

文本处理命令



Linux命令之CT机

grep

Linux命令之切片机

cut

Linux命令之粘合剂

paste

井井有条的sort命令

独一无二的uniq命令

擅长计数的wc命令

多才多艺的sed命令

Linux命令之翻译家

tr

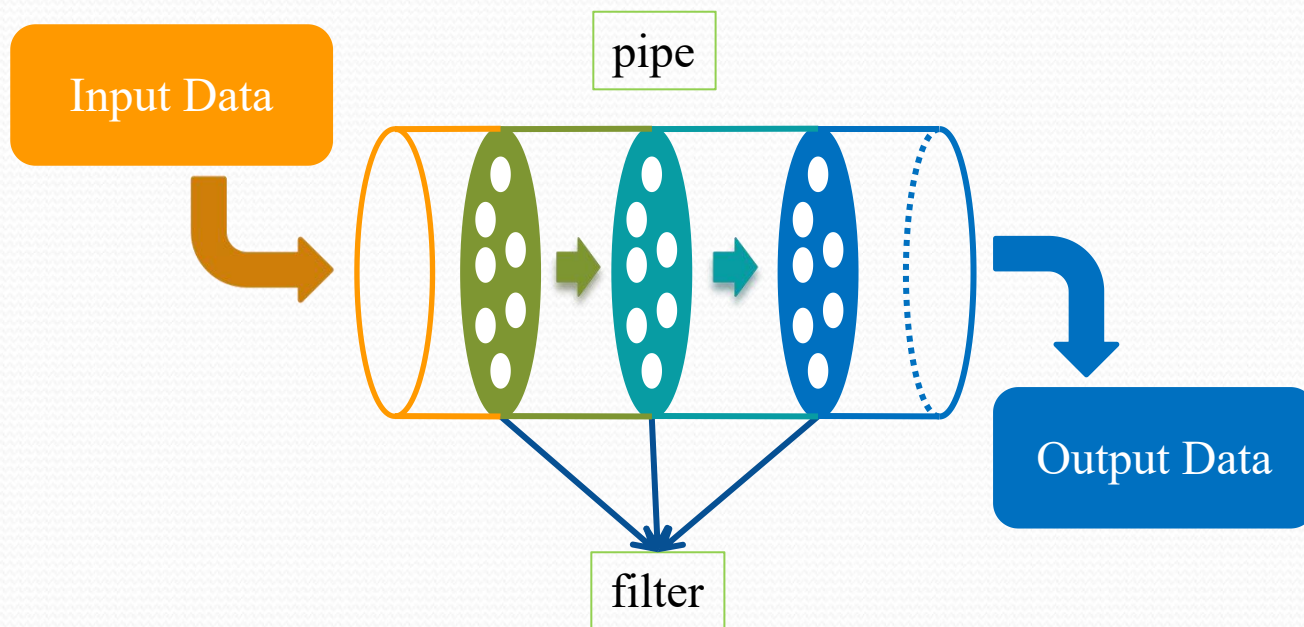
Linux命令之剪刀手

awk

瘦死的骆驼比马大

perl

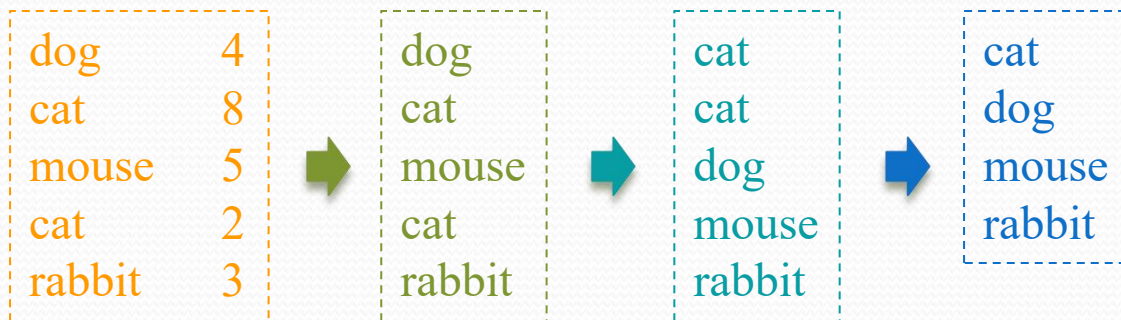
Linux下的文本处理原理



命令管道:

```
cat input.txt | cut -f1 | sort | uniq > output.txt
```

数据变化:



