# Awk

## Awk

<!-- @import "[TOC]" {cmd="toc" depthFrom=1 depthTo=6 orderedList=false} -->

<!-- code\_chunk\_output -->

\* [linux awk命令详解](#linux-awk命令详解)

    \* [简介](#简介)

    \* [使用方法](#使用方法)

    \* [调用awk](#调用awk)

    \* [入门实例](#入门实例)

    \* [awk内置变量](#awk内置变量)

    \* [print和printf](#print和printf)

    \* [awk编程](#awk编程)

        \* [变量和赋值](#变量和赋值)

        \* [条件语句](#条件语句)

        \* [循环语句](#循环语句)

        \* [数组](#数组)

<!-- /code\_chunk\_output -->

---

**# linux awk命令详解**

\* [linux awk命令详解 - ggjucheng - 博客园 ](http://www.cnblogs.com/ggjucheng/archive/2013/01/13/2858470.html)

**## 简介**

awk是一个强大的文本分析工具，相对于grep的查找，sed的编辑，awk在其对数据分析并生成报告时，显得尤为强大。简单来说awk就是把文件逐行的读入，以空格为默认分隔符将每行切片，切开的部分再进行各种分析处理。

awk有3个不同版本: awk、nawk和gawk，未作特别说明，一般指gawk，gawk 是 AWK 的 GNU 版本。

awk其名称得自于它的创始人 Alfred Aho 、Peter Weinberger 和 Brian Kernighan 姓氏的首个字母。实际上 AWK 的确拥有自己的语言： AWK 程序设计语言 ， 三位创建者已将它正式定义为“样式扫描和处理语言”。它允许您创建简短的程序，这些程序读取输入文件、为数据排序、处理数据、对输入执行计算以及生成报表，还有无数其他的功能。

**## 使用方法**

awk '{pattern + action}' {filenames}

尽管操作可能会很复杂，但语法总是这样，其中 pattern 表示 AWK 在数据中查找的内容，而 action 是在找到匹配内容时所执行的一系列命令。花括号（{}）不需要在程序中始终出现，但它们用于根据特定的模式对一系列指令进行分组。 pattern就是要表示的正则表达式，用斜杠括起来。

awk语言的最基本功能是在文件或者字符串中基于指定规则浏览和抽取信息，awk抽取信息后，才能进行其他文本操作。完整的awk脚本通常用来格式化文本文件中的信息。

通常，awk是以文件的一行为处理单位的。awk每接收文件的一行，然后执行相应的命令，来处理文本。

 本章重点介绍命令行方式。

**## 入门实例**

假设last -n 5的输出如下

```sh

[root@www ~]# last -n 5 <==仅取出前五行

root     pts/1   192.168.1.100  Tue Feb 10 11:21   still logged in

root     pts/1   192.168.1.100  Tue Feb 10 00:46 - 02:28  (01:41)

root     pts/1   192.168.1.100  Mon Feb  9 11:41 - 18:30  (06:48)

dmtsai   pts/1   192.168.1.100  Mon Feb  9 11:41 - 11:41  (00:00)

root     tty1                   Fri Sep  5 14:09 - 14:10  (00:01)

如果只是显示最近登录的5个帐号

#last -n 5 | awk  '{print $1}'

root

root

root

dmtsai

root

```

awk工作流程是这样的：读入有'\n'换行符分割的一条记录，然后将记录按指定的域分隔符划分域，填充域，$0则表示所有域,$1表示第一个域,$n表示第n个域。默认域分隔符是"空白键" 或 "[tab]键",所以$1表示登录用户，$3表示登录用户ip,以此类推。

如果只是显示/etc/passwd的账户

```sh

#cat /etc/passwd |awk  -F ':'  '{print $1}'

root

daemon

bin

sys

```

这种是awk+action的示例，每行都会执行action{print $1}。

-F指定域分隔符为':'。

如果只是显示/etc/passwd的账户和账户对应的shell,而账户与shell之间以tab键分割

```sh

#cat /etc/passwd |awk  -F ':'  '{print $1"\t"$7}'

root    /bin/bash

daemon  /bin/sh

bin     /bin/sh

sys     /bin/sh

