Lab01: 基础命令 & vi 操作 实验指导

1. 实验目的

- 初步认识Linux系统，了解Linux的历史。
- （重点）学习使用Linux字符界面（命令行界面）命令。
- （重点）上手 vi（vim）编辑器，为日后熟练使用打好基础。

2. 实验内容

- Linux常用各种命令（《Linux编程基础》1.4节）
    - man及info帮助命令
    - 文件系统管理命令（重点）
    - 系统及用户管理命令
    - 磁盘管理命令
    - 网络管理命令
    - 进程管理命令
    - 通配符的使用
- vim编辑器（《Linux编程基础》1.5节）
    - 掌握vim基础命令，了解三种模式，掌握打开、编辑、保存、退出等操作。
    - 了解vim高阶操作，提高vim使用效率。

3. 实验指南

3.1 常用命令

> *本章列出了比较常用的命令，掌握这些命令可以帮助大家以一个用户（非管理员）的角度比较顺畅的使用Linux系统*

Linux 命令是对 Linux 系统进行管理的工具。虽然现在的操作系统大多搭载了图形用户界面(GUI)，但是在比如说服务器上，一般只有命令行用户界面(CLI)，所以掌握一些常用的命令行操作是十分实用且有必要的。

> *安装命令*
>
> ```
> # ubuntu/debian
> sudo apt-get install package # 安装包
> sudo apt-get remove package # 删除包 比如apt-get install vim 安装vim
> sudo apt-get update # 更新源
>
> # centos 请自查yum命令
> ```

> *管理员root权限*
>
> *管理员身份执行命令* `sudo 命令`
>
> *进入管理员模式* `sudo -s`，`exit` *退出*

3.1.1 man及info帮助命令（包含2个命令）

(1) **man** manual的缩写，用于获得命令的帮助，常用用法：`man [参数选择] [节号] 命令`，如`man 2 write`，表示从man文档的第二节中查找write的帮助。

(2) **info** 用于获得命令的信息，常用用法：`info 命令`，如`info gcc`，表示查看gcc命令的信息。

3.1.2 文件系统管理命令（包含12个命令）

> *Linux 操作系统中存在着两种路径：绝对路径和相对路径。我们在访问文件或文件夹的时候，其实都是通过路径来操作的。两种路径在实际操作中能起到同等的作用。*
> *绝对路径是相对于根文件夹的。它们的标志是以“/”开头。*
> *相对路径是相对于我们所处的文件夹位置。它们的第一个字符没有“/”。*
> *比如，在`/a/b/`路径下，`c.txt`相对路径，代表的是`/a/b/c.txt`文件，而`a/b/c.txt`文件是该文件的绝对路径。*

> *在Linux命令中，使用相对路径和绝对路径会有不同的效果。*

(1) **ls** list的缩写。用于显示文件信息，常用用法：`ls [选项] [文件名/目录名]`，如`ls -a /`，表示查看`/`目录下所有的文件。

> *如果没有`-a`参数，不会显示隐藏文件。Linux的隐藏文件以.开头，比如.gitignore。*

(2) **pwd** 用于显示用户当前所在的目录，常用用法：`pwd`。

(3) **cd** change directory的缩写。用于切换目录。用法：`cd 目录名`，比如`cd /`，代表进入`/`目录，`cd ..`，进入上一级目录。

> *在Linux中，`.`代表当前目录，`..`代表上一级目录，`~`代表家目录。*

> *`cd -`，可以返回上一个所在的目录。*

(4) **mkdir** make directory的缩写。用于建立目录，常用用法：`mkdir [参数] 目录名`，比如`mkdir -p a/b/c`，`-p`代表递归创建不存在的目录，比如在空目录下，该命令自动创建`a/`，`a/b/`，`a/b/c/`三个目录。

(5) **rmdir** remove directory的缩写。用于删除目录，常用用法：`rmdir [参数] 目录名`，在`mkdir -p a/b/c`执行成功后，使用`rmdir -p a/b/c`会依次删除`/a/b/c`，`a/b`，`a`三个目录。

(6) **rm** remove的缩写。用于删除文件，常用用法：`rm 文件名`。

(7) **cp** copy的缩写。用于复制文件/目录，常用用法：`cp 源文件 目标文件` `cp -r 源目录 目标目录`。

(8) **mv** move的缩写。用于移动文件/目录，常用用法：`mv 源文件/源目录 目标文件/目标文件`。

(9) **cat** 用于查看文件内容，比如：`cat a.txt`。

> *`head`/`tail` 命令 可以用查看文件首/尾的内容。用法详见`man head`/`man tail`。*

(10) **grep** 打印匹配字符的行。比如：`cat a.txt | grep abc`，则可以找出a.txt中包含abc字符串的行。

(11) **more**

more功能类似 cat ，cat命令是整个文件的内容从上到下显示在屏幕上。 more会以一页一页的显示方便使用者逐页阅读，而最基本的指令就是按空白键（space）就往下一页显示，按 b 键就会往回（back）一页显示，而且还有搜寻字串的功能 。more命令从前向后读取文件，因此在启动时就加载整个文件。