Linux命令之CT机--grep命令



grep命令档案



命令全名： globally search a regular expression and print

命令用途： 查找并显示包含指定模式的行

命令格式： grep [option] pattern file

常用选项：

- c 统计结果行数
- E 使用扩展的正则表达式
- i 忽略大小写
- v 反向查找
- A n 显示找到的行及其后面n行内容

grep命令用法

∞ 查找文件中包含一个模式的行

```
$grep 'NP_173616.2' at_pep.fa
```

∞ 查找文件中包含一个模式的行及其后面几行内容

```
$grep -A 3 'NP_173616.2' at_pep.fa
```

∞ 查找当前目录中名字包含特定字符串的文件/目录

```
$ls | grep at
```

∞ 去掉文件中包含某个字符的行

```
$grep -v '#' ~/bin/wgetr
```

Linux命令之切片机--cut命令



cut命令档案



命令全名： cut

命令用途： 取出文本中指定的列

命令格式： cut [option] file

常用选项：

- d 指定列分隔符，默认是制表符
- f 指定取出的字段
- b 指定取出的字节
- c 指定取出的字符

cut命令的用法

取出一个文件中的一列，分隔符是制表符

```
$cut -f1 at_gene_partial.gff
```

取出一个文件中的中的两列，分隔符不是制表符

```
$cut -d: -f1,6 /etc/passwd
```

取出压缩文件的几列

```
$zcat at_gene_partial.gff.gz | cut -f1,3-5
```

cut命令选项示例

```
$ cat char.txt
```

China

中国

```
$ cat char.txt | cut -b1
```

C

中

```
$ cat char.txt | cut -b1-2
```

Ch

中

```
$ cat char.txt | cut -b1-3
```

Chi

中

```
$ cat char.txt | cut -c1
```

C

中

file命令查看一个文件的字符编码方式

```
$ file char_chs.txt
```

char.txt: UTF-8 Unicode text

字符编码

“a”的UTF-8编码:

0	1	1	0	0	0	0	1
---	---	---	---	---	---	---	---

“中”的UTF-8编码:

1	1	1	0	0	1	0	0
1	0	1	1	1	0	0	0
1	0	1	0	1	1	0	1

Linux命令之粘合剂--paste命令



paste命令档案



命令全名: paste

命令用途: 按列合并文件

命令格式: paste [option] file1 file2 ...

常用选项:

- d 指定合并后的列分隔符，默认的列分隔符是制表符

paste命令的用法

∞ 按列合并两个文件

```
$paste animal number
```

∞ 按列合并两个文件，分隔符是冒号

```
$paste -d: animal number
```

井井有条的sort命令



sort命令档案



命令全名: sort

命令用途: 按行排序文本

命令格式: sort [option] file

常用选项:

- r 反向排序
- n 按数字大小排序
- g 按科学计数法数字大小排序
- k 指定参与排序的字段

sort命令用法 (1)

∞ 按ASCII码值升序排序

```
$sort num
```

∞ 按数字大小升序排序

```
$sort -n num
```

∞ 按数字大小降序排序

```
$sort -nr num
```

∞ 按数字大小排序，包括科学计数法的值

```
$sort -g num
```

sort命令用法 (2)