```

如果只是显示/etc/passwd的账户和账户对应的shell,而账户与shell之间以逗号分割,而且在所有行添加列名name,shell,在最后一行添加"blue,/bin/nosh"。

```sh

cat /etc/passwd |awk  -F ':'  'BEGIN {print "name,shell"}  {print $1","$7} END {print "blue,/bin/nosh"}'

name,shell

root,/bin/bash

daemon,/bin/sh

bin,/bin/sh

sys,/bin/sh

....

blue,/bin/nosh

```

awk工作流程是这样的：先执行BEGING，然后读取文件，读入有/n换行符分割的一条记录，然后将记录按指定的域分隔符划分域，填充域，\$0则表示所有域,\$1表示第一个域,$n表示第n个域,随后开始执行模式所对应的动作action。接着开始读入第二条记录······直到所有的记录都读完，最后执行END操作。

搜索/etc/passwd有root关键字的所有行

```sh

#awk -F: '/root/' /etc/passwd

root:x:0:0:root:/root:/bin/bash

```

这种是pattern的使用示例，匹配了pattern(这里是root)的行才会执行action(没有指定action，默认输出每行的内容)。

搜索支持正则，例如找root开头的: awk -F: '/^root/' /etc/passwd

搜索/etc/passwd有root关键字的所有行，并显示对应的shell

```sh

# awk -F: '/root/{print $7}' /etc/passwd

/bin/bash

 这里指定了action{print $7}

```

> docker images | awk 'BEGIN{FS=" "}{printf("%s ",$3)}'

**## print和printf**

awk中同时提供了print和printf两种打印输出的函数。

其中print函数的参数可以是变量、数值或者字符串。字符串必须用双引号引用，参数用逗号分隔。如果没有逗号，参数就串联在一起而无法区分。这里，逗号的作用与输出文件的分隔符的作用是一样的，只是后者是空格而已。

printf函数，其用法和c语言中printf基本相似,可以格式化字符串,输出复杂时，printf更加好用，代码更易懂。

awk编程的内容极多，这里只罗列简单常用的用法，更多请参考 http://www.gnu.org/software/gawk/manual/gawk.html

分类: Linux/Unix

## 1.三种方式调用awk

**## 调用awk**

有三种方式调用awk

1.命令行方式

awk [-F  field-separator]  'commands'  input-file(s)

其中，commands 是真正awk命令，[-F域分隔符]是可选的。 input-file(s) 是待处理的文件。

在awk中，文件的每一行中，由域分隔符分开的每一项称为一个域。通常，在不指名-F域分隔符的情况下，默认的域分隔符是空格。

2.shell脚本方式

将所有的awk命令插入一个文件，并使awk程序可执行，然后awk命令解释器作为脚本的首行，一遍通过键入脚本名称来调用。

相当于shell脚本首行的：#!/bin/sh

可以换成：#!/bin/awk

3.将所有的awk命令插入一个单独文件，然后调用：

awk -f awk-script-file input-file(s)

其中，-f选项加载awk-script-file中的awk脚本，input-file(s)跟上面的是一样的。

**## 参考**

1. http://www.cnblogs.com/ggjucheng/archive/2013/01/13/2858470.html

## 2.awk内置变量

**## awk内置变量**

awk有许多内置变量用来设置环境信息，这些变量可以被改变，下面给出了最常用的一些变量。

```sh

ARGC               命令行参数个数

ARGV               命令行参数排列

ENVIRON            支持队列中系统环境变量的使用

FILENAME           awk浏览的文件名

FNR                浏览文件的记录数

FS                 设置输入域分隔符，等价于命令行 -F选项

NF                 浏览记录的域的个数

NR                 已读的记录数

OFS                输出域分隔符

ORS                输出记录分隔符

RS                 控制记录分隔符

```

此外,$0变量是指整条记录。$1表示当前行的第一个域,$2表示当前行的第二个域,......以此类推。

统计/etc/passwd:文件名，每行的行号，每行的列数，对应的完整行内容:

```sh

#awk  -F ':'  '{print "filename:" FILENAME ",linenumber:" NR ",columns:" NF ",linecontent:"$0}' /etc/passwd

filename:/etc/passwd,linenumber:1,columns:7,linecontent:root:x:0:0:root:/root:/bin/bash

filename:/etc/passwd,linenumber:2,columns:7,linecontent:daemon:x:1:1:daemon:/usr/sbin:/bin/sh

filename:/etc/passwd,linenumber:3,columns:7,linecontent:bin:x:2:2:bin:/bin:/bin/sh

filename:/etc/passwd,linenumber:4,columns:7,linecontent:sys:x:3:3:sys:/dev:/bin/sh

