High一下!

酷 壳 - CoolShell.cn

享受编程和技术所带来的快乐 - http://coolshell.cn

请输入关键字...

AWK 简明教程

2013年2月17日 陈皓 发表评论 阅读评论 216,359 人阅读

有一些网友看了前两天的《Linux下应该知道的技巧》希望我能教教他们用awk和sed，所以，出现了这篇文章。我估计这些80后的年轻朋友可能对awk/sed这类上古神器有点陌生了，所以需要我这个老家伙来炒炒冷饭。况且，AWK是贝尔实验室1977年搞出来的文本出现神器，今年是蛇年，是AWK的本命年，而且年纪和我相仿，所以非常有必要为他写篇文章。

之所以叫AWK是因为其取了三位创始人 Alfred Aho，Peter Weinberger，和 Brian Kernighan 的Family Name的首字符。要学AWK，就得提一提AWK的一本相当经典的书《The AWK Programming Language》，它在豆瓣上的评分是9.4分！在亚马逊上居然卖1022.30元。

我在这里的教程并不想面面俱到，本文和我之前的Go语言简介一样，全是示例，基本无废话。

我只想达到两个目的：

1）你可以在乘坐公交地铁上下班，或是在坐马桶拉大便时读完（保证是一泡大便的工夫）。

2）我只想让这篇博文像一个火辣的脱衣舞女挑起你的兴趣，然后还要你自己去下工夫去撸。

废话少说，我们开始脱吧（注：这里只是topless）。

起步上台

我从netstat命令中提取了如下信息作为用例：

```
$ cat netstat.txt
Proto Recv-Q Send-Q Local-Address           Foreign-Address         State
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:9000          0.0.0.0:*               LISTEN
tcp        0      0 coolshell.cn:80         124.205.5.146:18245     TIME_WAIT
tcp        0      0 coolshell.cn:80         61.140.101.185:37538    FIN_WAIT2
tcp        0      0 coolshell.cn:80         110.194.134.189:1032    ESTABLISHED
tcp        0      0 coolshell.cn:80         123.169.124.111:49809   ESTABLISHED
tcp        0      0 coolshell.cn:80         116.234.127.77:11502    FIN_WAIT2
tcp        0      0 coolshell.cn:80         123.169.124.111:49829   ESTABLISHED
tcp        0      0 coolshell.cn:80         183.60.215.36:36970     TIME_WAIT
tcp        0   4166 coolshell.cn:80         61.148.242.38:30901     ESTABLISHED
tcp        0      1 coolshell.cn:80         124.152.181.209:26825   FIN_WAIT1
tcp        0      0 coolshell.cn:80         110.194.134.189:4796    ESTABLISHED
tcp        0      0 coolshell.cn:80         183.60.212.163:51082    TIME_WAIT
tcp        0      1 coolshell.cn:80         208.115.113.92:50601    LAST_ACK
tcp        0      0 coolshell.cn:80         123.169.124.111:49840   ESTABLISHED
tcp        0      0 coolshell.cn:80         117.136.20.85:50025     FIN_WAIT2
tcp        0      0 :::22                   :::*                    LISTEN
```

下面是最简单最常用的awk示例，其输出第1列和第4例，

- 其中单引号中的被大括号括着的就是awk的语句，注意，其只能被单引号包含。
- 其中的$1..$n表示第几例。注：$0表示整个行。

```
$ awk '{print $1, $4}' netstat.txt
Proto Local-Address
tcp 0.0.0.0:3306
tcp 0.0.0.0:80
tcp 127.0.0.1:9000
tcp coolshell.cn:80
tcp coolshell.cn:80
tcp coolshell.cn:80
tcp coolshell.cn:80
tcp coolshell.cn:80
tcp coolshell.cn:80
tcp coolshell.cn:80
tcp coolshell.cn:80
tcp coolshell.cn:80
tcp coolshell.cn:80
tcp coolshell.cn:80
tcp coolshell.cn:80
tcp coolshell.cn:80
tcp coolshell.cn:80
tcp coolshell.cn:80
tcp :::22
```