指定字段排序

`$sort -k3 price`

`-kn1[.m1][,n2[.m2]]`

开始字段 开始字符 结束字段 结束字符

按指定字段的指定字符排序

`$sort -k1.6,1n price`

按多个字段排序

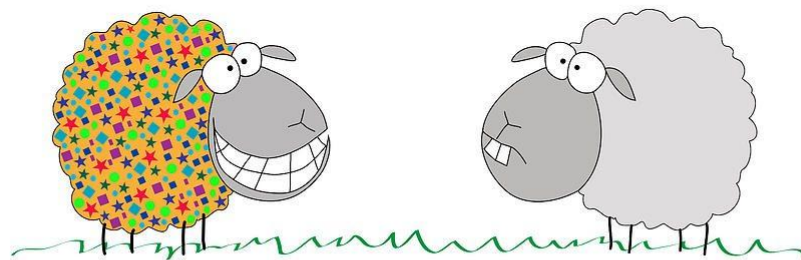
`$sort -k1.6,1n -k3 price`

sort的-k选项用法:

-k1	:	<u>fruit2</u>	<u>6.5</u>	<u>China</u>
-k2	:	<u>fruit1</u>	<u>5.4</u>	<u>US</u>
-k2,2	:	<u>fruit5</u>	<u>7.7</u>	US
-k2.1,2.2	:	<u>fruit6</u>	<u>3.2</u>	UK
-k2.2,2.4	:	<u>fruit1</u>	<u>4.7</u>	China
-k1.6,2	:	<u>fruit10</u>	<u>6.2</u>	China

独一无二的uniq命令

BE UNIQUE



uniq命令档案



命令全名: **unique**

命令用途: 检查和删除重复的行（经常用在sort后面）

命令格式: **uniq** [option] file

常用选项:

- c 统计行重复的次数

uniq命令用法

∞ 去掉文件中相邻的重复的行

```
$uniq animal
```

∞ 去掉文件中所有的重复的行

```
$sort animal | uniq
```

∞ 统计文件中行的重复次数

```
$sort animal | uniq -c
```

擅长计数的wc命令



WC命令档案



命令全名: word count

命令用途: 统计文本的行数、单词数、字符数或字节数

命令格式: wc [option] file

常用选项:

- l 只输出行数
- w 只输出单词数（空格或制表符分隔的字符串）
- m 只输出字符数
- c 只输出字节数

WC命令用法

∞ 统计文件的行数、单词数及字节数

```
$wc animal
```

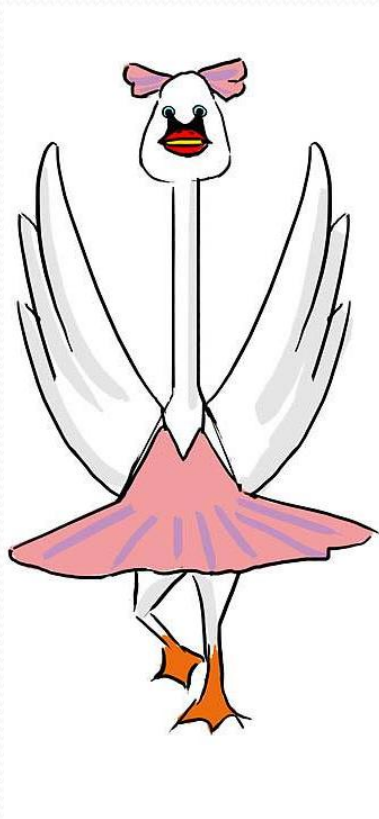
∞ 统计文件的行数

```
$wc -l animal
```

∞ 用在管道中，统计其它命令输出的文本的行数

```
$sort animal | uniq | wc -l
```

多才多艺的sed命令



sed命令档案



命令全名: **s**tream **e**ditor (流编辑器)

命令用途: 过滤或修改文本

命令格式: `sed [option] command file`

`sed [option] -f script_file file`

常用选项:

`-f script_file` 从脚本文件script_file读取要执行的命令

`-n` 仅输出command处理后的结果

sed命令用法

∞ 替换文件中部分文本

```
$sed '1,3s/cat/CAT/g' animal
```

∞ 将包含每个模式的行修改为指定文本

```
$cat animal | sed '/dog/c\old dog'
```

∞ 取出部分文本

```
$cat animal | sed -n '1~2p'
```

sed命令脚本格式

`$sed '1,3s/cat/CAT/g' animal`



sed命令脚本=操作范围+操作符:

`1,3s/cat/CAT/g`

`/dog/c\old dog`

`1~2p`

操作范围

∞ 绝对地址： 2 3,5 %（或1,\$）

∞ 相对地址： 2,+3

∞ 匹配地址： /dog/ /dog/,/pig/

∞ 组合地址： 1,/pig/ /dog/,+1

∞ 步进地址： n~m（n为起始行号，m为步长），如：

1~2（奇数行）

2~2（偶数行）

2~3（2,5,8,...行）

操作符

∞p操作: 2p

#打印第2行

∞=操作: /g/=

#打印匹配到g的行的行号

∞d操作: 3,5d

#删除第3-5行

∞s操作: %s/a/A/g

#替换所有的a为A

∞a操作: 2a\apple

#在第2行后面一行添加apple

∞i操作: 4i\apple

#在第4行前面插入apple

∞c操作: /dog/c\old dog

#将包含dog的行的内容修改为old dog

Linux命令之翻译家--tr命令



tr命令档案



命令全名: **tr**anslate

命令用途: 转换或删除字符

命令格式: **tr** [option] set1[set2]

常用选项:

- d 删除字符集set1指定的字符

tr命令用法

∞ 转换一个字符，如

```
$tr a A <animal
```

∞ 将小写字母转换成大写字母，如

```
$tr [:lower:] [:upper:] <animal 或
```

```
$tr a-z A-Z <animal
```

∞ 删除字符，如

```
$cat animal | tr -d 'dog'
```

字符集语法

普通字符集: ‘abc’ ‘135’ ... (任意字符的集合, 顺序无关)

字符范围: a-z A-Z 0-9 ... (ASCII表顺序)

转义字符: \n \r \t ...

字符类:

[[:alnum:]] 字母和数字

[[:alpha:]] 字母

[[:cntrl:]] 控制 (非打印) 字符

[[:digit:]] 数字

[[:graph:]] 图形字符

[[:lower:]] 小写字母

[[:print:]] 可打印字符

[[:punct:]] 标点符号

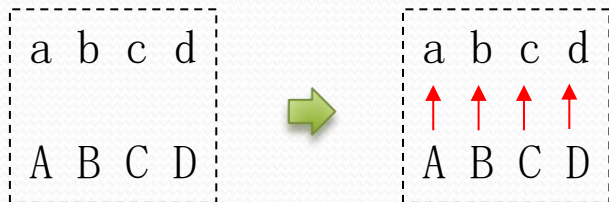
[[:space:]] 空白字符

[[:upper:]] 大写字母

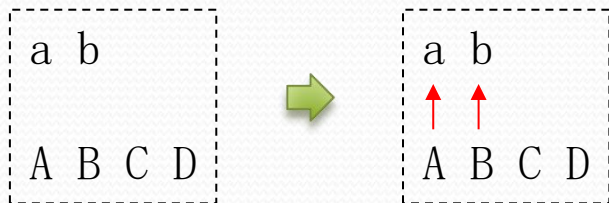
[[:xdigit:]] 十六进制字符

字符集替换规则

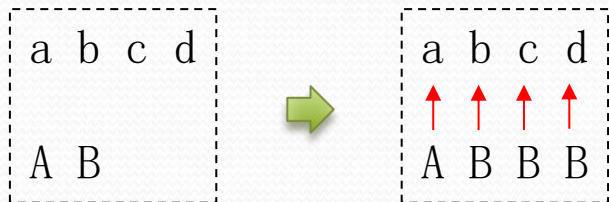
∞set1=set2: 逐一替换



∞set1<set2: set2多余的不用



∞set1>set2: set2最后一个字符重复使用至set1最后一个字符



Linux命令之剪刀手--awk命令



awk命令档案



命令全名: Aho-Weinberger-Kernighan

命令用途: 处理并格式化文本

命令格式:

```
awk [options] 'script' file
```

```
awk [options] -f scriptfile file
```

