梁博 shell 脚本训练集训学习小结 V0.1

从打开 linux 虚拟机里的 Terminal 开始，学习 shell 成了这两周主要的工作。在梁博的带领下，100 多位童鞋开始 shell 脚本的学习。两周学习收获颇多，在感谢梁博指导和向各位童鞋学习之余，我打算写一篇文章来总结一下自己在这两周里的收获。由于是总结，所以我并不打算按照我学习的顺序来写这篇文章，而是按照以下模块来构建这篇文章。

第一部分　Linux 基础应用篇
第二部分　shell 语法基础篇
第三部分　shell 技巧篇
第四部分　两周所用到的 API 总结篇

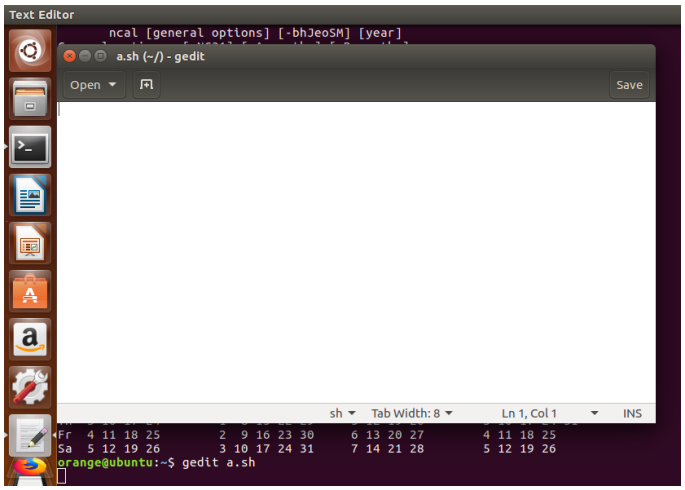
第一部分 Linux 基础应用篇

Linux 操作系统是有很多种的，这里我选用的操作系统是 Ubuntu。首先需要了解 Linux 操作系统既可以在可视化界面下操作，也可以在它的控制台 Terminal 中进行所有的操作。一般使用 Linux 主要看中的是在控制台 Terminal 中操作的方便性。比如说当你想要安装一个软件时，你可以使用 *sudo apt install+你所要安装软件的名称*就可以轻松地从网上下载安装所需要的软件。方便的前提是熟悉，在控制台环境下你需要熟练应用一些常用的命令来帮助你表达你的操作。

①*mkdir* 这个命令主要是建立新文件夹的；
②*cd*　　这个命令可以用来打开文件夹；
③*ls*　　这个命令可以来查看当前目录里的文件；
④*gzip, zcat* 这是压缩文件和读取压缩文件的命令；
⑤*rm*　　这个是删除文件的命令
⑥*chmod* 这个命令主要是来改变文件的权限的，当一个程序要运行时，首先要用 *chmod +x *.** 改变文件权限，然后才能执行文件。
⑦*./*　　这个命令是用来执行程序的

下面介绍两个比较常用的编写程序的软件 **gedit** 和 **Vim**

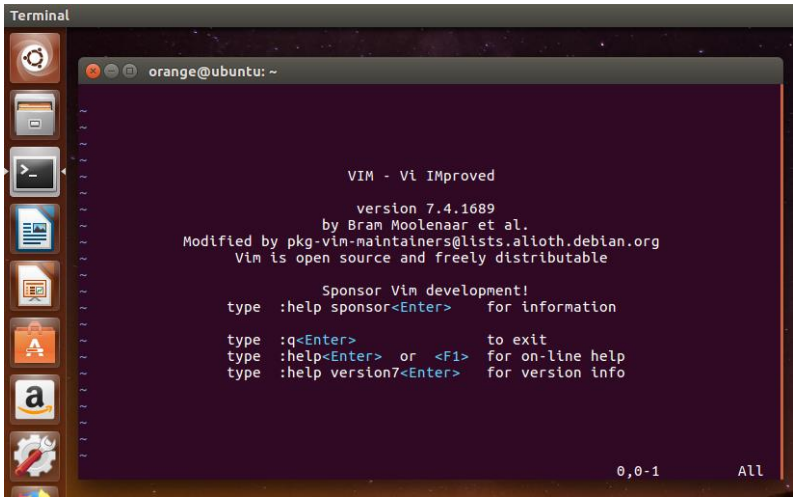
gedit 比较简单，通过 *gedit+文件名称*就可以生成新文件进行编写，像这样：



这个空白的界面就是编写程序的窗口，你可以在这个界面上编写不同的程序。文件名的后缀决定了这个程序到底需要用什么编译器进行编译。这是最基本的程序编写界面，这样的好处是很明显的，但是当要编写比较复杂的程序时，这样的软件就会存在这样那样的问题和

不便。

Vim 是一款十分经典的程序。它的操作也比较简单，与 gedit 不同的是，它并不会单独弹出一个新窗口来进行程序的编写，它会覆盖控制台，像这样：



这样的好处是在一些服务器平台上，它们没有界面化的窗口，这时候就只能使用 Vim 等软件来编写程序和脚本。基于上述的情况，学习 Vim 还是十分有必要的。一些常用的快捷键：