我们再来看看awk的格式化输出，和C语言的printf没什么两样：

```
$ awk '{printf "%-8s %-8s %-8s %-18s %-22s %-15s\n",$1,$2,$3,$4,$5,$6}' netstat.txt
```

```
Proto    Recv-Q   Send-Q   Local-Address      Foreign-Address        State
tcp      0        0        0.0.0.0:3306       0.0.0.0:*              LISTEN
tcp      0        0        0.0.0.0:80         0.0.0.0:*              LISTEN
tcp      0        0        127.0.0.1:9000     0.0.0.0:*              LISTEN
tcp      0        0        coolshell.cn:80    124.205.5.146:18245    TIME_WAIT
tcp      0        0        coolshell.cn:80    61.140.101.185:37538   FIN_WAIT2
tcp      0        0        coolshell.cn:80    110.194.134.189:1032   ESTABLISHED
tcp      0        0        coolshell.cn:80    123.169.124.111:49809  ESTABLISHED
tcp      0        0        coolshell.cn:80    116.234.127.77:11502   FIN_WAIT2
tcp      0        0        coolshell.cn:80    123.169.124.111:49829  ESTABLISHED
tcp      0        0        coolshell.cn:80    183.60.215.36:36970    TIME_WAIT
tcp      0        4166     coolshell.cn:80    61.148.242.38:30901    ESTABLISHED
tcp      0        1        coolshell.cn:80    124.152.181.209:26825  FIN_WAIT1
tcp      0        0        coolshell.cn:80    110.194.134.189:4796   ESTABLISHED
tcp      0        0        coolshell.cn:80    183.60.212.163:51082   TIME_WAIT
tcp      0        1        coolshell.cn:80    208.115.113.92:50601   LAST_ACK
tcp      0        0        coolshell.cn:80    123.169.124.111:49840  ESTABLISHED
tcp      0        0        coolshell.cn:80    117.136.20.85:50025    FIN_WAIT2
tcp      0        0        :::22              :::*                   LISTEN
```

脱掉外套

过滤记录

我们再来看看如何过滤记录（下面过滤条件为：第三列的值为0 && 第6列的值为LISTEN）

```
$ awk '$3==0 && $6=="LISTEN" ' netstat.txt
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:9000          0.0.0.0:*               LISTEN
tcp        0      0 :::22                   :::*                    LISTEN
```

其中的“==”为比较运算符。其他比较运算符：!=, >, <, >=, <=

我们来看看各种过滤记录的方式：

```
$ awk ' $3>0 {print $0}' netstat.txt
Proto Recv-Q Send-Q Local-Address          Foreign-Address            State
tcp        0   4166 coolshell.cn:80        61.148.242.38:30901        ESTABLISHED
tcp        0      1 coolshell.cn:80        124.152.181.209:26825      FIN_WAIT1
tcp        0      1 coolshell.cn:80        208.115.113.92:50601       LAST_ACK
```

如果我们需要表头的话，我们可以引入内建变量NR：

```
$ awk '$3==0 && $6=="LISTEN" || NR==1 ' netstat.txt
Proto Recv-Q Send-Q Local-Address          Foreign-Address            State
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:9000          0.0.0.0:*               LISTEN
tcp        0      0 :::22                   :::*                    LISTEN
```

再加上格式化输出：

```
$ awk '$3==0 && $6=="LISTEN" || NR==1 {printf "%-20s %-20s %s\n",$4,$5,$6}' netstat.txt
```

```
Local-Address       Foreign-Address     State
0.0.0.0:3306        0.0.0.0:*           LISTEN
0.0.0.0:80          0.0.0.0:*           LISTEN
127.0.0.1:9000      0.0.0.0:*           LISTEN
:::22               :::*                LISTEN
```

内建变量

说到了内建变量，我们可以来看看awk的一些内建变量：

$0	当前记录（这个变量中存放着整个行的内容）
$1~$n	当前记录的第n个字段，字段间由FS分隔
FS	输入字段分隔符 默认是空格或Tab
NF	当前记录中的字段个数，就是有多少列
NR	已经读出的记录数，就是行号，从1开始，如果有多个文件话，这个值也是不断累加中。
FNR	当前记录数，与NR不同的是，这个值会是各个文件自己的行号
RS	输入的记录分隔符， 默认为换行符
OFS	输出字段分隔符， 默认也是空格
ORS	输出的记录分隔符，默认为换行符
FILENAME	当前输入文件的名字