常用选项:

- F 指定输入文件的字段分隔符，默认是任何空白字符
- f 从脚本文件中读取awk脚本

相关概念

记录	banana	4.6	China		
	apple	7.0	China		记录分隔符（换行符）
	orange	4.9	China		
	litchi	5.9	Thailand		
	字段	字段	字段（域）		
					字段分隔符（空格）

awk脚本基本格式

☞ `'BEGIN{ actions } pattern{ actions } END{ actions }'`

☞ `BEGIN{ actions }`: 在从文件或标准输入读入数据前执行

☞ `pattern{ actions }`: 每从文件或标准输入中读取一行就执行一次

☞ `END{ actions }`: 处理完从文件或标准输入读入的数据后执行

☞ pattern（模式）

☞ 条件表达式，如： `$2>5`（第二个字段的值大于5）

☞ 正则表达式，如： `/apple/`（记录中如果包含apple）

☞ 模式范围，如： `/apple/,/orange/`（apple第一次出现至orange第一次出现之间的所有的行）

☞ 模式匹配表达式，如： `$1~/^a/`（第一个字段是a开头的行）

☞ actions（操作）

☞ 打印，如： `print $1`（打印第1个字段）

☞ 变量赋值，如： `total=0`

☞ 算术运算，如： `total+=$2`

☞

实例：表格处理

```
$awk 'BEGIN{OFS="\t";print "Month","Salary"}\'
```

```
$1>6{print;total+= $2}\'
```

```
END{print "Total",total}' income
```

1	5200
2	5400
3	5400
4	5500
5	6000
6	5700
7	5600
8	5500
9	5400
10	5000
11	5800
12	6800



Month	Salary
7	5600
8	5500
9	5400
10	5000
11	5800
12	6800
Total	34100

处理过程：

添加表头(BEGIN)

打印并累加符合
月份>6的条件的工资

打印工资总额(END)

awk编程

∞awk变量

∞内置变量: \$0, \$1, NF, FS, OFS, RS, ORS, ...

∞自定义变量: num, total, fruit_price, ...

∞数组: price[1]=3.9 (下标形式), price["apple"]=4.5 (键值形式)

∞awk控制结构

∞判断: if(\$1>6){print \$2}

∞循环: for(i=1;i<6;i++){...}, for(i in price){...}, while(i<5){...}, do{...}while(i<5)

∞awk函数

∞内置函数: log(), sin(), index(), systime()...

∞自定义函数: function sum(){...}

awk内置变量

- ∞ \$n 当前记录的第n个字段
- ∞ \$0 整个当前记录
- ∞ NF 当前记录中的字段数（**N**umber of **F**ields）
- ∞ FS 输入的字段分隔符（默认是任何连续空白字符）（**F**ield **S**eparator）
- ∞ OFS 输出字段分隔符（默认是一个空格）（**O**utput **F**ield **S**eparator）
- ∞ RS 输入记录分隔符（默认是一个换行符）（**R**ecord **S**eparator）
- ∞ ORS 输出记录分隔符（默认是一个换行符）（**O**utput **R**ecord **S**eparator）
- ∞

awk的更多应用

∞ 打印包含某个模式的记录

```
$awk '/^(apple|banana)/' price
```

```
'BEGIN{actions} pattern{actions} END{actions}'
```

∞ 打印某个字段符合某条件的记录的部分字段

```
$awk '$2>6000 {print $1}' income
```

```
'BEGIN{actions} pattern{ actions } END{actions}'
```

∞ 重排列的位置

```
$cat price | awk '{print $1,$3,$2}'
```

```
'BEGIN{actions} pattern{ actions } END{actions}'
```