```

使用printf替代print,可以让代码更加简洁，易读

awk  -F ':'  '{printf("filename:%10s,linenumber:%s,columns:%s,linecontent:%s\n",FILENAME,NR,NF,$0)}' /etc/passwd

**## 参考**

1. http://www.cnblogs.com/ggjucheng/archive/2013/01/13/2858470.html

## 3.awk编程

**## awk编程**

**### 变量和赋值**

除了awk的内置变量，awk还可以自定义变量。

下面统计/etc/passwd的账户人数

awk '{count++;print $0;} END{print "user count is ", count}' /etc/passwd

root:x:0:0:root:/root:/bin/bash

......

user count is  40

count是自定义变量。之前的action{}里都是只有一个print,其实print只是一个语句，而action{}可以有多个语句，以;号隔开。

这里没有初始化count，虽然默认是0，但是妥当的做法还是初始化为0:

awk 'BEGIN {count=0;print "[start]user count is ", count} {count=count+1;print $0;} END{print "[end]user count is ", count}' /etc/passwd

[start]user count is  0

root:x:0:0:root:/root:/bin/bash

...

[end]user count is  40

统计某个文件夹下的文件占用的字节数

ls -l |awk 'BEGIN {size=0;} {size=size+$5;} END{print "[end]size is ", size}'

[end]size is  8657198

如果以M为单位显示:

ls -l |awk 'BEGIN {size=0;} {size=size+$5;} END{print "[end]size is ", size/1024/1024,"M"}'

[end]size is  8.25889 M

注意，统计不包括文件夹的子目录。

**### 条件语句**

 awk中的条件语句是从C语言中借鉴来的，见如下声明方式：

```sh

if (expression) {

    statement;

    statement;

    ... ...

}

if (expression) {

    statement;

} else {

    statement2;

}

if (expression) {

    statement1;

} else if (expression1) {

    statement2;

} else {

    statement3;

}

```

统计某个文件夹下的文件占用的字节数,过滤4096大小的文件(一般都是文件夹):

ls -l |awk 'BEGIN {size=0;print "[start]size is ", size} {if($5!=4096){size=size+$5;}} END{print "[end]size is ", size/1024/1024,"M"}'

[end]size is  8.22339 M

**### 循环语句**

awk中的循环语句同样借鉴于C语言，支持while、do/while、for、break、continue，这些关键字的语义和C语言中的语义完全相同。

**### 数组**

因为awk中数组的下标可以是数字和字母，数组的下标通常被称为关键字(key)。值和关键字都存储在内部的一张针对key/value应用hash的表格里。由于hash不是顺序存储，因此在显示数组内容时会发现，它们并不是按照你预料的顺序显示出来的。数组和变量一样，都是在使用时自动创建的，awk也同样会自动判断其存储的是数字还是字符串。一般而言，awk中的数组用来从记录中收集信息，可以用于计算总和、统计单词以及跟踪模板被匹配的次数等等。

显示/etc/passwd的账户

```sh

awk -F ':' 'BEGIN {count=0;} {name[count] = $1;count++;}; END{for (i = 0; i < NR; i++) print i, name[i]}' /etc/passwd

0 root

1 daemon

2 bin

3 sys

4 sync

5 games

......

```

这里使用for循环遍历数组

**## 参考**

1. http://www.cnblogs.com/ggjucheng/archive/2013/01/13/2858470.html

## 4.内建数学函数

14.8.3. 内建数学函数

Table 4.

函数名称    返回值

atan2(x,y)  y,x范围内的余切

cos(x)  余弦函数

exp(x)  求幂

int(x)  取整

log(x)  自然对数

rand()  随机数

sin(x)  正弦

sqrt(x) 平方根

srand(x)    x是rand()函数的种子

int(x)  取整，过程没有舍入

rand()  产生一个大于等于0而小于1的随机数

```sh

### 磁盘使用量超过30

df -hP|awk 'NR>1 && int($5) > 30'

df -hP|awk '{print int($5)}'

```

**## 参考**

1. https://www.cnblogs.com/emanlee/p/3327576.html

2. https://www.cnblogs.com/mousean/p/6102105.html

## 5.awk指令不显示第一行最后一行

linux awk指令不显示第一行 - LiaoS http://www.liaozhiwei.com/p/1243

使用awk时，有时第一行为列名，需要过滤掉，从第二行开始显示。

如获取80端口使用情况

lsof -i:80 | awk '{print $2}'

