

# 基础实验篇第2章

---

Shell编程： 批量文本文件内容过滤



1

Shell字符处理

2

Shell文件读写

3

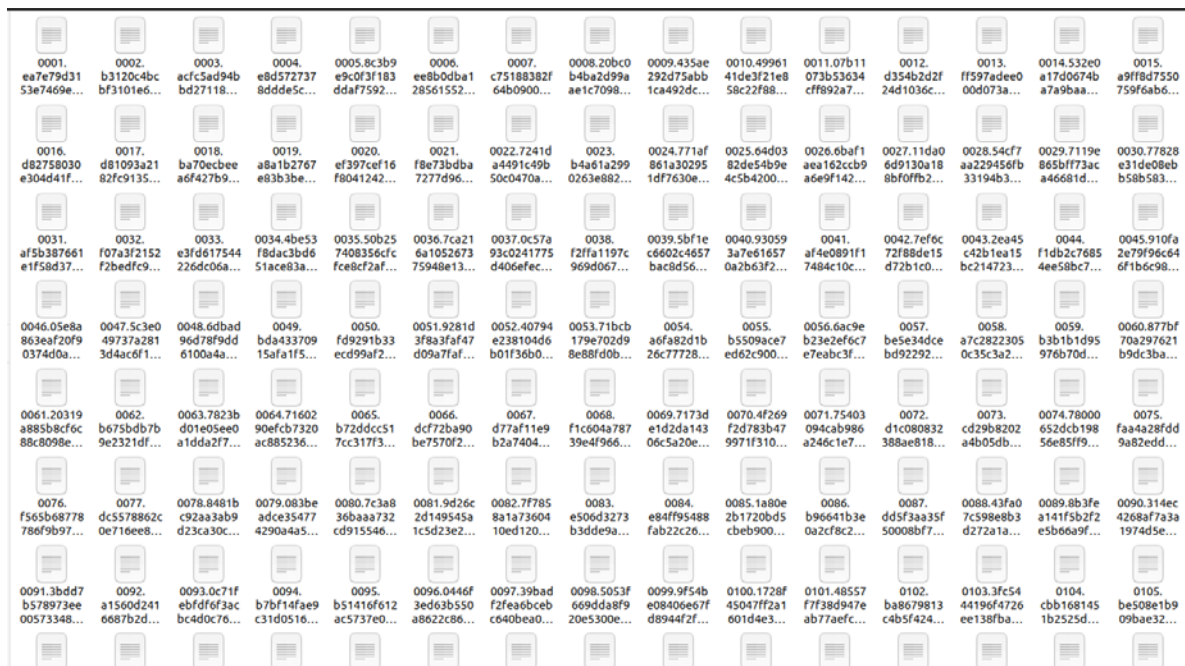
Linux定时任务

4

Shell实战：批量文本文件内容过滤

5

Linux服务管理扩展阅读



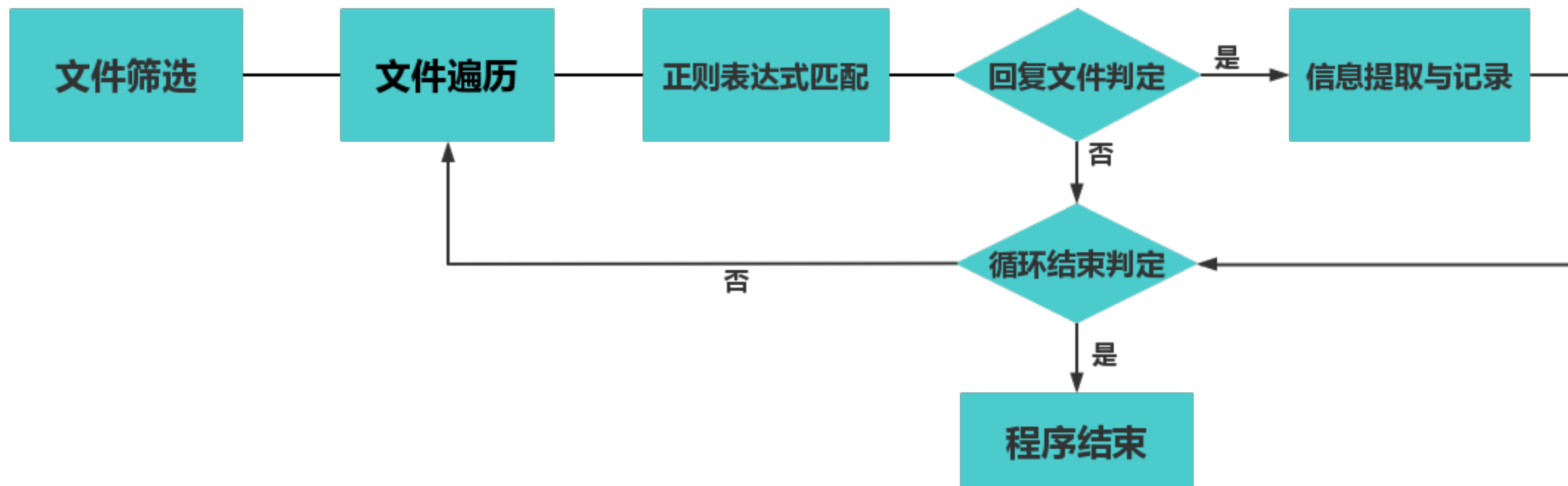
大量格式相似的文本文件

```
Open [icon] 0001.ea7e79d31535e7469e7a9c3e0af6a357e ~/Desktop/ham

1 From exmh-workers-admin@redhat.com Thu Aug 22 12:36:23 2002
2 Return-Path: <exmh-workers-admin@example.com>
3 Delivered-To: zzzz@localhost.netnoteinc.com
4 Received: from localhost (localhost [127.0.0.1])
5   by phobos.labs.netnoteinc.com (Postfix) with ESMTPE id D03E543C36
6   for <zzzz@localhost>; Thu, 22 Aug 2002 07:36:16 -0400 (EDT)
7 Received: from phobos [127.0.0.1]
8   by localhost with IMAP (fetchmail-5.9.0)
9   for zzzz@localhost (single-drop); Thu, 22 Aug 2002 12:36:16 +0100 (IST)
10 Received: from listman.example.com (listman.example.com [66.187.233.211]) by
11   dogma.slashnull.org (8.11.6/8.11.6) with ESMTPE id g7MBYrZ04811 for
12   <zzzz-exmh@example.com>; Thu, 22 Aug 2002 12:34:53 +0100
13 Received: from listman.example.com (localhost.localdomain [127.0.0.1]) by
14   listman.redhat.com (Postfix) with ESMTPE id 8386540858; Thu, 22 Aug 2002
15   07:35:02 -0400 (EDT)
16 Delivered-To: exmh-workers@listman.example.com
17 Received: from int-mx1.corp.example.com (int-mx1.corp.example.com
18   [172.16.52.254]) by listman.redhat.com (Postfix) with ESMTPE id 10CF8406D7
19   for <exmh-workers@listman.redhat.com>; Thu, 22 Aug 2002 07:34:10 -0400
20   (EDT)
21 Received: (from mail@localhost) by int-mx1.corp.example.com (8.11.6/8.11.6)
22   id g7MBY7g11259 for exmh-workers@listman.redhat.com; Thu, 22 Aug 2002
23   07:34:07 -0400
24 Received: from mx1.example.com (mx1.example.com [172.16.48.31]) by
25   int-mx1.corp.redhat.com (8.11.6/8.11.6) with SMTP id g7MBY7Y11255 for
26   <exmh-workers@redhat.com>; Thu, 22 Aug 2002 07:34:07 -0400
27 Received: from ratree.psu.ac.th ([202.28.97.6]) by mx1.example.com
28   (8.11.6/8.11.6) with SMTP id g7MBIhl25223 for <exmh-workers@redhat.com>;
29   Thu, 22 Aug 2002 07:18:55 -0400
30 Received: from delta.cs.mu.OZ.AU (delta.coe.psu.ac.th [172.30.0.98]) by
31   ratree.psu.ac.th (8.11.6/8.11.6) with ESMTPE id g7MBWcl29762;