怎么使用呢，比如：我们如果要输出行号：

```
$ awk '$3==0 && $6=="ESTABLISHED" || NR==1 {printf "%02s %s %-20s %-20s %s\n",NR, FNR, $4,$5
01 1 Local-Address       Foreign-Address      State
07 7 coolshell.cn:80     110.194.134.189:1032 ESTABLISHED
08 8 coolshell.cn:80     123.169.124.111:49809 ESTABLISHED
10 10 coolshell.cn:80     123.169.124.111:49829 ESTABLISHED
14 14 coolshell.cn:80     110.194.134.189:4796 ESTABLISHED
17 17 coolshell.cn:80     123.169.124.111:49840 ESTABLISHED
```

指定分隔符

```
$  awk  'BEGIN{FS=":"} {print $1,$3,$6}' /etc/passwd
root 0 /root
bin 1 /bin
daemon 2 /sbin
adm 3 /var/adm
lp 4 /var/spool/lpd
sync 5 /sbin
shutdown 6 /sbin
halt 7 /sbin
```

上面的命令也等价于：（-F的意思就是指定分隔符）

```
$ awk  -F: '{print $1,$3,$6}' /etc/passwd
```

注：如果你要指定多个分隔符，你可以这样来：

```
awk -F '[;:]'
```

再来看一个以\t作为分隔符输出的例子（下面使用了/etc/passwd文件，这个文件是以:分隔的）：

```
$ awk  -F: '{print $1,$3,$6}' OFS="\t" /etc/passwd
root    0       /root
bin     1       /bin
daemon  2       /sbin
adm     3       /var/adm
lp      4       /var/spool/lpd
sync    5       /sbin
```

脱掉衬衫

字符串匹配

我们再来看几个字符串匹配的示例：

```
$ awk '$6 ~ /FIN/ || NR==1 {print NR,$4,$5,$6}' OFS="\t" netstat.txt
1       Local-Address   Foreign-Address State
6       coolshell.cn:80 61.140.101.185:37538    FIN_WAIT2
9       coolshell.cn:80 116.234.127.77:11502    FIN_WAIT2
13      coolshell.cn:80 124.152.181.209:26825   FIN_WAIT1
18      coolshell.cn:80 117.136.20.85:50025     FIN_WAIT2

$ $ awk '$6 ~ /WAIT/ || NR==1 {print NR,$4,$5,$6}' OFS="\t" netstat.txt
1       Local-Address   Foreign-Address State
5       coolshell.cn:80 124.205.5.146:18245     TIME_WAIT
6       coolshell.cn:80 61.140.101.185:37538    FIN_WAIT2
9       coolshell.cn:80 116.234.127.77:11502    FIN_WAIT2
11      coolshell.cn:80 183.60.215.36:36970     TIME_WAIT
13      coolshell.cn:80 124.152.181.209:26825   FIN_WAIT1
15      coolshell.cn:80 183.60.212.163:51082    TIME_WAIT
18      coolshell.cn:80 117.136.20.85:50025     FIN_WAIT2
```

上面的第一个示例匹配FIN状态， 第二个示例匹配WAIT字样的状态。其实 ~ 表示模式开始。/ /中是模式。这就是一个正则表达式的匹配。

其实awk可以像grep一样的去匹配第一行，就像这样：

```
$ awk '/LISTEN/' netstat.txt
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:9000          0.0.0.0:*               LISTEN
tcp        0      0 :::22                   :::*                    LISTEN
```

我们可以使用 “/FIN|TIME/” 来匹配 FIN 或者 TIME ：

```
$ awk '$6 ~ /FIN|TIME/ || NR==1 {print NR,$4,$5,$6}' OFS="\t" netstat.txt
1       Local-Address   Foreign-Address State
5       coolshell.cn:80 124.205.5.146:18245     TIME_WAIT
6       coolshell.cn:80 61.140.101.185:37538    FIN_WAIT2
9       coolshell.cn:80 116.234.127.77:11502    FIN_WAIT2
11      coolshell.cn:80 183.60.215.36:36970     TIME_WAIT
```

```
13      coolshell.cn:80 124.152.181.209:26825   FIN_WAIT1
15      coolshell.cn:80 183.60.212.163:51082    TIME_WAIT
18      coolshell.cn:80 117.136.20.85:50025     FIN_WAIT2
```