这时显示的结果为

PID

1033

1650

1806

1806

需要去过滤掉第一行，指令为：

lsof -i:80 | awk 'NR!=1{print $2}'

**## 最后一行**

kubectl get no| awk 'END{print $1}'

**## 排除第一行和最后一行**

kubectl get no| sed '1d;$d'| awk '{print $1}'

## 6.shell怎么解析科学计数

https://zhidao.baidu.com/question/757128722935056724.html

Shell中科学计数法一般用awk进行处理，浮点类型一般可以用bc进行计算

例如将一个科学计数法表示的数字（不太大，可以不用科学计数法表示的。例如1e3，也就是1000，可以不用科学计数法）转为正常表示的数字：

cnbubble@~/test:$ echo "1e3" |awk '{print $0\*1}'

1000

awk中对$0进行乘以1计算（也可以加0或减0），强制将输入当成数值，不然它会认为是文本，原样输出

## Linux查看当前目录下修改日期为最新文件夹内的文件

Linux查看当前目录下修改日期为最新文件夹内的文件\_花家七童\_新浪博客

http://blog.sina.com.cn/s/blog\_50d43ad50101g3p2.html

示例：

root@alex:/tmp# ll

    drwxr-xr-x 2 sun sun 4096 08-08 00:57 20130807

    drwxr-xr-x 2 sun sun 4096 08-07 00:57 20130806

    drwxr-xr-x 2 sun sun 4096 08-06 00:57 20130805

root@alex:/tmp# cd 20130807

root@alex:/tmp# ls

    20130807\_add.tar.z

`ls -t |awk '{if(NR==1)print $1}'`

命令：

cd /tmp;ml=$(ls -t |awk '{if(NR==1)print $1}';);cd $ml;ls;

**## awk内置变量**

awk有许多内置变量用来设置环境信息，这些变量可以被改变，下面给出了最常用的一些变量。

```sh

ARGC               命令行参数个数

ARGV               命令行参数排列

ENVIRON            支持队列中系统环境变量的使用

FILENAME           awk浏览的文件名

FNR                浏览文件的记录数

FS                 设置输入域分隔符，等价于命令行 -F选项

NF                 浏览记录的域的个数

NR                 已读的记录数

OFS                输出域分隔符

ORS                输出记录分隔符

RS                 控制记录分隔符

```

# Crontab

## 5.如何查看crontab的日志记录

https://www.cnblogs.com/doseoer/p/5663187.html

在Unix和类Unix的操作系统之中，crontab命令常用于设置周期性被执行的指令，也可以理解为设置定时任务。

crontab中的定时任务有时候没有成功执行，什么原因呢？这时就需要去日志里去分析一下了，那该如何查看crontab的日志记录呢？

1. linux

看 /var/log/cron.log这个文件就可以，可以用tail -f /var/log/cron.log观察

tail -f /var/log/cron

2. unix

在 /var/spool/cron/tmp文件中，有croutXXX001864的tmp文件，tail 这些文件就可以看到正在执行的任务了。

3. mail任务

在 /var/spool/mail/root 文件中，有crontab执行日志的记录，用tail -f /var/spool/mail/root 即可查看最近的crontab执行情况。

有朋友问到关于linux的crontab不知道是否到底执行了没有，也算写过一些基本备份的shell脚本，结合自己的实际生产环境简单讲述下如何通过cron执行的日志来分析crontab是否正确执行。

例如服务器下oracle用户有如下的计划任务

[oracle@localhost6 ~]$ crontab -l

00 1 \* \* 0 /home/oracle/backup/hollyipcc.sh

00 1 1 \* \* /home/oracle/backup/hollyreport\_hollycrm.sh

关于系统的计划任务都会先在/var/log

[root@localhost ~]# cd /var/log/

[root@localhost log]# less cron

Sep 22 04:22:01 localhost crond[32556]: (root) CMD (run-parts /etc/cron.weekly)

Sep 22 04:22:01 localhost anacron[32560]: Updated timestamp for job `cron.weekly' to 2013-09-22

Sep 22 05:01:01 localhost crond[22768]: (root) CMD (run-parts /etc/cron.hourly)