(12) **less**

less 工具也是对文件或其它输出进行分页显示的工具，应该说是linux正统查看文件内容的工具，功能极其强大。less 的用法比起 more 更加的有弹性。 在 more 的时候，我们并没有办法向前面翻， 只能往后面看，但若使用了 less 时，就可以使用 [pageup] [pagedown] 等按 键的功能来往前往后翻看文件，更容易用来查看一个文件的内容！除此之外，在 less 里头可以拥有更多的搜索功能，不止可以向下搜，也可以向上搜。

> *在线上查看超大文件时，比如日志，严谨使用cat、vim命令查看会占用大量内存导致线上崩溃，推荐使用less命令。*

(13) 压缩/解压缩命令

`zip`/`unzip` 用法：`zip [-r] [压缩后文件名称] 文件或目录`与`unzip [选项] 压缩包包名`。`-r`代表压缩的是一个目录，压缩时会保留源文件。

自学`tar`命令等其他命令。

(14) **awk**（自学）

功能强大的数据处理工具。

3.1.3 系统及用户管理命令（包括3个命令）

(1) **shutdown** 关机。常用用法：`shutdown -P`，一分钟后关机；`shutdown -c`取消关机。

(2) **date** 查看当前日期时间。

(3) **who** 查看当前登录的用户。常用用法：`who`。

3.1.4 磁盘管理命令（包括2个命令）

(1) **du** 查看目录占用空间大小。 用法：`du [选项] [文件名称]`。

(2) **df** 显示磁盘的使用率及剩余空间。用法：`df [可选参数]`，比如`df -k` 显示系统所配置的每一个磁盘当前被占用的空间大小。

3.1.5 网络管理命令（包括3个命令）

(1) **ifconfig** 用途：查看当前网络配置，比如ip。

(2) **netstat** 用途：查看当前网络状态，比如`nestat -a`。

(3) **ping** 用途：查看是否能连通某域名（如果网站防ping，那么也会无法成功）比如`ping www.baidu.com`。

3.1.6 进程管理命令（包括3个命令）

(1) **ps** 显示当前的进程。常用用法：`ps [可选参数]`，比如`ps a`，查看所有用户的所有进程。常用方法：`ps aux`，参数请通过`man ps`查看。

(2) **kill** 杀死某个进程，比如`kill 12345`，表示杀死进程id为12345的进程。可能需要root权限才能成功，但是kill失败不会报错。

(3) **pstree** 以树的形式显示进程之间的父子关系。

3.1.7 通配符的使用 csdn

> *通配符的使用可以大大提升操作效率，对批处理等操作有非常大的帮助。*

(1) `?`可以代表任何字符。以下是例子。

```
root@server:/home/chen# ls
a2.txt  a.txt  b.txt
root@server:/home/chen# ls ?.txt
a.txt  b.txt
```

因为`?`只能代表一个字符，所以`a2.txt`无法显示。

(2) `*`可以代表任意长的字符。以下是例子。

```
root@server:/home/chen# ls
a2.txt  a.txt  b.txt
root@server:/home/chen# ls *.txt
a2.txt. a.txt  b.txt
```