32   Thu, 22 Aug 2002 18:32:40 +0700 (ICT)
33 Received: from munnari.OZ.AU (localhost [127.0.0.1]) by delta.cs.mu.OZ.AU
34   (8.11.6/8.11.6) with ESMTPE id g7MBQPW13260; Thu, 22 Aug 2002 18:26:25
35   +0700 (ICT)
36 From: Robert Elz <kre@munnnari.OZ.AU>
37 To: Chris Garrigues <cwg-dated-1030377287.06fa6d@DeepEddy.Com>
38 Cc: exmh-workers@example.com
39 Subject: Re: New Sequences Window
40 In-Reply-To: <1029945287.4797.TMDA@deepeddy.vircio.com>
41 References: <1029945287.4797.TMDA@deepeddy.vircio.com>
42   <1029882468.3116.TMDA@deepeddy.vircio.com> <9627.1029933001@munnnari.OZ.AU>
43   <1029943066.26919.TMDA@deepeddy.vircio.com>
44   <1029944441.308.TMDA@deepeddy.vircio.com>
```

发件人、主题等信息单独成行

想从中找出所有的回复文件——利用shell编程和正则表达式，找出Subject一行中包含“Re”或“RE”的文件即可



**文件遍历：** 使用迭代结构循环遍历文件夹中的文件。

**正则表达式匹配：** 采用正则表达式对字符串进行模式匹配，以便识别和提取回复文件。

# Shell字符处理

---

## 字符串声明

1、单引号' '定义：任何字符都会被视为字面量，不会被Shell解释，会原样输出。在其中使用变量是无效的，并且字符串中不能出现单引号，即使对单引号进行转义也不行

```
user@os-virtual-machine:~/Desktop$ str1='This is a string'
user@os-virtual-machine:~/Desktop$ str2='This is a $str1'
user@os-virtual-machine:~/Desktop$ echo $str1
This is a string
user@os-virtual-machine:~/Desktop$ echo $str2
This is a $str1
```

str2中的\$str1被看做是一个字符串，而不是变量。

## 字符串声明

2、双引号" "定义：如果其中包含了某个变量或特殊字符，那么该变量会被解析，得到该变量的值，并且字符串中可以出现双引号，只要它被转义了就行

```
user@os-virtual-machine:~/Desktop$ str1="string"
user@os-virtual-machine:~/Desktop$ str2="This is a $str1"
user@os-virtual-machine:~/Desktop$ str3="This is a \"string\""
user@os-virtual-machine:~/Desktop$ echo $str1
string
user@os-virtual-machine:~/Desktop$ echo $str2
This is a string
user@os-virtual-machine:~/Desktop$ echo $str3
This is a "string"
```

str2中的\$str1是一个变量，会被Shell解释器解析；str3中的双引号则被转义符转义了，所以可以出现。



## 字符串声明

3、反引号` ` 定义：用于将命令输出作为字符串赋值给变量

```
user@os-virtual-machine:~/Desktop$ str1=`date`  
user@os-virtual-machine:~/Desktop$ echo $str1  
2023年 03月 30日 星期四 16:04:41 CST
```

`date`会被解释成一个命令，执行后会返回当前日期和时间的字符串，这个字符串会被赋值给变量str1。

4、\$( )定义：将命令输出作为字符串赋值给变量，与反引号类似

```
user@os-virtual-machine:~/Desktop$ str1=$(date)  
user@os-virtual-machine:~/Desktop$ echo $str1  
2023年 03月 30日 星期四 16:09:47 CST
```