Sep 22 06:01:01 localhost crond[25522]: (root) CMD (run-parts /etc/cron.hourly)

Sep 22 07:01:01 localhost crond[28255]: (root) CMD (run-parts /etc/cron.hourly)

Sep 22 08:01:01 localhost crond[30982]: (root) CMD (run-parts /etc/cron.hourly)

。。。

上面的/var/log/cron只会记录是否执行了某些计划的脚本，但是具体执行是否正确以及脚本执行过程中的一些信息则linux会每次都发邮件到该用户下。

如上述oracle的计划任务执行信息，linux会发邮件到/var/spool/mail下面

[root@localhost6 log]# cd /var/spool/mail/

[root@localhost6 mail]# less oracle

Date: Sun, 25 Aug 2013 01:00:01 +0800

Message-Id: <201308241700.r7OH01aG016679@localhost6.localdomain6>

From: root@localhost6.localdomain6 (Cron Daemon)

To: oracle@localhost6.localdomain6

Subject: Cron /home/oracle/backup/hollyipcc.sh

Content-Type: text/plain; charset=UTF-8

Auto-Submitted: auto-generated

X-Cron-Env:

X-Cron-Env:

X-Cron-Env:

X-Cron-Env:

X-Cron-Env:

backup hollyipcc

Export: Release 10.2.0.4.0 - Production on Sunday, 25 August, 2013 1:00:02

Copyright (c) 2003, 2007, Oracle. All rights reserved.

Connected to: Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - Production

With the Partitioning, OLAP, Data Mining and Real Application Testing options

Starting "SYSTEM"."SYS\_EXPORT\_SCHEMA\_01": system/*\*\*\*\*\*\**\*\* dumpfile=hollyipcc\_20130825.dmp logfile=hollyipcc\_20130825.log directory

=back schemas=hollyipcc parfile=/home/oracle/backup/parfile.par

Estimate in progress using BLOCKS method...

Processing object type SCHEMA\_EXPORT/TABLE/TABLE\_DATA

Total estimation using BLOCKS method: 5.932 GB

Processing object type SCHEMA\_EXPORT/TABLE/TABLE

Processing object type SCHEMA\_EXPORT/TABLE/GRANT/OWNER\_GRANT/OBJECT\_GRANT

Processing object type SCHEMA\_EXPORT/TABLE/INDEX/INDEX

Processing object type SCHEMA\_EXPORT/TABLE/CONSTRAINT/CONSTRAINT

Processing object type SCHEMA\_EXPORT/TABLE/INDEX/STATISTICS/INDEX\_STATISTICS

Processing object type SCHEMA\_EXPORT/TABLE/STATISTICS/TABLE\_STATISTICS

. . exported "HOLLYIPCC"."BASE\_CONTACTSTATE\_BAK":"P201203" 1.389 GB 15716014 rows

此时我们就能很好的判断crontab脚本是否执行，已经执行过程中是否正确以及一些错误的信息，希望这个可以帮助有些朋友很好的了解crontab的原理和诊断crontab的问题。

## cron.hourly

35.1.1. Configuring Cron Tasks

 https://www.centos.org/docs/5/html/5.2/Deployment\_Guide/s2-autotasks-cron-configuring.html

 The main configuration file for cron, /etc/crontab, contains the following lines:

```conf

SHELL=/bin/bash

PATH=/sbin:/bin:/usr/sbin:/usr/bin

MAILTO=root HOME=/

# run-parts

01 \* \* \* \* root run-parts /etc/cron.hourly

02 4 \* \* \* root run-parts /etc/cron.daily

22 4 \* \* 0 root run-parts /etc/cron.weekly

42 4 1 \* \* root run-parts /etc/cron.monthly

```

The first four lines are variables used to configure the environment in which the cron tasks are run. The SHELL variable tells the system which shell environment to use (in this example the bash shell), while the PATH variable defines the path used to execute commands. The output of the cron tasks are emailed to the username defined with the MAILTO variable. If the MAILTO variable is defined as an empty string (MAILTO=""), email is not sent. The HOME variable can be used to set the home directory to use when executing commands or scripts.