同理，`ls *`相当于列出所有文件（隐藏文件除外）。

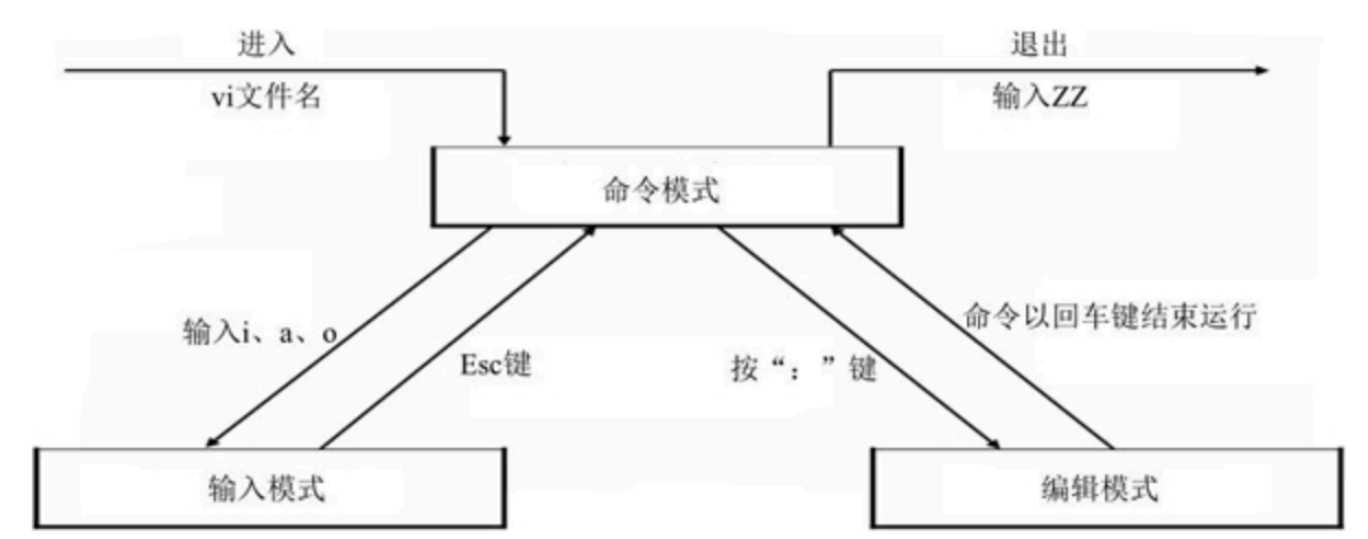
(3) [charset] 可替代charset集中的任何单个字符。以下是例子。

```
root@server:/home/chen# ls
a2.txt  a.txt  b.txt  c.txt  d.txt
root@server:/home/chen# ls [ab].txt
a.txt  b.txt
root@server:/home/chen# ls [a-c].txt
a.txt  b.txt  c.txt
```

`[a-c]`表示从`a`到`c`，即`abc`。

3.2 vim编辑器

MIT的一门课的一个lecture中非常系统的、彻底的讲解了vim的使用，可以帮助你在50分钟内快速学习vim。



> *如果没有安装`vim`，使用`sudo apt-get install vim`。*

3.2.1 基本用法

方向键（命令模式）：下`j`，上`k`，左`h`，右`l`。

进入编辑模式：`i` (insert)。可以在光标所在位置开始输入。

> *其他进入编辑模式的命令?*
>
> *`a` 在光标所在位置的下一个字符开始输入。*
>
> *`o` 另起一行开始输入。*
>
> *`shift o` 从上面起一行开始输入。*

编辑结束后，按`esc`返回命令模式。

在命令模式，`:`进入末行模式，`q 回车`退出vim，`w 回车`保存，`wq 回车` 保存退出，`wq 回车` 可以用`x 回车` 代替。`:q!`表示强制退出，比如在编辑了其他用户的文件时，不能保存，但是想直接退出vim。

> *下`j`，上`k`，左`h`，右`l`的由来?*

> *`j`是右手的默认键位，而且右手是大部分人的惯用手，在阅读时，最多的操作是向下，所以`j`是向下。食指为`j`，中指自然放在`k`上，于是`j`、`k`为上、下。`jk`左边为`h`，因此`h`为向左；右边是`l`，因此`l`为向右。虽然小键盘方向键也可以用来控制方向，但是不建议使用，因为会导致效率的瓶颈。*

3.2.2 其他有用的命令，选择使用，可以大大提高操作效率。

> *在学习命令时，推荐大小写同时记。*

(1) 快速定位光标。

操作符	说明
:行号	光标定位到某一行
0	光标定位到行首
$	光标定位到行末
gg	光标移动到第一行
shift g	光标移动到最后一行
w(word)	下一个词的开头
shift w	下一个大词的开头
e	（下一个）词的末尾
shift e	（下一个）大词的末尾
b	（上一个）词的开头
shift b	（上一个）大词的开头
{	光标移动到段落首字符
}	光标移动到段落尾字符

> *大词：不包含空格的一个连续字符串。*
>
> *比如一个字符串`abc-123`，可以分为三个词，`abc`、`-`、`123`，需要三次`w`才能走完，而只需要一个`shift w`可以走完。*

(2) 光标不动（不换行），整体上下移动页面。（命令模式）

操作符	说明
ctrl e	向下一行
ctrl y	向上一行
ctrl d （down）	向下半页
ctrl u （up）	向上半页

(3) 查找。

操作符	说明
/查找内容（常用）	回车后，查找指定内容，n 下一个，N 上一个
:s/被替换内容/替换内容/	替换当前行的第一个目标
:s/被替换内容/替换内容/g	替换当前行的全部目标
:%s/被替换内容/替换内容/g	替换整个文档的全部目标
:%s/被替换内容/替换内容/gc	替换整个文档的全部目标，替换每个内容时都询问