\$(date)会被解释成一个命令，执行后会返回当前日期和时间的字符串，这个字符串会被赋值给变量str1。



## 字符串声明

5、不被引号包围：字符串中出现变量时也会被解析，这一点和双引号包围的字符串一样。字符串中不能出现空格，否则空格后边的字符串会作为其他变量或者命令解析。

```
user@os-virtual-machine:~/Desktop$ str1=string
user@os-virtual-machine:~/Desktop$ str2=Thisisa$str1
user@os-virtual-machine:~/Desktop$ str3=Thisis date
2023年 03月 30日 星期四 16:15:30 CST
user@os-virtual-machine:~/Desktop$ echo $str2
Thisisastring
```

str2中的\$str1是一个变量，会被Shell解释器解析；str3中的date前出现了空格，使用date会被解释成一个命令，执行后会返回当前日期和时间的字符串。

## 计算字符串长度

1、`${#string}`: 来计算字符串的长度, 其中string是字符串的名字。例如:

```
user@os-virtual-machine:~/Desktop$ str="https://cc.nankai.edu.cn/"
user@os-virtual-machine:~/Desktop$ echo ${#str}
25
```

2、`wc -L`命令: 该命令可以获取当前行的长度, 因此对于单独行的字符串可以用这个简单的方法获取; 另外可以用`wc -l`获取当前字符串内容的行数。例如:

```
user@os-virtual-machine:~/Desktop$ echo "https://cc.nankai.edu.cn/" | wc -L
25
user@os-virtual-machine:~/Desktop$ echo "https://cc.nankai.edu.cn/" | wc -l
1
```

3、`expr length string`命令: 来计算字符串的长度, 其中string是字符串的名字。例如:

```
user@os-virtual-machine:~/Desktop$ expr length "https://cc.nankai.edu.cn/"
25
```

## 字符串拼接

在 Shell编程中，字符串的拼接不需要使用任何运算符，将两个字符串并排放在一起就能实现拼接

```
user@os-virtual-machine:~/Desktop$ str1="os"
user@os-virtual-machine:~/Desktop$ str2="this is an important course"
user@os-virtual-machine:~/Desktop$ str3=$str1$str2
user@os-virtual-machine:~/Desktop$ str4="$str1 $str2"
user@os-virtual-machine:~/Desktop$ str5="$str1":"$str2"
user@os-virtual-machine:~/Desktop$ str6="$str1:$str2"
user@os-virtual-machine:~/Desktop$ str7="${str1}(operating system):${str2}"
user@os-virtual-machine:~/Desktop$ echo $str3
osthis is an important course
user@os-virtual-machine:~/Desktop$ echo $str4
os this is an important course
user@os-virtual-machine:~/Desktop$ echo $str5
os:this is an important course
user@os-virtual-machine:~/Desktop$ echo $str6
os:this is an important course
user@os-virtual-machine:~/Desktop$ echo $str7
os(operating system):this is an important course
```

像str3这样没有被双引号包围的，则中间不能有空格；像str4这样有双引号包围的，则中间可以有空格；变量与变量之间也可以出现别的字符串，像str5、str6和str7。

## 字符串截取

从指定位置开始截取字符串需要两个参数：除了指定**起始位置**，还需要**截取长度**，才能最终确定要截取的字符串。

Shell 同时支持两种计数方式，既可以从字符串左边开始计数，也可以从字符串右边开始计数。

1、**从字符串左边开始计数**：`${string: start :length}`，其中，string 是要截取的字符串，start 是起始位置，从左边开始数，并且从 0 开始计数，length 是要截取的长度（省略的话表示直到字符串的末尾）

```
user@os-virtual-machine:~/Desktop$ str1="https://cc.nankai.edu.cn/"
user@os-virtual-machine:~/Desktop$ echo ${str1:11:6}
nankai
```

2、**从字符串右边开始计数**：`${string: 0-start :length}`，同左边截取相比，右边截取仅仅多了0-，这是固定的写法，专门用来表示从字符串右边开始计数。

```
user@os-virtual-machine:~/Desktop$ str1="https://cc.nankai.edu.cn/"
user@os-virtual-machine:~/Desktop$ echo ${str1:0-14:6}
nankai
```

①从左边开始计数时，起始数字是 0；从右边开始计数时，起始数字是 1。计数方向不同，起始数字也不同。

②不管从哪边开始计数，**截取方向都是从左到右**。



## 正则表达式

正则表达式(Regular Expression)是一种文本模式，包括普通字符（例如，a 到 z 之间的字母）和特殊字符（称为"元字符"），使用单个字符串来描述、匹配一系列匹配某个句法规则的字符串，许多程序设计语言都支持利用正则表达式进行字符串操作。

特殊字符	描述
\$	匹配输入字符串的结尾位置。如果设置了 RegExp 对象的 Multiline 属性，则 \$ 也匹配 '\n' 或 '\r'。要匹配 \$ 字符本身，请使用 \\$。
()	标记一个子表达式的开始和结束位置。子表达式可以获取供以后使用。要匹配这些字符，请使用 \ ( 和 \ )。
*	匹配前面的子表达式零次或多次。要匹配 * 字符，请使用 \*。
+	匹配前面的子表达式一次或多次。要匹配 + 字符，请使用 \+。
.	匹配除换行符 \n 之外的任何单字符。要匹配 .，请使用 \.。
[	标记一个中括号表达式的开始。要匹配 [，请使用 \[。
?	匹配前面的子表达式零次或一次，或指明一个非贪婪限定符。要匹配 ? 字符，请使用 \?。
\	将下一个字符标记为或特殊字符、或原义字符、或向后引用、或八进制转义符。例如， 'n' 匹配字符 'n'。'\n' 匹配换行符。序列 '\\' 匹配 "\"，而 \"(' 则匹配 "("。
^	匹配输入字符串的开始位置，除非在方括号表达式中使用，当该符号在方括号表达式中使用，表示不接受该方括号表达式中的字符集合。要匹配 ^ 字符本身，请使用 \^。
{	标记限定符表达式的开始。要匹配 {，请使用 \{。
	指明两项之间的一个选择。要匹配  ，请使用 \ 。