再来看看模式取反的例子：

```
$ awk '$6 !~ /WAIT/ || NR==1 {print NR,$4,$5,$6}' OFS="\t" netstat.txt
1       Local-Address   Foreign-Address State
2       0.0.0.0:3306    0.0.0.0:*       LISTEN
3       0.0.0.0:80      0.0.0.0:*       LISTEN
4       127.0.0.1:9000  0.0.0.0:*       LISTEN
7       coolshell.cn:80 110.194.134.189:1032    ESTABLISHED
8       coolshell.cn:80 123.169.124.111:49809   ESTABLISHED
10      coolshell.cn:80 123.169.124.111:49829   ESTABLISHED
12      coolshell.cn:80 61.148.242.38:30901     ESTABLISHED
14      coolshell.cn:80 110.194.134.189:4796    ESTABLISHED
16      coolshell.cn:80 208.115.113.92:50601    LAST_ACK
17      coolshell.cn:80 123.169.124.111:49840   ESTABLISHED
19      :::22   :::*    LISTEN
```

或是：

```
awk '!/WAIT/' netstat.txt
```

折分文件

awk拆分文件很简单，使用重定向就好了。下面这个例子，是按第6例分隔文件，相当的简单（其中的NR!=1表示不处理表头）。

```
$ awk 'NR!=1{print > $6}' netstat.txt

$ ls
ESTABLISHED  FIN_WAIT1  FIN_WAIT2  LAST_ACK  LISTEN  netstat.txt  TIME_WAIT

$ cat ESTABLISHED
tcp        0      0 coolshell.cn:80         110.194.134.189:1032        ESTABLISHED
tcp        0      0 coolshell.cn:80         123.169.124.111:49809       ESTABLISHED
tcp        0      0 coolshell.cn:80         123.169.124.111:49829       ESTABLISHED
tcp        0   4166 coolshell.cn:80         61.148.242.38:30901         ESTABLISHED
tcp        0      0 coolshell.cn:80         110.194.134.189:4796        ESTABLISHED
tcp        0      0 coolshell.cn:80         123.169.124.111:49840       ESTABLISHED

$ cat FIN_WAIT1
tcp        0      1 coolshell.cn:80         124.152.181.209:26825       FIN_WAIT1

$ cat FIN_WAIT2
tcp        0      0 coolshell.cn:80         61.140.101.185:37538        FIN_WAIT2
tcp        0      0 coolshell.cn:80         116.234.127.77:11502        FIN_WAIT2
tcp        0      0 coolshell.cn:80         117.136.20.85:50025         FIN_WAIT2

$ cat LAST_ACK
tcp        0      1 coolshell.cn:80         208.115.113.92:50601        LAST_ACK

$ cat LISTEN
```

```
tcp        0      0 0.0.0.0:3306            0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:9000          0.0.0.0:*                   LISTEN
tcp        0      0 :::22                   :::*                        LISTEN

$ cat TIME_WAIT
tcp        0      0 coolshell.cn:80         124.205.5.146:18245         TIME_WAIT
tcp        0      0 coolshell.cn:80         183.60.215.36:36970         TIME_WAIT
tcp        0      0 coolshell.cn:80         183.60.212.163:51082        TIME_WAIT
```