(4) 复制（命令模式）。

操作符	说明
yy	复制当前行
p	粘贴在光标后
shift p	粘贴在光标前
v	进入选中模式，一个字一个字用来选中
shift v	进入选中模式，一行一行选中
选中模式时 按y	复制所选内容

> *`p(P)` 命令，如果剪切板中是一个字（词），那么就会在光标后（前）粘贴；如果剪切板中是一行，那么就会在光标所在行的下（上）一行粘贴。*

(5) 删除（命令模式）。

操作符	说明
x	剪切一个字符（放入剪切板，可以用p粘贴）
dd	剪切一行（可以用p粘贴）
r	当前字符变成下一个输入的字符

(6)撤销（取消撤销）。

操作符	说明
u (undo)	撤销
ctrl r (redo)	取消撤销

3.2.3 大大提升编辑效率的命令

（1）批处理

几乎所有的vim命令前面可以添加数字，比如`3dd`，就是重复`dd`三次，删除三行；`3j`，向下三行。

（2）编辑命令+位移命令

编辑命令有`d(delete)`、`x(剪切)`、`c(change)`、`y(复制)`。位移命令比如`j(向下)`、`w(下一个词)`、`}(段末)`。

（3）自动补全 `ctrl p`。

（4）分屏。

`sp` 上下分屏。
`vsp` 左右分屏。
`ctrl w` + 上下左右方向键选择操作的分屏。

e.g.
`dw` 删掉一个词。
`y$` 复制当前光标位置到行末的字符串。
`c}`删除当前字符到行末，并进入编辑模式。

`di(`，意思是delete in ()，即删除光标所在的`()`中的内容。
`ci[`，意思是change in []，即修改光标所在的`[]`中的内容。

3.2.4 vim定制化 csdn

vim可以在末行模式进行一些配置，比如`set nu`可以显示行号，`set tabstop`可以设置缩进长度，`syntax on`开启语法高亮等等。

在末行模式下的设置在重新打开vim时不会生效，持久化配置可以在`~/.vimrc`中设置。

4 实验步骤

> *实验答案在`answer template.md`文件中，填写后生成PDF上传spoc即可。*

1. 使用`apt-get`命令安装`vim`（不需要作答）。
2. 打开终端（Terminal），以下所有实验请在终端中进行（不需要作答）。
3. 查看当前登录的用户和所在的目录，分别列出所使用的命令。
4. 创建一个空目录`test`，进入该目录，列出所使用的命令。
5. 在`test`目录构建出以下结构的文件目录，并列出所使用的命令。

> *以下目录树由`tree`命令生成。*

```
.
├── a
│   └── b
│       ├── c.txt
│       └── d.c
├── a2
└── a.rs
```

6. 将`test`目录变成以下结构，并列出所使用的命令。**能否通过两条命令实现？** Hint: cp + 通配符

```
.
├── a
```

```
│   ├── b
│   │   ├── c.txt
│   │   └── d.c
│   └── b2
│       ├── c.txt
│       └── d.c
├── a2
│   ├── b
│   │   ├── c.txt
│   │   └── d.c
│   └── b2
│       ├── c.txt
│       └── d.c
└── a.rs
```

7. 使用`wget https://gitee.com/qq994876761/22_Fall_sp_labs/raw/main/lab01/practice.zip`获取`practice.zip`文件，查看解压后的`practice.c`的最后一行，写出`secret`的值。
8. 使用vim打开`practice.c`，依次完成以下操作，写出每次使用的命令。

   要求：

   - 不要进入编辑模式。
   - 用尽量少的命令来完成。（查看3.2.3(1) 批处理操作一节）

   比如`删除第13行的代码并保存修改`，使用的命令可以是

   ```
   jjjjjjjjjjjjdd:w
   ```

   即向下移动12次，删除本行，保存。
   也可以是

   ```
   :13
   dd:w
   ```

   即光标定位到13行，删除本行，保存。

> *`shift w`这种操作，可以打成`W`。比如光标向后移动三个大词，操作命令可以是`WWW`或者`3W`。*

(1) 删除第13行的代码并保存修改。

(2) 删除最后一行的`secret`的注释并保存。

(3) 删除第1000行到第1020行（共21行）并保存。

(4) 将第50行复制到第100行并保存。

(5) 将全文的`static`替换成`stastic`并保存。

(6) 退出vim。

参考资料：

1.https://www.openvim.com/
2.https://missing-semester-cn.github.io/的15小节，B站链接