## 正则表达式

限定符用来指定正则表达式的一个给定组件必须要出现多少次才能满足匹配。

有 \* 或 + 或 ? 或 {n} 或 {n,} 或 {n,m} 共6种。

限定符	描述
*	匹配前面的子表达式零次或多次。例如，zo* 能匹配 "z" 以及 "zoo"。* 等价于 {0,}。
+	匹配前面的子表达式一次或多次。例如，zo+ 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。 + 等价于 {1,}。
?	匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do"、"does" 中的 "does"、 "doxy" 中的 "do"。? 等价于 {0,1}。
{n}	n 是一个非负整数。匹配确定的 n 次。例如，o{2} 不能匹配 "Bob" 中的 o，但是能匹 配 "food" 中的两个 o。
{n,}	n 是一个非负整数。至少匹配n 次。例如，o{2,} 不能匹配 "Bob" 中的 o，但能匹 配 "fooooood" 中的所有 o。o{1,} 等价于 o+。o{0,} 则等价于 o*。
{n,m}	m 和 n 均为非负整数，其中 n <= m。最少匹配 n 次且最多匹配 m 次。例如，o{1,3} 将匹 配 "fooooood" 中的前三个 o。o{0,1} 等价于 o?。请注意在逗号和两个数之间不能有空格。

## 正则表达式

运算符的优先级：正则表达式从左到右进行计算，并遵循优先级顺序，下表从最高到最低说明了各种正则表达式运算符的优先级顺序：

运算符	描述
\	转义符
(), (?,:), (?=), []	圆括号和方括号
*, +, ?, {n}, {n,}, {n,m}	限定符
^, \$, \任何元字符、任何字符	定位点和序列（即位置和顺序）
	字符具有高于替换运算符的优先级，使得"m food"匹配"m"或"food"。若要匹配"mood"或"food"，请使用括号创建子表达式，从而产生"(m f)ood"。



## 正则表达式

如何利用正则表达式在大量文本文件中查找回复文件呢？

```
Open 0001.ea7e79d3153e7469e7a9c3e0af6a357e
~/Desktop/ham

1 From exmh-workers-admin@redhat.com Thu Aug 22 12:36:23 2002
2 Return-Path: <exmh-workers-admin@example.com>
3 Delivered-To: zzzz@localhost.netnoteinc.com
4 Received: from localhost (localhost [127.0.0.1])
5     by phobos.labs.netnoteinc.com (Postfix) with ESMTTP id D03E543C36
6     for <zzzz@localhost>; Thu, 22 Aug 2002 07:36:16 -0400 (EDT)
7 Received: from phobos [127.0.0.1]
8     by localhost with IMAP (fetchmail-5.9.0)
9     for zzzz@localhost (single-drop); Thu, 22 Aug 2002 12:36:16 +0100 (IST)
10 Received: from listman.example.com (listman.example.com [66.187.233.211]) by
11     dogma.slashnull.org (8.11.6/8.11.6) with ESMTTP id g7MBYrZ04811 for
12     <zzzz-exmh@example.com>; Thu, 22 Aug 2002 12:34:53 +0100
13 Received: from listman.example.com (localhost.localdomain [127.0.0.1]) by
14     listman.redhat.com (Postfix) with ESMTTP id 8386540858; Thu, 22 Aug 2002
15     07:35:02 -0400 (EDT)
16 Delivered-To: exmh-workers@listman.example.com
17 Received: from int-mx1.corp.example.com (int-mx1.corp.example.com
18     [172.16.52.254]) by listman.redhat.com (Postfix) with ESMTTP id 10CF8406D7
19     for <exmh-workers@listman.redhat.com>; Thu, 22 Aug 2002 07:34:10 -0400
20     (EDT)
21 Received: (from mail@localhost) by int-mx1.corp.example.com (8.11.6/8.11.6)
22     id g7MBY7g11259 for exmh-workers@listman.redhat.com; Thu, 22 Aug 2002
23     07:34:07 -0400
24 Received: from mx1.example.com (mx1.example.com [172.16.48.31]) by
25     int-mx1.corp.redhat.com (8.11.6/8.11.6) with SMTP id g7MBY7Y11255 for
26     <exmh-workers@redhat.com>; Thu, 22 Aug 2002 07:34:07 -0400
27 Received: from ratree.psu.ac.th ([202.28.97.6]) by mx1.example.com
28     (8.11.6/8.11.6) with SMTP id g7MBIhL25223 for <exmh-workers@redhat.com>;
29     Thu, 22 Aug 2002 07:18:55 -0400
30 Received: from delta.cs.mu.OZ.AU (delta.coe.psu.ac.th [172.30.0.98]) by
31     ratree.psu.ac.th (8.11.6/8.11.6) with ESMTTP id g7MBWl29762;
32     Thu, 22 Aug 2002 18:32:40 +0700 (ICT)
33 Received: from munnari.OZ.AU (localhost [127.0.0.1]) by delta.cs.mu.OZ.AU
34     (8.11.6/8.11.6) with ESMTTP id g7MBQPW13260; Thu, 22 Aug 2002 18:26:25
35     +0700 (ICT)
36 From: Robert Elz <kre@munnnari.OZ.AU>
37 To: Chris Garrigues <cwg-dated-1030377287.06fa6d@DeepEddy.Com>
38 Cc: exmh-workers@example.com
39 Subject: Re: New Sequences Window
40 In-Reply-To: <1029945287.4797.TMDA@deepeddy.vircio.com>
41 References: <1029945287.4797.TMDA@deepeddy.vircio.com>
42     <1029882468.3116.TMDA@deepeddy.vircio.com> <9627.1029933001@munnnari.OZ.AU>
43     <1029943066.26919.TMDA@deepeddy.vircio.com>
44     <1029944441.398.TMDA@deepeddy.vircio.com>
```

在“Subject”一行查找“Re”或“RE”即可： **^Subject: \*R[eE]**

# Shell文件读写

---

1、**读取文件中的行**：可以使用read命令逐行读取文件内容，read 命令从标准输入中读取一行，并把输入行的每个字段的值指定给 shell 变量，用 IFS（内部字段分隔符）变量中的字符作为分隔符。每次调用read，VariableName（变量名）参数中会存储一个得到的字段值，以此类推，直到最后一个字段；当文件没有可读的行时，read命令将以非零状态退出，例如：

```
while read line
do
#对每行执行操作
    echo ${line}
done < filename
```