你也可以把指定的列输出到文件：

```
awk 'NR!=1{print $4,$5 > $6}' netstat.txt
```

再复杂一点：（注意其中的if-else-if语句，可见awk其实是个脚本解释器）

```
$ awk 'NR!=1{if($6 ~ /TIME|ESTABLISHED/) print > "1.txt";
else if($6 ~ /LISTEN/) print > "2.txt";
else print > "3.txt" }' netstat.txt

$ ls ?.txt
1.txt  2.txt  3.txt

$ cat 1.txt
tcp        0      0 coolshell.cn:80         124.205.5.146:18245         TIME_WAIT
tcp        0      0 coolshell.cn:80         110.194.134.189:1032        ESTABLISHED
tcp        0      0 coolshell.cn:80         123.169.124.111:49809       ESTABLISHED
tcp        0      0 coolshell.cn:80         123.169.124.111:49829       ESTABLISHED
tcp        0      0 coolshell.cn:80         183.60.215.36:36970         TIME_WAIT
tcp        0   4166 coolshell.cn:80         61.148.242.38:30901         ESTABLISHED
tcp        0      0 coolshell.cn:80         110.194.134.189:4796        ESTABLISHED
tcp        0      0 coolshell.cn:80         183.60.212.163:51082        TIME_WAIT
tcp        0      0 coolshell.cn:80         123.169.124.111:49840       ESTABLISHED

$ cat 2.txt
tcp        0      0 0.0.0.0:3306            0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:9000          0.0.0.0:*                   LISTEN
tcp        0      0 :::22                   :::*                        LISTEN

$ cat 3.txt
tcp        0      0 coolshell.cn:80         61.140.101.185:37538        FIN_WAIT2
tcp        0      0 coolshell.cn:80         116.234.127.77:11502        FIN_WAIT2
tcp        0      1 coolshell.cn:80         124.152.181.209:26825       FIN_WAIT1
tcp        0      1 coolshell.cn:80         208.115.113.92:50601        LAST_ACK
tcp        0      0 coolshell.cn:80         117.136.20.85:50025         FIN_WAIT2
```

统计

下面的命令计算所有的C文件，CPP文件和H文件的文件大小总和。

```
$ ls -l  *.cpp *.c *.h | awk '{sum+=$5} END {print sum}'
2511401
```

我们再来看一个统计各个connection状态的用法：（我们可以看到一些编程的影子了，大家都是程序员我就不解释了。注意其中的数组的用法）

```
$ awk 'NR!=1{a[$6]++;} END {for (i in a) print i ", " a[i];}' netstat.txt
TIME_WAIT, 3
FIN_WAIT1, 1
ESTABLISHED, 6
FIN_WAIT2, 3
LAST_ACK, 1
LISTEN, 4
```

再来看看统计每个用户的进程的占了多少内存（注：sum的RSS那一列）

```
$ ps aux | awk 'NR!=1{a[$1]+=$6;} END { for(i in a) print i ", " a[i]"KB";}'
dbus, 540KB
mysql, 99928KB
www, 3264924KB
root, 63644KB
hchen, 6020KB
```

脱掉内衣

awk脚本

在上面我们可以看到一个END关键字。END的意思是“处理完所有的行的标识”，即然说到了END就有必要介绍一下BEGIN，这两个关键字意味着执行前和执行后的意思，语法如下：

- BEGIN{ 这里面放的是执行前的语句 }
- END {这里面放的是处理完所有的行后要执行的语句 }
- {这里面放的是处理每一行时要执行的语句}

为了说清楚这个事，我们来看看下面的示例：

假设有这么一个文件（学生成绩表）：

```
$ cat score.txt
Marry   2143 78 84 77
Jack    2321 66 78 45
Tom     2122 48 77 71
Mike    2537 87 97 95
Bob     2415 40 57 62
```

我们的awk脚本如下（我没有写有命令行上是因为命令行上不易读，另外也在介绍另一种用法）：

```
$ cat cal.awk
#!/bin/awk -f
#运行前
BEGIN {
    math = 0
    english = 0
    computer = 0

    printf "NAME    NO.   MATH  ENGLISH  COMPUTER   TOTAL\n"
    printf "---------------------------------------------\n"
}
```

```
#运行中
{
    math+=$3
    english+=$4
    computer+=$5
    printf "%-6s %-6s %4d %8d %8d %8d\n", $1, $2, $3,$4,$5, $3+$4+$5
}
#运行后
END {
    printf "-----------------------------------------\n"
    printf "  TOTAL:%10d %8d %8d \n", math, english, computer
    printf "AVERAGE:%10.2f %8.2f %8.2f\n", math/NR, english/NR, computer/NR
}
```

我们来看一下执行结果：（也可以这样运行 ./cal.awk score.txt）

```
$ awk -f cal.awk score.txt
NAME    NO.   MATH  ENGLISH  COMPUTER   TOTAL
---------------------------------------------
Marry  2143     78       84       77       239
Jack   2321     66       78       45       189
Tom    2122     48       77       71       196
Mike   2537     87       97       95       279
Bob    2415     40       57       62       159
---------------------------------------------
  TOTAL:        319      393      350
AVERAGE:      63.80    78.60    70.00
```

