重定向符号

符号	解释	案例
>	输出重定向覆盖符号	cat 1.txt > 2.txt
>>	输出重定向追加符号	cat 1.txt >> 2.txt
<	标准输入重定向	xage -n 3 < 2.txt
<<	标准输入追加重定向	cat >> gushi.txt <<EOF

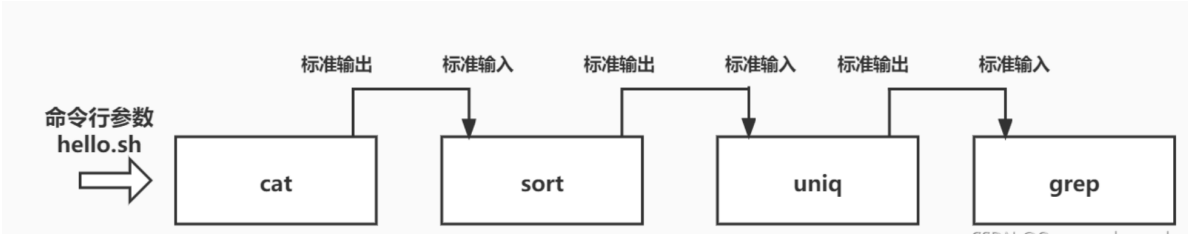
cat命令

- -b 对非空行显示行号
- -n 对所有行显示行号
- -s 多个空行减为一个
- cat 1.txt 2.txt > 3.txt 将多个文件内容合并写入新的文件中
- cat >> 2.txt << EOF 将控制台内容写入2.txt
- cat清空文件
  - 直接清空文件，留下空行 echo> gushi.txt
  - 直接清空文件内容，不留空行 > gushi.txt
  - 利用cat读取一个黑洞文件，清空 cat /dev/null > gushi.txt

tac命令

把文件内容倒过来输出

管道符



对字符串进行二次过滤

```
#查看gushi.txt文本内容，且对文本进行二次过滤，找出有关content的内容
cat gushi.txt | grep "content"
```

more less命令

```
more gushi.txt
#分屏显示当前文件内容
#q 推出
#= 显示行号
#enter 下一行
#space 向下滚动一个屏大小
```

head tail 命令

```
#head tail 默认显示10行
head -n 3 gushi.txt
head -c 5 gushi.txt #指定输出这个文本的5个字符
tail -n 5 gushi.txt
tail -f gushi.txt #实时刷新文件内容变化
```

cut命令：显示指定列内容

```
cut -c 4 gushi.txt #显示第四列的内容
cut -c 4-7 gushi.txt #显示第4-7列的内容
cut -c 4,6 gushi.txt #显示4，6两列的内容
cut -c  -7 gushi.txt #显示开头到7列的内容
cut -c 8- gushi.txt #显示8列到结尾的内容
cut -d ":" -f 1-7 gushi.txt #按照冒号切割，并找到第1-7个的内容
```

sort命令：内容排序

```
sort -n gushi.txt #以第一个数据进行排序
sort -nr gushi.txt #以第一个数据降序排序
sort -u gushi.txt #对数据排序，去重
sort -t "." -k 4 gushi.txt #对数据按.分割，然后按第四个进行排序
```

uniq命令：重复内容去重

```
uniq gushi.txt #对文件内容去重

#结合sort使用
sort -t "." -k 4 ip.txt | uniq #对内容去重排序去重
sort -t "." -k 4 ip.txt | uniq -c #统计重复次数
sort -t "." -k 4 ip.txt | uniq -d #找出重复行，并显示次数
sort -t "." -k 4 ip.txt | uniq -u #找出不重复的行
```

wc命令：文件行数

```
wc -l gushi.txt #统计文件行数
echo "alax wudongqiankun anlx" | wc -w #统计单词数量
echo "alex" |wc -m #统计字符数，字符后面有$符
cat gushi.txt | wc -L #统计文件的字符数
```

tr命令：替换字符

```
echo "my name is alex" | tr '[a-z]' '[A-Z]' #替换字符命令
echo "my name is alex111" | tr -d 'a-z' #删除字符命令
tr 'a' 'A' < alex.txt #把文件的内容进行替换
```

find命令：查找文件

```
find / -name "*.txt" #指定根目录找后缀为txt的文件
find / -maxdepth 1 -name "*.txt" #最大深度为1的txt文件
find / -type d -name "*" #找目录
find / -type f -name "*" #找文件
find . -atime -2 #两天以内访问的文件
find . -type f -size +200M
du -h xxx.txt #查看文件大小
find . -path "./test_find" -prune -name "*.txt" -print #忽略文件夹搜索
find . -type f -name "*.txt" -ok rm {} \; #ok是执行命令，删除找到的所有文件
```