这个命令会将文件中的每一行依次读入变量`line`中，然后对每行执行该操作。

2、**写入文件**：可以使用`echo`命令将内容写入文件中，例如：

```
echo “这是一行文本” > filename
```

这个命令会将指定的文本写入文件中。如果文件已经存在，则会覆盖文件中原有的内容。如果想要将内容追加到文件的末尾，可以使用`>>`运算符，例如：

```
echo “这是另一行文本” >> filename
```

这个命令会将指定的文本追加到文件的末尾。

3、**截断文件**：可以使用`truncate`命令截断文件，例如：

```
truncate -s 0 filename
```

这个命令会将文件的大小截为0，即清空文件中的内容。

4、**覆盖文件**：可以使用`cp`命令将一个文件的内容复制到另一个文件中，例如：

```
cp source_file dest_file
```

这个命令会将source\_file文件中的内容复制到dest\_file文件中。如果dest\_file文件已经存在，则会覆盖原有的内容。

## 5、grep命令：

(1) 查找包含指定字符串的行：

grep 'test' filename

这个命令将输出文件中包含“test”的行的内容。例如：

```
user@os-virtual-machine:~/Desktop$ cat test.txt
test
text
user@os-virtual-machine:~/Desktop$ grep 'test' test.txt
test
```

(2) 查找不包含指定字符串的行：

grep -v 'test' filename

这个命令将输出文件中不包含“test”的行的内容。例如：

```
user@os-virtual-machine:~/Desktop$ cat test.txt
test
text
user@os-virtual-machine:~/Desktop$ grep -v 'test' test.txt
text
```

(3) 查找包含指定字符串的文件：

grep -l 'test' \*

这个命令将输出所有包含“test”的文件名。

## 6、awk命令：

### (1) 输出指定列：

`awk '{print $n}' filename`

这个命令将输出文件中的第n列。例如：

```
user@os-virtual-machine:~/Desktop$ cat test.txt
test  hhh
text  ttt
user@os-virtual-machine:~/Desktop$ awk '{print $1}' test.txt
test
text
user@os-virtual-machine:~/Desktop$ awk '{print $2}' test.txt
hhh
ttt
```

在这个例子中，分别输出了“test.txt”的第一列和第二列。

### (2) 根据指定条件过滤行：

`awk '/test/ {print}' filename`

这个命令将输出文件中包含“test”的行。例如：

```
user@os-virtual-machine:~/Desktop$ cat test.txt
test  hhh
text  ttt
user@os-virtual-machine:~/Desktop$ awk '/test/ {print}' test.txt
test  hhh
```

在这个例子中，输出了包含“test”的行。

## 6、awk命令:

(3) 将指定行输出到新文件:

```
awk 'NR==n' A >> B
```

这个命令将文件A中的第n行输出新的文件B中。例如:

```
user@os-virtual-machine:~/Desktop$ cat test.txt
test
text
user@os-virtual-machine:~/Desktop$ awk 'NR==1' test.txt >> new.txt
user@os-virtual-machine:~/Desktop$ cat new.txt
test
user@os-virtual-machine:~/Desktop$
```