环境变量

即然说到了脚本，我们来看看怎么和环境变量交互：（使用-v参数和ENVIRON，使用ENVIRON的环境变量需要export）

```
$ x=5

$ y=10
$ export y

$ echo $x $y
5 10

$ awk -v val=$x '{print $1, $2, $3, $4+val, $5+ENVIRON["y"]}' OFS="\t" score.txt
Marry   2143    78      89      87
Jack    2321    66      83      55
Tom     2122    48      82      81
Mike    2537    87      102     105
Bob     2415    40      62      72
```

几个花活

最后，我们再来看几个小例子：

```
#从file文件中找出长度大于80的行
```

```
awk 'length>80' file

#按连接数查看客户端IP
netstat -ntu | awk '{print $5}' | cut -d: -f1 | sort | uniq -c | sort -nr

#打印99乘法表
seq 9 | sed 'H;g' | awk -v RS='' '{for(i=1;i<=NF;i++)printf("%dx%d=%d%s", i, NR, i*NR, i==NR
```

自己撸吧

关于其中的一些知识点可以参看gawk的手册：

- 内建变量，参看：http://www.gnu.org/software/gawk/manual/gawk.html#Built_002din-Variables
- 流控方面，参看：http://www.gnu.org/software/gawk/manual/gawk.html#Statements
- 内建函数，参看：http://www.gnu.org/software/gawk/manual/gawk.html#Built_002din
- 正则表达式，参看：http://www.gnu.org/software/gawk/manual/gawk.html#Regexp

（全文完）

关注CoolShell微信公众账号可以在手机端搜索文章

177

分类：Unix/Linux， 杂项资源， 编程工具 标签：awk， gawk， Linux， Shell， Unix

（88 人打了分，平均分： 4.68 ）

相关文章

- 2012年11月23日 你可能不知道的Shell
- 2013年01月09日 应该知道的Linux技巧
- 2013年04月26日 Unix考古记：一个“遗失”的shell
- 2013年02月20日 sed 简明教程
- 2012年07月11日 28个Unix/Linux的命令行神器
- 2010年08月24日 使用grep恢复被删文件内容
- 2009年06月21日 Linux/Unix 新手和专家教程
- 2009年06月12日 Unix 40年：Unix年鉴

评论 (2) Trackbacks (7) 发表评论 Trackback

1. 百里
   2016年7月27日19:30 | #1
   回复 | 引用

   很熟悉套路，你看就是老司机~

2. 要饭真得好难
   2016年10月21日09:55 | #2
   回复 | 引用

   AWK是贝尔实验室1977年搞出来的文本出现神器 =》 文本处理神器

评论分页
« 上一页 1 … 3 4 5 9070

1. 2016年4月27日20:39 | #1
   qlcoder跬步 - skysider's blog
2. 2016年5月19日08:03 | #2
   使用GoAccess分析Nginx日志以及sed/awk手动分析实践 (github.io)
3. 2016年8月16日11:27 | #3
   sed 简明教程 - 17haha-blog
4. 2016年8月22日10:02 | #4
   Linux常用缩写 | Wangmk
5. 2016年9月10日09:32 | #5
   Java Web技术经验总结（三） | 编程,编程,再编程!
6. 2016年10月6日20:43 | #6
   Linux学习笔记之awk - 赵鹏的个人博客
7. 2016年10月20日17:30 | #7
   AWK 简明教程-运维博客

昵称（必填）
电子邮箱（我们会为您保密）（必填）
网址

订阅评论

提交评论

sed 简明教程 Linus：利用二级指针删除单向链表

订阅

Twitter

本站公告

访问 酷壳404页面 寻找遗失儿童！

酷壳建议大家多使用RSS访问阅读（本站已经是全文输出，推荐使用cloud.feedly.com 或digg.com）。有相关事宜欢迎电邮：haoel(at)hotmail.com。最后，感谢大家对酷壳的支持和体谅！