前四行是变量用于配置的cron任务的环境中运行。shell变量告诉系统要使用哪个shell环境（在这个例子中是shell），而PATH变量定义了用于执行命令的路径。这个cron任务的输出是通过电子邮件与邮件变量定义的用户名。如果邮寄地址的变量定义为空字符串（mailto =”），电子邮件是不是发。可以使用home变量设置主目录，以便在执行命令或脚本时使用该目录。

Each line in the /etc/crontab file represents a task and has the following format:

minute   hour   day   month   dayofweek   command

\* minute — any integer from 0 to 59

\* hour — any integer from 0 to 23

\* day — any integer from 1 to 31 (must be a valid day if a month is specified)

\* month — any integer from 1 to 12 (or the short name of the month such as jan or feb)

\* dayofweek — any integer from 0 to 7, where 0 or 7 represents Sunday (or the short name of the week such as sun or mon)

\* command — the command to execute (the command can either be a command such as ls /proc >> /tmp/proc or the command to execute a custom script)

As shown in the /etc/crontab file, the run-parts script executes the scripts in the /etc/cron.hourly/, /etc/cron.daily/, /etc/cron.weekly/, and /etc/cron.monthly/ directories on an hourly, daily, weekly, or monthly basis respectively. The files in these directories should be shell scripts.

If a cron task is required to be executed on a schedule other than hourly, daily, weekly, or monthly, it can be added to the /etc/cron.d/ directory. All files in this directory use the same syntax as /etc/crontab. Refer to Example 35.1, “Crontab Examples” for examples.

```conf

 # record the memory usage of the system every monday

# at 3:30AM in the file /tmp/meminfo

30 3 \* \* mon cat /proc/meminfo >> /tmp/meminfo

# run custom script the first day of every month at 4:10AM

10 4 1 \* \* /root/scripts/backup.sh

Example 35.1. Crontab Examples

```

## linux下用cron定时执行任务的方法

linux下用cron定时执行任务的方法\_Linux\_脚本之家

http://www.jb51.net/article/15008.htm

名称 : crontab

使用权限 : 所有使用者

使用方式 :

crontab file [-u user]-用指定的文件替代目前的crontab。

crontab-[-u user]-用标准输入替代目前的crontab.

crontab-1[user]-列出用户目前的crontab.

crontab-e[user]-编辑用户目前的crontab.

crontab-d[user]-删除用户目前的crontab.

crontab-c dir- 指定crontab的目录。

crontab文件的格式：M H D m d cmd.

基本格式 :

\* \* \* \* \* command

分 时 日 月 周 命令

M: 分钟（0-59）。每分钟用\*或者 \*/1表示

H：小时（0-23）。（0表示0点）

D：天（1-31）。

m: 月（1-12）。

d: 一星期内的天（0~6，0为星期天）。

cmd要运行的程序，程序被送入sh执行，这个shell只有`USER,HOME,SHELL`这三个环境变量

说明 :

crontab 是用来让使用者在固定时间或固定间隔执行程序之用，换句话说，也就是类似使用者的时程表。-u user 是指设定指定 user 的时程表，这个前提是你必须要有其权限(比如说是 root)才能够指定他人的时程表。如果不使用 -u user 的话，就是表示设定自己的时程表。

参数 :

crontab -e : 执行文字编辑器来设定时程表，内定的文字编辑器是 VI，如果你想用别的文字编辑器，则请先设定 VISUAL 环境变数来指定使用那个文字编辑器(比如说 setenv VISUAL joe)

crontab -r : 删除目前的时程表

crontab -l : 列出目前的时程表

crontab file [-u user]-用指定的文件替代目前的crontab。

时程表的格式如下 :

f1 f2 f3 f4 f5 program

其中 f1 是表示分钟，f2 表示小时，f3 表示一个月份中的第几日，f4 表示月份，f5 表示一个星期中的第几天。program 表示要执行的程序。

当 f1 为 \* 时表示每分钟都要执行 program，f2 为 \* 时表示每小时都要执行程序，其馀类推

当 f1 为 a-b 时表示从第 a 分钟到第 b 分钟这段时间内要执行，f2 为 a-b 时表示从第 a 到第 b 小时都要执行，其馀类推

当 f1 为 \*/n 时表示每 n 分钟个时间间隔执行一次，f2 为 \*/n 表示每 n 小时个时间间隔执行一次，其馀类推

当 f1 为 a, b, c,... 时表示第 a, b, c,... 分钟要执行，f2 为 a, b, c,... 时表示第 a, b, c...个小时要执行，其馀类推