在这个例子中, “test.txt”的第一行被输出到了“new.txt”中。

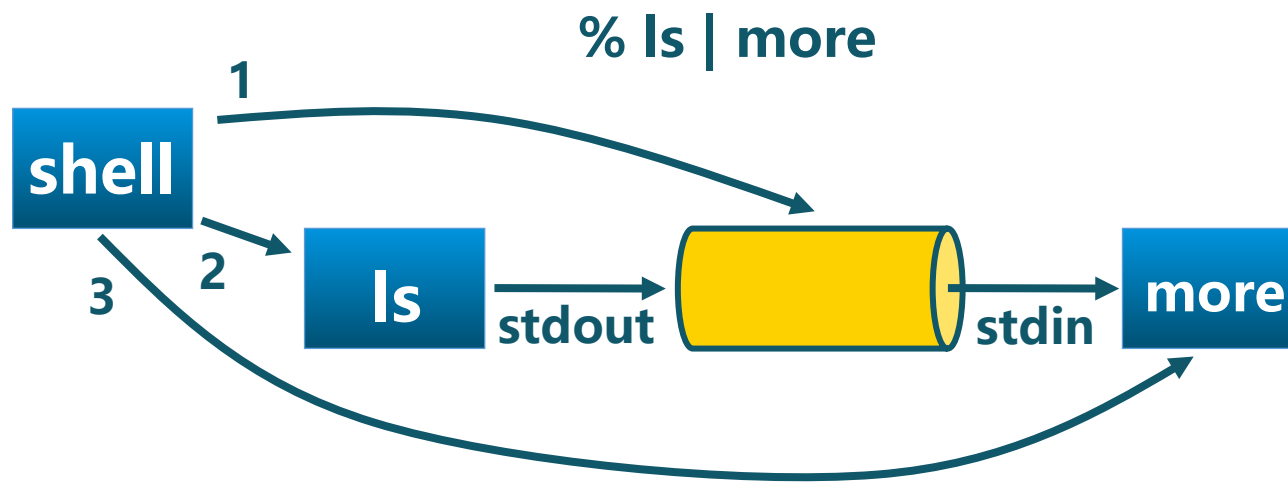




# Shell中的信息传递

---

## 管道原理示意



shell

创建管道

为ls创建一个进程, 设置 stdout为 管道写端

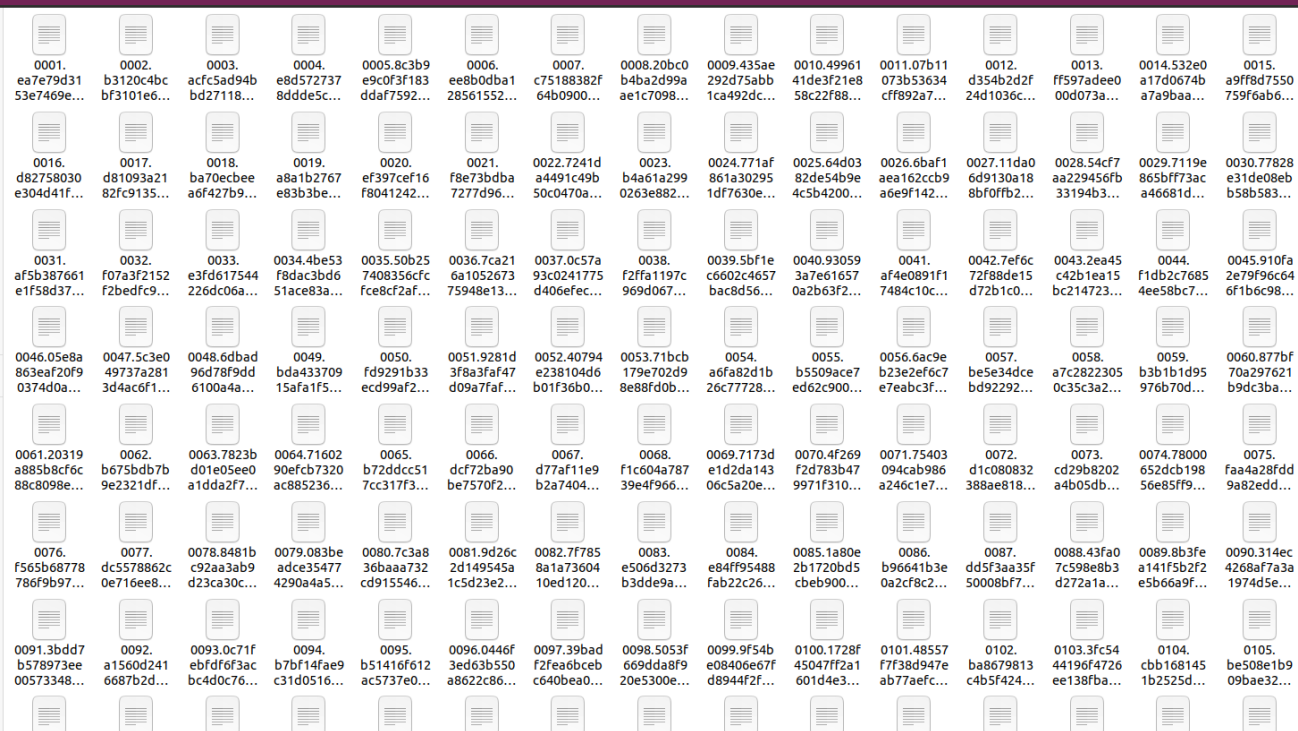
为more 创建一个进程, 设置 stdin 为管道读端



# Shell实战：批量文本文件内容过滤

---

# Shell实战：批量文本文件内容过滤



```
1 From exmh-workers-admin@redhat.com Thu Aug 22 12:36:23 2002
2 From exmh-users-admin@redhat.com Thu Aug 22 14:44:07 2002
3 From Stewart.Smith@ee.ed.ac.uk Thu Aug 22 14:44:26 2002
4 From martin@srv0.ems.ed.ac.uk Thu Aug 22 14:54:39 2002
5 From Stewart.Smith@ee.ed.ac.uk Thu Aug 22 15:05:07 2002
6 From exmh-workers-admin@redhat.com Thu Aug 22 15:15:12 2002
7 From exmh-workers-admin@redhat.com Fri Aug 23 11:06:58 2002
8 From spamassassin-devel-admin@lists.sourceforge.net Thu Aug 22 16:27:25 2002
9 From exmh-workers-admin@redhat.com Thu Aug 22 16:37:36 2002
10 From ilug-admin@linux.ie Thu Aug 22 17:19:31 2002
11 From timc@2ubh.com Thu Aug 22 17:31:00 2002
12 From ilug-admin@linux.ie Thu Aug 22 17:45:53 2002
13 From ilug-admin@linux.ie Thu Aug 22 17:45:54 2002
14 From exmh-workers-admin@redhat.com Thu Aug 22 18:17:16 2002
15 From lejones@ucla.edu Thu Aug 22 18:29:58 2002
16 From exmh-workers-admin@redhat.com Thu Aug 22 18:29:40 2002
17 From ilug-admin@linux.ie Fri Aug 23 11:07:47 2002
18 From fork-admin@xent.com Fri Aug 23 11:08:20 2002
19 From exmh-workers-admin@redhat.com Fri Aug 23 11:04:05 2002
20 From craig@deersoft.com Fri Aug 23 11:03:54 2002
21 From exmh-users-admin@redhat.com Fri Aug 23 11:04:05 2002
22 From exmh-users-admin@redhat.com Fri Aug 23 11:04:17 2002
23 From exmh-users-admin@redhat.com Fri Aug 23 11:04:30 2002
24 From fork-admin@xent.com Fri Aug 23 11:08:25 2002
25 From fork-admin@xent.com Fri Aug 23 11:08:26 2002
26 From fork-admin@xent.com Fri Aug 23 11:08:30 2002
27 From ilug-admin@linux.ie Fri Aug 23 11:07:51 2002
28 From exmh-workers-admin@redhat.com Fri Aug 23 11:04:57 2002
29 From hauns_froehlingsdorf@infinetivity.com Fri Aug 23 11:05:35 2002
30 From exmh-workers-admin@redhat.com Fri Aug 23 11:06:00 2002
31 From neugens@libero.it Fri Aug 23 11:06:09 2002
32 From exmh-workers-admin@redhat.com Fri Aug 23 11:06:06 2002
33 From ilug-admin@linux.ie Fri Aug 23 11:07:57 2002
34 From fork-admin@xent.com Fri Aug 23 11:08:44 2002
35 From ilug-admin@linux.ie Fri Aug 23 11:07:57 2002
36 From iiu-admin@taint.org Fri Aug 23 11:06:32 2002
37 From fork-admin@xent.com Fri Aug 23 11:08:46 2002
38 From fork-admin@xent.com Fri Aug 23 11:08:50 2002
39 From fork-admin@xent.com Fri Aug 23 11:08:50 2002
40 From fork-admin@xent.com Fri Aug 23 11:08:55 2002
41 From fork-admin@xent.com Fri Aug 23 11:08:56 2002
42 From exmh-workers-admin@redhat.com Fri Aug 23 11:07:16 2002
43 From ilug-admin@linux.ie Fri Aug 23 11:08:03 2002
44 From fork-admin@xent.com Fri Aug 23 11:09:00 2002
```