xargs命令

- 管道命令

```
xargs < jiang.txt   #多行变单行
xargs -n 2 < jiang.txt #一行变2个数
echo "alex,mjj,cunzhang,jiang" | xargs -d "," #-d指定分隔符
find . -name '*.txt' | xargs -i mv {} dir/ #将find查询出来的文件移动到dir目录下
find . -name '*.txt' | xargs -I alltxt mv alltxt ./#找出txt文件，将所有的txt改名
alltxt，并把所有alltxt文件放入 ./目录
```

linux练习题

```
#查看文件显示行号
cat -n 1.txt
#清空文件内容
> 1.txt
#显示文件前30行
head -n 30 1.txt
#显示文件后50行
tail -n 50 1.txt
#实时刷新文件内容
tail -f 1.txt #一般看log日志用这个
#读取文件内容并倒叙排序
cat 1.txt | sort -nr
#读取文件内容排序后去重
cat 1.txt | sort -n| uniq
#读取文件内容排序，统计重复行次数
cat 1.txt |sort -n |uniq -c
#统计文件有多上行
wc -l 1.txt
#查看文件详细信息
stat 1.txt
#找出“.py”结尾的文件
find / -type f -name "*.py"
#找出两天内访问的数据
find / -atime 2 -type f
#找出大于50m的文件
find / -size +50M -type f
#找出/tmp/目录下所有的txt文件，然后删除
find /tmp/ -type d -name '*.txt' -ok rm {}
#把/data/html/文件夹打包成data_html.tgz
```

```
#-c 打包  -x解包  -v显示过程 -f指定文件 -z压缩
tar -czvf   /data/html data_html.tgz
#如何解压all.gz文件
tar -zxvf all.gz ./
#显示系统时间
date
#彻底粉碎文件
shred -u 1.txt
```

linux通配符

```
#找出目录最大深度为3，以l为头，小写字母为结尾，中间出现任意一个字符
find -maxdepth 3 -type f  -name "l?[a-z]"
find -maxdepth 3 -type f  -name "l?[[:lower]]"
#找出/tmp以数字开头，以非数字结尾的文件
find /tmp -type f -name "[0-9][!0-9]"

#非字母开头，后面跟着一个字母及其他任意长度字符的文件
find /tmp -type f -name "[^a-zA-Z][a-zA-Z]*"

#移动/tmp下，所有以非字母开头的文件，复制到/tmp/allNum/目录下
find /tmp -name '[^a-zA-Z]*' -ok cp {} /tmp/allNum

#复制/tmp目录下所有的.txt文件，以y，t开头的文件，放入/data目录
cp /tmp/[yt]*.txt /data
```

grep命令 文本搜索，对文本内容过滤，筛选

```
# 统计pwd.txt中有多少行root有关的,-c显示行号
grep -i "root" ./pwd.txt -c
# 找出所有的空行
grep '^$' 1.txt -n
grep '^$' 1.txt -n -v #找出所有非空行
grep '^#' 1.txt -v | grep '^$'-v #找出不带#开头的行以及非空行

#找出所有M开头的行
grep -i -n '^m' 1.txt
#找出以.结尾的行
grep -n ".$" 1.txt
#找出以/bin/bash结尾的行
grep -n "/bin/bash$" passwd.txt
#匹配所有内容
grep ".*" passwd.txt
#贪婪匹配
grep ".*e" passwd.txt
```

sed命令

```
#输出带有linux字符的行
sed "/linux/p" 1.txt -n #-n代表只显示匹配的行
#删除带有game字符的行
sed "/game/d" 1.txt -i #-i代表把源文件匹配到到的内容删除
#将文件中的my替换为i "s/原内容/要替换的内容/g全局"
```

```
sed "s/my/I/g" 1.txt
#将my替换为i，将100替换为1
sed -e "s/my/I/g" -e "s/100/1/g" 1.txt
#再文件第二行后添加i am good
sed "2a i am good" 1.txt
#再文件第四行前添加i am good
sed "4i i am good" 1.txt
#再每一行后都加分隔符
sed "a -------" 1.txt

#找到ipconfig 中eth0 inet地址
#第一种方法
ifconfig eth0 | sed "2p" -n #找出第二行数据
ifconfig eth0 | sed "2p" -n | sed "s/^.*inet//"#将inet后续的内容提取出来
ifconfig eth0 | sed "2p" -n | sed "s/^.*inet//" | sed "s/net.*$//"
#第二种方法
ifconfig eth0 | sed -e "2s/^.*inet//" -e "2s/^.*inet//" -n
```

awk命令

内置变量	解释
$n	指定分隔符后，当前记录的第N个字段
$0	完整的输入记录
FS	字段分隔符，默认是空格
NF	分割后，当前行一共有多少个字段
NR	当前记录数，行数

```
#输出第一列，第四列，第五列
awk '{print $1,$4,$5}' 1.txt #,代表默认分隔符
#输出第二列的数据
awk '{print $2}' 1.txt
#输出第五行六的数据
awk 'NR==5,NR==6{print $0}' 1.txt
#输出第30-40列，并加行号
awk 'NR==30,NR==40{print NR,$0}' 1.txt
#输出导数第二列
awk '{print $(NF-1)}' 1.txt
```