使用者也可以将所有的设定先存放在档案 file 中，用 crontab file 的方式来设定时程表。

例子 :

```sh

#每天早上7点执行一次 /bin/ls :

0 7 \* \* \* /bin/ls

#在 12 月内, 每天的早上 6 点到 12 点中，每隔3个小时执行一次 /usr/bin/backup :

0 6-12/3 \* 12 \* /usr/bin/backup

#周一到周五每天下午 5:00 寄一封信给 alex@domain.name :

0 17 \* \* 1-5 mail -s "hi" alex@domain.name < /tmp/maildata

#每月每天的午夜 0 点 20 分, 2 点 20 分, 4 点 20 分....执行 echo "haha"

20 0-23/2 \* \* \* echo "haha"

```

注意 :

当程序在你所指定的时间执行后，系统会寄一封信给你，显示该程序执行的内容，若是你不希望收到这样的信，请在每一行空一格之后加上 > /dev/null 2>&1 即可

例子2 :

```sh

#每天早上6点10分

10 6 \* \* \* date

#每两个小时

0 \*/2 \* \* \* date

#晚上11点到早上8点之间每两个小时，早上8点

0 23-7/2，8 \* \* \* date

#每个月的4号和每个礼拜的礼拜一到礼拜三的早上11点

0 11 4 \* mon-wed date

#1月份日早上4点

0 4 1 jan \* date

```

范例

$crontab -l 列出用户目前的crontab.

/usr/lib/cron/cron.allow表示谁能使用crontab命令。如果它是一个空文件表明没有一个用户能安排作业。如果这个文件不存在，而有另外一个文件/usr/lib/cron/cron.deny,则只有不包括在这个文件中的用户才可以使用crontab命令。如果它是一个空文件表明任何用户都可安排作业。两个文件同时存在时cron.allow优先，如果都不存在，只有超级用户可以安排作业。

crontab文件的一些例子：

```sh

30 21 \* \* \* /usr/local/etc/rc.d/lighttpd restart

#上面的例子表示每晚的21:30重启apache。

45 4 1,10,22 \* \* /usr/local/etc/rc.d/lighttpd restart

#上面的例子表示每月1、10、22日的4 : 45重启apache。

10 1 \* \* 6,0 /usr/local/etc/rc.d/lighttpd restart

#上面的例子表示每周六、周日的1 : 10重启apache。

0,30 18-23 \* \* \* /usr/local/etc/rc.d/lighttpd restart

#上面的例子表示在每天18 : 00至23 : 00之间每隔30分钟重启apache。

0 23 \* \* 6 /usr/local/etc/rc.d/lighttpd restart

#上面的例子表示每星期六的11 : 00 pm重启apache。

\* \*/1 \* \* \* /usr/local/etc/rc.d/lighttpd restart

#每一小时重启apache

\* 23-7/1 \* \* \* /usr/local/etc/rc.d/lighttpd restart

#晚上11点到早上7点之间，每隔一小时重启apache

0 11 4 \* mon-wed /usr/local/etc/rc.d/lighttpd restart

#每月的4号与每周一到周三的11点重启apache

0 4 1 jan \* /usr/local/etc/rc.d/lighttpd restart

#一月一号的4点重启apache

```

例子：

每两个时间值中间使用逗号分隔。

除了数字还有几个个特殊的符号就是”*\*”、”/”和”-”、”,”，\**代表所有的取值范围内的数字，”/”代表每的意思,”\*/5″表示每5个单位，”-”代表从某个数字到某个数字,”,”分开几个离散的数字。

```sh

#每天早上6点

0 6 \* \* \* echo "Good morning." >> /tmp/test.txt //注意单纯echo，从屏幕上看不到任何输出，因为cron把任何输出都email到root的信箱了。

#每两个小时

0 \*/2 \* \* \* echo "Have a break now." >> /tmp/test.txt

#晚上11点到早上8点之间每两个小时，早上八点

0 23-7/2，8 \* \* \* echo "Have a good dream：）" >> /tmp/test.txt

#每个月的4号和每个礼拜的礼拜一到礼拜三的早上11点

0 11 4 \* 1-3 command line

#1月1日早上4点

0 4 1 1 \* command line

```

每次编辑完某个用户的cron设置后，cron自动在/var/spool/cron下生成一个与此用户同名的文件，此用户的cron信息都记录在这个文件中，这个文件是不可以直接编辑的，只可以用crontab -e 来编辑。cron启动后每过一份钟读一次这个文件，检查是否要执行里面的命令。因此此文件修改后不需要重新启动cron服务。

