输出、查看命令

- 命令**echo**用以显示输入的内容
- 命令**cat**用以显示文件内容
- 命令**head**用以显示文件的头几行(默认10行)

```
-n     指定显示的行数1
```

- 命令**tail**用以显示文件的末尾几行(默认10行)

```
-n     指定显示的行数
-f     追踪显示文件更新(一般用于查看日志,命令不会退出,而是持续显示新加入的内容)12
```

归档、压缩

- 命令**zip**用以压缩文件

```
zip linuxcast.zip myfile1
```

- 命令**unzip**用以解压缩zp文件

```
unzip linuxcast.zip1
```

- 命令**gzip**用以压缩文件

```
gzip linuxcast.net1
```

- 命令**tar**用以归档(archive)文件

```
tar -cvf out.tar linuxcast
tar -xvf linuxcast.tar
tar -cvzf backup.tar.gz/etc
   -z参数将归档后的归档文件进行gzip压缩以减少大小
```

AWK

awk是一个非常好用的数据处理工具。相较于sed常常一整行处理，awk则比较倾向于一行当中分成数个“字段”处理，awk处理方式：依次对每一行进行处理，然后输出

```
$ awk '条件类型1{动作1} 条件类型2{动作2} ...' filename
```

查看最近5条登录用户和ip地址

```
$ last -n 5 | awk '{print $1"\t"$3}'
```

变量名称	代表意义
NF	每一行($0)拥有字段总数
NR	目前awk处理的第几行
FS	目前分隔符，默认是空白
$0	整行
$d	第d行
-F	指定分隔符
-f	指定脚本文件
-v	定义变量 var=value

```
awk (-F | -f | -v) `BEGIN{} //{command1; command2} END{}' file
```

```
while语句
awk -F: 'BEGIN{i=1} {while(i<NF) print NF,$i,i++}' /etc/passwd
数组
netstat -anp | awk 'NR!=1{a($6)++} END{for (i in a) print i,"\t",a(i)}'
```

GREP

grep（global search regular expression(RE) and print out the line，全面搜索正则表达式并把行打印出来）是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打印出来。

```
-i 忽略字符大小写的差别。
-n 在显示符合范本样式的那一列之前，标示出该列的编号。
-R/-r 此参数的效果和指定"-d recurse"参数相同。在目录下递归查找
-v 反转查找。显示不匹配的文本行
-y 此参数效果跟"-i"相同。
-e / egrep  使用正则表达式
-o 只输出匹配到的部分
```

在文件中搜索一个单词，命令会返回一个包含**“match_pattern”**的文本行：

```
grep match_pattern file_name
grep "match_pattern" file_name
```

在多个文件中查找：

```
grep "match_pattern" file_1 file_2 file_3 ...
```

```
输出除之外的所有行 -v 选项
grep -v "match_pattern" file_name
使用正则表达式 -E 选项：
grep -E "(1-9)+"
或
egrep "(1-9)+"
```

只输出文件中匹配到的部分 **-o** 选项：

```
echo this is a test line. | grep -o -E "(a-z)+\."
line.

echo this is a test line. | egrep -o "(a-z)+\."
line.
```

统计文件或者文本中包含匹配字符串的行数 **-c** 选项：

```
grep -c "text" file_name
```

输出包含匹配字符串的行数 **-n** 选项：

```
grep "text" -n file_name
或
cat file_name | grep "text" -n

#多个文件
grep "text" -n file_1 file_2
```

搜索多个文件并查找匹配文本在哪些文件中：

```
grep -l "text" file1 file2 file3...
```

在多级目录中对文本进行递归搜索：

```
grep "text" . -r -n
# .表示当前目录。
```

忽略匹配样式中的字符大小写：

```
echo "hello world" | grep -i "HELLO"
hello
```

查看内存CPUIO

`vmstat` 查看cpu的us、sy、id、wa 使用情况 也可查询内存使用情况 `vmstat 2 5`

`top` 查询cpu的详细使用情况和占cpu较高的进程

`iostat -x 2 5` 查看各磁盘的%util情况，越高说明磁盘对应的io越高

`iotop` io版的top

`pidstat` 统计进程(pid)的stat,进程的stat自然包括进程的IO状况