最新文章

- 如何读懂并写出装逼的函数式代码
- 什么是工程师文化？
- 关于高可用的系统
- 这多年来我一直在钻研的技术
- 缓存更新的套路
- 为什么我不在微信公众号上写文章
- 性能测试应该怎么做？
- 让我们来谈谈分工
- Cuckoo Filter：设计与实现
- Docker基础技术：DeviceMapper
- Docker基础技术：AUFS
- Docker基础技术：Linux CGroup
- Docker基础技术：Linux Namespace（上）
- Docker基础技术：Linux Namespace（下）
- 关于移动端的钓鱼式攻击
- Linus：为何对象引用计数必须是原子的
- DHH 谈混合移动应用开发
- HTML6 展望
- Google Inbox如何跨平台重用代码？
- vfork 挂掉的一个问题
- Leetcode 编程训练
- State Threads 回调终结者
- bash代码注入的安全漏洞
- 互联网之子 - Aaron Swartz
- 谜题的答案和活动的心得体会
- 【活动】解谜题送礼物
- 开发团队的效率
- TCP 的那些事儿（下）
- TCP 的那些事儿（上）
- 「我只是认真」聊聊工匠情怀

全站热门

- 程序员技术练级攻略

- 简明 Vim 练级攻略
- 做个环保主义的程序员
- 如何学好C语言
- AWK 简明教程
- TCP 的那些事儿（上）
- 应该知道的Linux技巧
- “21天教你学会C++”
- 6个变态的C语言Hello World程序
- 编程能力与编程年龄
- 由12306.cn谈谈网站性能技术
- sed 简明教程
- 28个Unix/Linux的命令行神器
- “作环保的程序员，从不用百度开始”
- 我是怎么招聘程序员的
- 性能调优攻略
- 二维码的生成细节和原理
- Lua简明教程
- MySQL性能优化的最佳20+条经验
- Web开发中需要了解的东西
- C++ 程序员自信心曲线图
- 如何学好C++语言
- Android将允许纯C/C++开发应用
- 无插件Vim编程技巧
- 如何写出无法维护的代码
- 20本最好的Linux免费书籍
- Windows编程革命简史
- 加班与效率
- 编程真难啊
- 分布式系统的事务处理

新浪微博

标签

agile AJAX Algorithm Android Bash C++ Coding CSS Database Design design pattern ebook Flash Game Go Google HTML IE Java Javascript jQuery Linux MySQL OOP password Performance PHP Programmer Programming programming language Puzzle Python Ruby SQL TDD UI Unix vim Web Windows XML 安全 程序员 算法 面试

分类目录

- .NET编程 (3)
- Ajax开发 (9)
- C/C++语言 (71)
- Erlang (1)
- Java语言 (32)

- PHP脚本 (11)
- Python (23)
- Ruby (5)
- Unix/Linux (75)
- Web开发 (103)
- Windows (12)
- 业界新闻 (26)
- 企业应用 (2)
- 技术新闻 (33)
- 技术管理 (15)
- 技术读物 (117)
- 操作系统 (49)
- 数据库 (11)
- 杂项资源 (271)
- 流程方法 (48)
- 程序设计 (88)
- 系统架构 (9)
- 编程工具 (65)
- 编程语言 (175)
- 网络安全 (27)
- 职场生涯 (34)
- 趣味问题 (19)
- 轶事趣闻 (147)

归档

- 2016年十月 (1)
- 2016年九月 (1)
- 2016年八月 (2)
- 2016年七月 (3)
- 2015年十二月 (1)
- 2015年九月 (1)
- 2015年八月 (2)
- 2015年四月 (4)
- 2014年十二月 (3)
- 2014年十一月 (2)
- 2014年十月 (2)
- 2014年九月 (2)
- 2014年八月 (2)
- 2014年六月 (1)
- 2014年五月 (4)
- 2014年四月 (4)
- 2014年三月 (5)
- 2014年二月 (3)
- 2014年一月 (2)
- 2013年十二月 (3)
- 2013年十一月 (1)
- 2013年十月 (6)
- 2013年八月 (1)
- 2013年七月 (8)
- 2013年六月 (2)
- 2013年五月 (3)
- 2013年四月 (3)
- 2013年三月 (3)
- 2013年二月 (5)
- 2013年一月 (1)
- 2012年十二月 (4)