①*shift+zz* 这个命令表示保存退出
②*gg* 直接定位到首行
③*G* 直接定位到尾行
④*u* 撤销
⑤*Esc* 切换到正常模式
⑥*I* 在光标处插入
⑦*: q* 退出
⑧*:wq* 保存并退出

熟练掌握这些快捷键，对于你使用 Vim 来进行编程是十分方便的。

第二部分　shell 语法基础篇

对于 shell 我是很陌生的，从定义的角度来讲 shell 和 shell 脚本是不一样的。这里我们主要讲的是 shell 脚本，通过编写脚本完成对文件的设置和管理。shell 既是一种命令语言，又是一种程序设计语言。作为命令语言，它主要来解释和执行用户输入的命令；作为程序设计语言，它定义各种变量和参数，并提供控制结构比如循环和分支。

shell 的种类是很多的，一般采用的 Bash 这种。Bash 也是大多数 Linux 系统默认的 shell。在编写 shell 脚本时，第一行往往要写上*#!/bin/bash* 来确定脚本是采用何种 shell 来执行的。

下面来熟悉一下 shell 的几个常用的命令，最简单的方法其是在控制台输入命令名+‘--help’（比如:*sort --help*）就会显示该命令的全部用法。这里我们只是简单介绍一下方便熟悉和回顾。

①*awk*

awk 可以逐行扫描文件，需要匹配特定模式的行，并在这些行上进行你想要的操作。其基本结构可以写为： *awk pattern {action}*

首先从原理来分析一下这条命令：*awk* 一次从文件中读取一条记录，并将记录存储在字

段变量$0 中，记录被分割为字段并存储在$1，$2，…，$NF 中（默认空格为分隔符）。

例如： *awk '{print $1,$2}' test.txt* （这就表示从 test.txt 文件中读取前两段字符）

再比如： *awk '{print NF}' test.txt* （输出每一行都可以分隔为几段）

对于分隔符不是空格的情况，主要通过-F 或 FS 变量来改变分隔符。

例如：*awk -F: '{print $1}' test.txt*

或者 *awk 'BEGIN {FS=":"} {print $1}' test.txt*

（分隔符改为:，输出第 1 段的字符）

awk 有一些内置的变量，比如 NR（输出文件当前记录的编号），OFS（设置输出字段的分隔符，默认为空格）等等

例如：*awk '{print NR}' test.txt*

awk 'BEGIN{OFS="-"}{print $1,$2,$3}' test.txt

除了一些内置函数外，*awk* 还可以和一些函数进行搭配运算。例如 *gsub(x,y,z)*，其含义是在字符串 z 中使用字串 y 替换与正则表达式 x 相匹配的所有字串，z 默认为$0。

例如：*awk -F: 'gsub("111","aaa"){print $0}' test.txt*

更加复杂的用法是加入 *if*，*while*，*for* 循环等，这一点将在第三章中有所涉及。

②*uniq*

uniq 的作用主要是去重，但是需要说明的是为了使 *uniq* 起作用，所有的重复行必须是相邻的。

例如：*uniq filename*

uniq -i filename（忽略大小写字符的不同）

uniq -c filename（进行计数）

uniq -u filename （只显示唯一的行）

③*split*

split 主要的作用是分割文件，一般来说分为两种：一种是按行分割，具体写法是 *split -l 10 filename*；另外一种是按字节分割，具体写法是 *split -C/-b 10 filename* 二者在使用上有一点小区别，前者*-c* 会尽量保证每行的完整性，而后者*-b* 则会完全按照字节大小进行文件分割。

④*sort*

sort 是比较重要的一条命令，它的作用是串联排序所有指定文件并将结果写到标准输出。例如：*sort -f filename*（忽略字母的大小写），*sort -n filename*（根据字符串数值大小进行排序），*sort -r filename*（逆序输出排序结果）

sort 也可以完成对文件的去重过程，写法为：*sort -u filename*。但要注意的是它与 *uniq* 去重还是有一些不同之处的。后者去重需要重复行是相邻的，而前者则并不需要。

对于多列的文件，*sort* 还可以指定列数进行排序。写法是 *sort -k 列数 filename*。这样就可以规定排序的方式是按照哪一列来排列的。

例如：*sort -k2,2rn filename*（这就表示以第 2 列为准进行排序，排序按照降序进行）

⑤*join*

join 命令主要完成两个文件的合并。最简单的例子：

join file1.txt file2.txt（将两个文件进行匹配）

对于含有多列的两个文件，可以通过-o 将需要的匹配列进行输出合并

例如：*join -o 1.2 -o 2.2 file1.txt file2.txt*

⑥*grep*

gerp 也是一个比较有用的命令，它可以对文件内容进行搜索并输出搜索到的结果。常用的主要有以下几种：

grep -c 匹配项 filename（只输出匹配行的计数）

grep -i 匹配项 filename（不区分大小写）

grep -w 匹配项 filename（匹配整个字符串，而不是只要字符串包含所匹配项就会输出）

grep -v 匹配项 filename（显示除去已匹配的内容）

⑦*cut*

cut 比较简单，形象的说主要是用来裁剪文件的。通过 *cut* 输出文件中想要得到的部分。例如：

cut -f/-b/-c

cut -f3,5 filename（输出文件的第 3,5 部分，一般来说就是第 3,5 列）

⑧*cat*

cat 主要作用是输出文件内容。

⑨*echo*

*echo*类似于 c 语言的 printf，作用是打印输出。一般第一个 shell 脚本都是输出 hello world 的，可以用 echo 来写如下：

echo "hello world!"

