.1 < NO.2"
27 fi
28
29 # 查找/root/目录下是否存在该文件
30 #!/bin/bash
31 echo "enter a file name:"
32 read a
33 if (test  -e /root/$a)
34 then
35   echo "the file is exist!"
36 else
37   echo "the file is not exist!"
38 fi
39
40 # for循环的使用
41 #!/bin/bash
42 clear
43 for num in 1 2 3 4 5 6 7 8 9 10
44 do
45   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

# 创建文件夹
#!/bin/bash
while :
do
  echo "please input file's name:"
  read a
  if (test -e /root/$a)
  then
    echo "The file is existing! Please input new file name!"
  else
    mkdir $a
    echo "mkdir sussesful!"
    break
  fi
done

# 获取本机IP地址
#!/bin/bash
ifconfig | grep "inet addr:" | awk '{ print $2  }'| sed 's/addr://g'

# 查找最大文件
#!/bin/bash
a=0
for name in *.*
do
  b=$(ls -l $name | awk '{print $5}')
  if (test $b -gt $a)
  then a=$b
    namemax=$name
  fi
done
echo "the max file is $namemax"

# case语句练习
```

```
#!/bin/bash
clear
echo "enter a number from 1 to 5:"
read num
case $num in
  1) echo "you enter 1"
  ;;
  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 is $(date)"
echo "the user is :"
who
echo "this is $(uname -s)"
```

```
169 echo "that's all folks!"
170
171 # 输入分数显示好坏
172 #!/bin/bash
173 echo "Please enter score: "
174 read score
175 if [ $score -lt 80  ]
176 then
177   echo "bad!!"
178 elif [ $score -ge 80 -a $score -lt 90  ]
179 then
180   echo "good!!"
181 else
182   echo "very good!!"
183 fi
184
185 # 编辑服务列
186 #!/bin/bash
187 echo "Services: "
188 echo -n "1) ls "
189 echo -n "2) ls -l"
190 echo -n "3) exit"
191 echo "Please choice [1-3]"
192 read choice
193 case $choice in
194   1) ls
195   ;;
196   2) ls -l
197   ;;
198   3)exit
199   ;;
200   *) echo "wrong choice"
201   ;;
202 esac
203
204 # Fork炸弹：总共只用了13个字符（包括空格）
205 # 警告：切勿尝试，后果自负!
206 :(){ :|:& };:
207 # 注解如下
208 :()       # 定义函数，函数名为":"，即每当输入":"时就会自动调用{}内的代码
209 {         # ":"函数起始字符
```

```
:          # 用递归方式调用":"函数本身
|          # 并用管道(pipe)将其输出引至...(因为有一个管道操作字符，因此会生成一个新的进程)
:          # 另一次递归调用的":"函数
# 综上，":|:"表示的即是每次调用函数":"的时候就会产生两份拷贝
&          # 调用后台运行字符，以使最初的":"函数被关闭后为其所调用的两个":"函数还能继续执行
}          # ":"函数终止字符
;          # ":"函数定义结束后将要进行的操作...
:          # 调用":"函数，"引爆"fork炸弹
```

（七）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。