- 2012年十一月 (4)
- 2012年十月 (3)
- 2012年九月 (4)
- 2012年八月 (8)
- 2012年七月 (4)
- 2012年六月 (7)
- 2012年五月 (6)
- 2012年四月 (6)
- 2012年三月 (6)
- 2012年二月 (3)
- 2012年一月 (6)
- 2011年十二月 (5)
- 2011年十一月 (9)
- 2011年十月 (6)
- 2011年九月 (5)
- 2011年八月 (14)
- 2011年七月 (6)
- 2011年六月 (12)
- 2011年五月 (5)
- 2011年四月 (18)
- 2011年三月 (16)
- 2011年二月 (16)
- 2011年一月 (18)
- 2010年十二月 (11)
- 2010年十一月 (11)
- 2010年十月 (19)
- 2010年九月 (15)
- 2010年八月 (10)
- 2010年七月 (20)
- 2010年六月 (9)
- 2010年五月 (13)
- 2010年四月 (12)
- 2010年三月 (11)
- 2010年二月 (7)
- 2010年一月 (9)
- 2009年十二月 (22)
- 2009年十一月 (27)
- 2009年十月 (17)
- 2009年九月 (14)
- 2009年八月 (21)
- 2009年七月 (18)
- 2009年六月 (19)
- 2009年五月 (27)
- 2009年四月 (53)
- 2009年三月 (43)
- 2008年十月 (1)
- 2007年十二月 (1)
- 2006年十一月 (1)
- 2004年六月 (1)

最新评论

- zhouzhou: 赞同。之前几次面试，因为过于紧张发挥不好。
- 霏霏: 不错！分享下分布式事务解决方案的效果演示（结合支付系统真实应 用场景）：http://www.iqiyi.com/w_19rsveq lhh.html
- forP: 真的会被同事打的~~
- chalife: 好多错的

- Rogue: 我有一个问题：在带多个参数及多个decorator里，wra pper为什么一定要return,我写成print就报错： print “” + fn(*args, **kwargs) +...
- 阿瞒: 陈大哥，有关C++有个问题想请教下您， 两个函数原型的区别void pt (int *), void pt (int (&a)[10]，两个函数接受的参数并不一样，比如i nt a[3] = {1, 2, 3} int...
- 小小啊强: 天天如此循环。。。。。。。。。
- Zony: 十分赞同博主的看法，与强者为伍。
- yanse: 看了这篇文章，收获颇丰，顺带贡献一个脚本（脚本虽然比较简单， 但是是之前从网上找的，不是我写的）。 脚本作用很简单。就是拷贝Linux下面的程序和程序所依赖的库 文件到指定的 rootfs 。...
- xumenger: 博主说在最新的STL中，string的COW特性被去掉了，大 概对应VC++、g++的哪些版本？ 另外在VC++6.0中如果我想通过指针获取一个string的 引用计数值应该定位到哪个地址？
- NoAnyLove: Ubuntu 16.04 x64，gcc 5.4下，char s[0]的例子不再使用lea指令，而是将0x4的偏移加到了r ax上， 0x40052e movq $0x0,-0x8(%rbp) //...
- 墨梅: 看了您的技术练级攻略，几年前的文章，偶然发现这里，内容不错， 陈老师也是一个用心的人。作为一个干了几年技术，如今对未来有点 手足无措的人，您的文章给我了一起启发，表示感谢。
- zyx954: 试了一下 还真挺管用 -_-!
- xuyue: @andrew_show 有的场景是没法解的，比如要统计单机在高并发下的实时qps。
- Charlie: 看了博主的文章，深感自己职业生涯混乱，完全没有规划，没有入行 。做过global support，解决问题的思维方式变成了如何快速找到别人曾有 的解决办法和回答客户提的问题，对技术问题缺乏深度研究。...

友情链接

- 陈皓的博客
- 并发编程
- 四火的唠叨
- HelloGcc Working Group
- 吕毅的Blog
- Todd Wei的Blog
- C++爱好者博客
- HTML5研究小组
- 朱文昊Albert Zhu
- C瓜哥的博客
- 开源吧
- ACMer
- 陈鹏个人博客
- OneCoder
- More Than Vimer
- 运维派
- 书巢

功能

- 注册
- 登录
- 文章RSS
- 评论RSS
- WordPress.org