输出的结果就是 hello world!

⑩*sed*

sed 可以完成打印，替换，删除等多个功能，是一个比较强大的命令。

例如： *sed 's/abc/def/g' filename*（意思是替换文中 abc 为 def）

更多具体的功能可以使用 *sed --help* 来查看

以上就是 shell 脚本的一些常用的命令。虽然不是很全，但是基本上可以满足平时的需要。学习的时候最好一个一个进行练习，这样就能清楚的了解各个命令的用法，从而加深印象，以便灵活应用。

第三部分　shell 技巧篇

这部分想总结一下在这两周学习过程中用到的脚本片段。主要的作用是方便以后的应用。

①*awk 'END{print NR}' filename*（统计文件里内容的行数）

②*c=`expr $a / $b`*（给 a,b 赋值，然后进行除法运算）

③*wget -O filename "http://api.pullword.com/post.php","source=$var &param1=0&param2=1"*（调用 API 并将结果返回到 filename 中）

④
```
if [ "$a" -ge "$b" ]
then
  cishu=$b
else
  cishu=$a
```

fi（if 语句的写法）

⑤*for ((i=1;i<=k-1;i=i+2));*

do

echo “$i”

done（for 循环语句的写法）

⑥*grep -E "[A-Za-z0-9._-]+@[A-Za-z0-9_-]+[\.][A-Za-z.]*" filename*（查找并提取邮箱地址，这里用到了正则表达式来对文件进行筛选。RegexBuddy 是个很好的正则表达式生成学习的软件）

⑦*join day1.txt day2.txt >result*

join -v 1 day1.txt day2.txt > tmp1.txt
awk '$2=$2 " 0"' tmp1.txt >> result

join -v 2 day1.txt day2.txt > tmp2.txt
awk '$1=$1 " 0"' tmp2.txt >> result

sort result -k1,1n|grep -vw 0 （以第 1 列为标准，将两个文件中的两行内容进行匹配，相同的进行合并，不同的仍然要显示出来）

⑧*sed -n '1 p'*（输出第 1 行的内容，配合 awk 就可以输出指定行，指定列的内容）

第四部分　两周所用到的 API 总结篇

1 分词（来自梁博所维护的分词平台）

wget "http://api.pullword.com/post.php","source=清华大学是个好学校&param1=0&param2=1"

输出的结果是：

清华:0.604942 清华大学:1 华大:0.0540265 大学:0.951753 是个:0.658764 个好:0.212575 好学:0.412944 学校:0.934511

可以看到通过分词 API 可以从一段话里找到关键词。

2 地理位置

①BaiduAPI（百度 API 只能输出国内的信息）

wget“http://api.map.baidu.com/geocoder?callback=renderReverse&location=30.1,102.3&output=json&pois=1”

输出结果为：

```
{
    "status":"OK",
    "result":{
        "location":{
            "lng":102.3,
            "lat":30.1
        },
        "formatted_address":"四川省雅安市天全县",
        "business":"",
        "addressComponent":{
            "city":"雅安市",
            "direction":"",
            "distance":"",
            "district":"天全县",
            "province":"四川省",
            "street":"",
            "street_number":""
        },
        "cityCode":76
    }
}
```


② openstreetmapAPI（这个 API 输出内容不多，但是可以输出国外信息）

wget" https://www.openstreetmap.org/geocoder/search_geonames_reverse?lat=30.1&lon=102.3"

输出结果：

- 四川省，中国

③GoogleAPI（最后的 KEY 是自己申请的密钥）

wget" https://maps.googleapis.com/maps/api/geocode/json?latlng=40.714224,-73.961452&key="*

输出结果：


```
,
"formatted_address" : "277 Bedford Ave, Brooklyn, NY 11211美国",
"geometry" : {
   "location" : {
      "lat" : 40.7142205,
      "lng" : -73.9612903
   },
   "location_type" : "ROOFTOP",
   "viewport" : {
      "northeast" : {
         "lat" : 40.71556948029149,
         "lng" : -73.95994131970849
      },
      "southwest" : {
         "lat" : 40.7128715197085,
         "lng" : -73.9626392802915
      }
   }
},
"place_id" : "ChIJd8BlQ2BZwokRAFUEcm_qrcA",
"types" : [ "street_address" ]
```