从批量的文本文件中找出回复文件，并将发件人、日期等信息  
输出在新的文件中

如何利用shell实现呢？以下是可能会用到的知识点：

- ①循环：用于遍历文件夹；
- ②正则表达式：用于匹配规定的特殊格式；
- ③文件的读写：用于从原文件中读出

## 其中一种实现方式：

```
1  #!/bin/bash
2
3  dir=$(dirname "$0")/input
4
5  unzip -q "$dir/ham.zip" -d "$dir" # 解压 ham.zip
6
7  cat /dev/null > ./output/new.txt
8  for i in "$dir/ham"/*
9  do
10     filename=$i
11     # 查找主题为回复的邮件并打印第一行内容，即发送者和日期
12     # 如果第一行的Subject 包含 Re 或 RE
13     if [ $(grep -c "^Subject: *R[eE]" "$filename") -ne '0' ];then
14         # 打印第一行内容
15         awk 'NR==1' "$filename" >> ./output/new.txt
16     fi
17 done
```



# Linux定时任务

---



## 什么是定时任务？

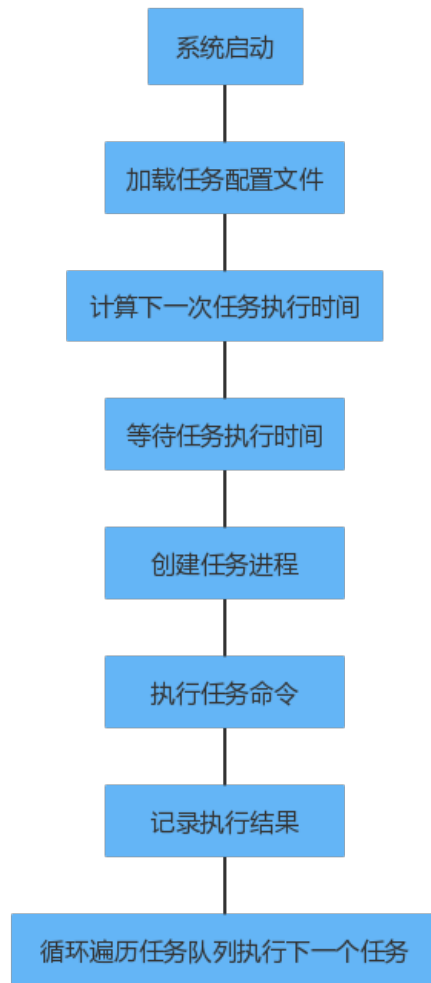
定时任务的原理是基于计算机系统中的**定时器**和**任务调度器**。在Linux系统中，定时任务通常由Cron和类似的工具来实现。这些工具利用操作系统提供的定时器（Timer）和任务调度器（Scheduler），在**指定的时间点自动执行**预定的命令或脚本。

实现定时任务的工具有以下几种：

- 1、**Cron**：是Linux下非常常见的一种任务调度器，它可以让用户在指定的时间间隔内自动执行命令或脚本。用户可以通过命令行编辑Cron的配置文件，设置定时任务的执行时间和要执行的命令。Cron会在指定的时间间隔内检查任务，如果任务需要执行，则会自动执行命令或脚本。
- 2、**Systemd timer**：是Linux下比较新的系统管理工具，它提供了一种称为systemd timer的任务调度器。使用Systemd服务管理器来管理任务。用户可以通过systemctl命令来创建和管理Systemd timer任务。
- 3、**Anacron**：是一个在Linux系统中运行的后台进程，它可以帮助用户在系统闲置时执行任务。用户可以通过编辑Anacron的配置文件来设置定时任务的执行时间和要执行的命令。
- 4、**At**：是一种基于时间的任务调度器，它允许用户在指定的时间点执行命令或脚本。用户可以使用at命令来创建和管理定时任务。At会在指定的时间点自动执行任务，并将任务的输出发送给用户。