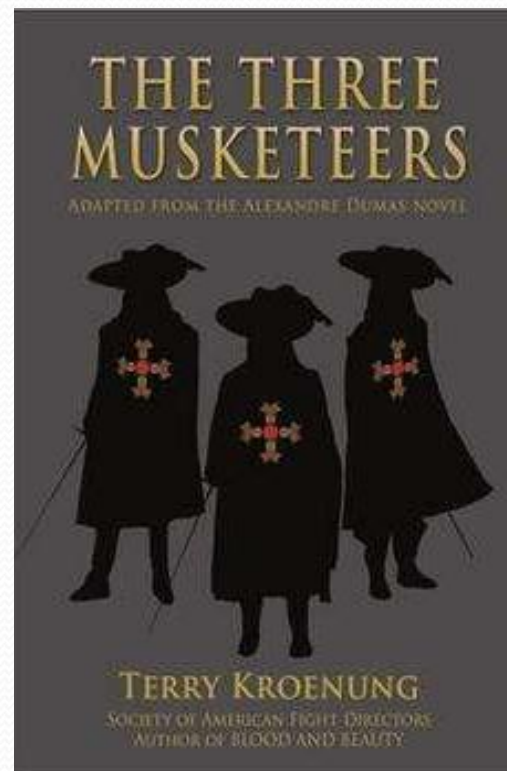
grep、sed与awk总结

∞ Linux文本处理三剑客

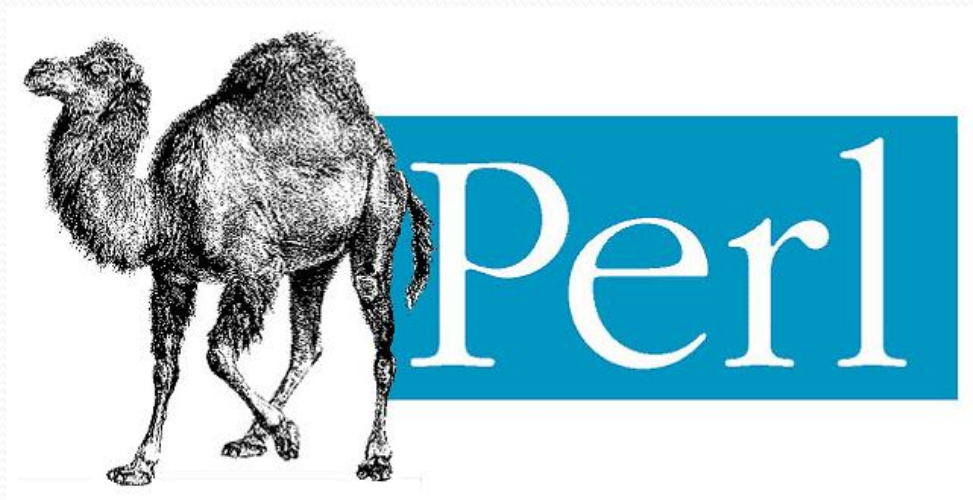
∞ grep 更适合单纯的查找或匹配文本

∞ sed 更适合编辑匹配到的文本

∞ awk 更适合格式化文本，对文本进行较复杂地格式处理



瘦死的骆驼比马大--perl命令



perl命令档案



命令全名: perl

命令用途: 文本处理, 类似于sed

命令格式:

```
perl -pe script file
```

```
perl -ne script file
```

常用选项:

-e 在命令行而不是在脚本中执行perl

-n 使 Perl 隐式地循环遍历指定的文件, 并只打印代码里规定输出的行。自动循环, 相当于 `while(<>) { 脚本; }`

-p 使 Perl 隐式地循环遍历指定的文件, 同时打印所有的行。自动循环+自动输出, 相当于 `while(<>) { 脚本; print; }`

perl命令用法

∞ 模式替换

```
$ cat plant | perl -pe 's/rose/Rose/g'
```

∞ 每行的首字母改成大写

```
$ perl -ne 'print ucfirst()' plant
```



The End