## Cron的原理

Cron是Linux系统下一个常用的定时任务工具，它主要是基于操作系统的定时器和任务调度器实现的。

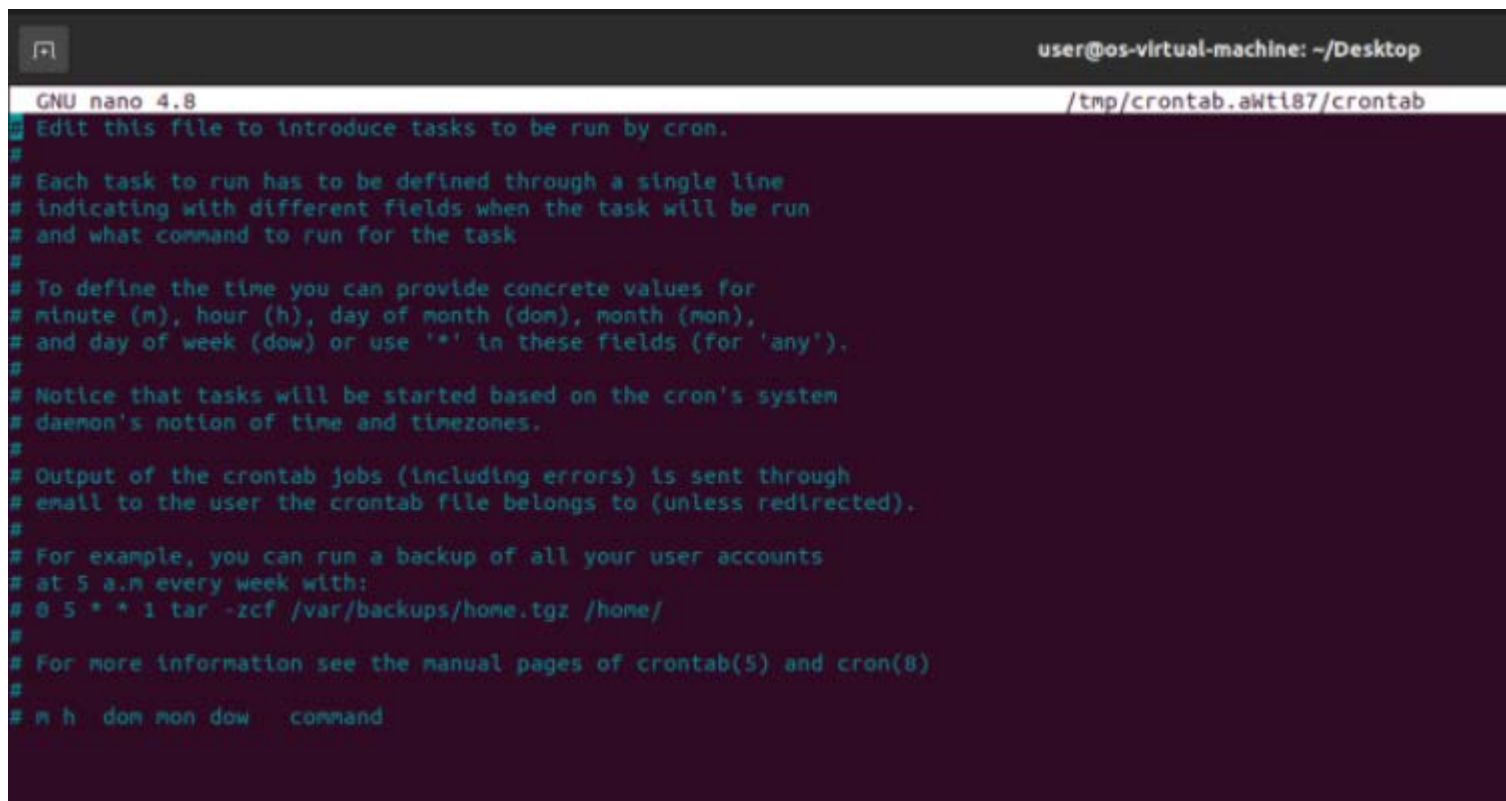


术语	含义
定时任务	“任务”是指一段启动并运行的程序。cron 可以按照约定的时间计划运行各种任务，这样的任务通常叫做“定时任务”。
cron	这是安装在系统上的实际执行定时任务的 守护进程。
crontab	这是一个文件，用于定义定时任务。一个 crontab 文件可以通过表格形式（每一行就是一个定时任务）定义多个定时任务。

## Cron的创建

假如此处我们需要创建一个定时任务，每分钟创建一次~/test 文件

1、首先运行`crontab -e`，这会打开一个文件，即`crontab`，用于编辑定时任务：



```
GNU nano 4.8 /tmp/crontab.awti87/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
```

其中，所有以#开头的注释都是在指导你如何使用cron。

## Cron的创建

定时任务都是以“分钟 小时 天 月 周 命令”的形式呈现：

```
* * * * *
```

```
- - - - -
```

```
| | | | |  
| | | | +----- 星期中星期几 (0 - 6) (星期天 为0)
```

```
| | | +----- 月份 (1 - 12)
```

```
| | +----- 一个月中的第几天 (1 - 31)
```

```
| +----- 小时 (0 - 23)
```

```
+----- 分钟 (0 - 59)
```

所以每分钟创建一次~/test 文件的语句应该为：

```
* * * * * touch ~/test
```

## Cron的创建

### 2、将该命令写进crontab中

```
GNU nano 4.8
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
* * * * * touch ~/test
```

### 3、下一分钟就可以在home目录下发现test文件

```
user@os-virtual-machine:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  test  Videos
user@os-virtual-machine:~$
```

### 4、可以使用crontab -r删除crontab文件中所有的定时任务

## Cron的应用

回到最开始的问题，若要每天晚上八点进行批量文本过滤，可以使用如下语句：

```
0 20 * * * /path/to/filter.sh
```

在filter.sh中，可以用grep或其他命令来查找过滤符合条件的文本。



# Linux服务管理扩展阅读

---

1、Systemd 是现代 Linux 发行版中广泛使用的服务管理器。你可以访问 Systemd 的官方文档，其中包含了有关单元文件（unit files）、服务管理、日志记录等方面的详细信息。可参考：

<https://systemd.io/>

2、SysVinit 是传统的 Linux 系统初始化和服务管理方案。尽管它在现代 Linux 发行版中逐渐被 Systemd 取代，但仍然有一些系统在使用。可以查阅 SysVinit 的官方文档以了解相关信息。可参考：

<https://wiki.debian.org/LSBInitScripts>

3、不同的 Linux 发行版通常都有自己的文档和指南，这些文档提供了针对该发行版的特定服务管理信息。可以访问你使用的 Linux 发行版的官方网站，查找与服务管理相关的文档部分。



# 本章作业

---

# 程序设计

利用脚本编程实现一个文件夹下的所有邮件文件分类。

1. 以Subject为分类条件，每个Subject为一个文件夹
2. 如果Subject中存在无法用做文件名的字符，以下划线代替
3. 所有回复该主题的文件都放在该文件夹中
4. 文件按照邮件中标出的发送的日期时间对文件进行重命名

## 观察并完成实验报告

1. 观察被处理的文件大小对系统可用内存的影响
2. 观察处理文件的总体数量对系统可用内存的影响



感谢阅读

---