2.编辑/etc/crontab 文件配置cron

`cron 服务每分钟不仅要读一次/var/spool/cron内的所有文件`，

还需要读一次/etc/crontab,

因此我们配置这个文件也能运用cron服务做一些事情。用crontab配置是针对某个用户的，

而编辑`/etc/crontab是针对系统的任务`。此文件的文件格式是：

SHELL=/bin/bash

PATH=/sbin:/bin:/usr/sbin:/usr/bin

MAILTO=root //如果出现错误，或者有数据输出，数据作为邮件发给这个帐号

HOME=/ //使用者运行的路径,这里是根目录

**# run-parts**

01 \* \* \* \* root run-parts /etc/cron.hourly //每小时执行/etc/cron.hourly内的脚本

02 4 \* \* \* root run-parts /etc/cron.daily //每天执行/etc/cron.daily内的脚本

22 4 \* \* 0 root run-parts /etc/cron.weekly //每星期执行/etc/cron.weekly内的脚本

42 4 1 \* \* root run-parts /etc/cron.monthly //每月去执行/etc/cron.monthly内的脚本

二、cron 定时

cron是一个linux下的定时执行工具，可以在无需人工干预的情况下运行作业。由于Cron 是Linux的内置服务，但它不自动起来，可以用以下的方法启动、关闭这个服务：

/sbin/service crond start //启动服务

/sbin/service crond stop //关闭服务

/sbin/service crond restart //重启服务

/sbin/service crond reload //重新载入配置

你也可以将这个服务在系统启动的时候自动启动：

在/etc/rc.d/rc.local这个脚本的末尾加上：

/sbin/service crond start

现在Cron这个服务已经在进程里面了，我们就可以用这个服务了，Cron服务提供以下几种接口供大家使用：

1、直接用crontab命令编辑

cron服务提供crontab命令来设定cron服务的，以下是这个命令的一些参数与说明：

crontab -u //设定某个用户的cron服务，一般root用户在执行这个命令的时候需要此参数

crontab -l //列出某个用户cron服务的详细内容

crontab -r //删除某个用户的cron服务

crontab -e //编辑某个用户的cron服务

比如说root查看自己的cron设置：`crontab -u root -l`

再例如，root想删除fred的cron设置：`crontab -u fred -r`

在编辑cron服务时，编辑的内容有一些格式和约定，输入：crontab -u root -e

进入vi编辑模式，编辑的内容一定要符合下面的格式：\*/1 \* \* \* \* ls >> /tmp/ls.txt

这个格式的前一部分是对时间的设定，后面一部分是要执行的命令，如果要执行的命令太多，可以把这些命令写到一个脚本里面，然后在这里直接调用这个脚本就可以了，调用的时候记得写出命令的完整路径。时间的设定我们有一定的约定，前面五个\*号代表五个数字，数字的取值范围和含义如下：

分钟 （0-59）

小時 （0-23）

日期 （1-31）

月份 （1-12）

星期 （0-6）//0代表星期天

除了数字还有几个个特殊的符号就是"*\*"、"/"和"-"、","，\**代表所有的取值范围内的数字，"/"代表每的意思,"\*/5"表示每5个单位，"-"代表从某个数字到某个数字,","分开几个离散的数字

cron用法很简单：先来一个速成的：

第一步：写cron脚本文件。例如：取名一个 crontest.cron的文本文件，只需要写一行：

             15,30,45,59 \* \* \* \* echo "xgmtest.........." >> xgmtest.txt

             表示，每隔15分钟，执行打印一次命令

第二步：添加定时任务。执行命令 “crontab crontest.cron”。搞定

第三步：如不放心，可以输入 "crontab -l" 查看是否有定时任务

详细信息：

crontab用法

　　crontab命令用于安装、删除或者列出用于驱动cron后台进程的表格。也就是说，用户把需要执行的命令序列放到crontab文件中以获得执行。每个用户都可以有自己的crontab文件。下面就来看看如何创建一个crontab文件。

　　在/var/spool/cron下的crontab文件不可以直接创建或者直接修改。crontab文件是通过crontab命令得到的。现在假设有个用户名为foxy，需要创建自己的一个crontab文件。首先可以使用任何文本编辑器建立一个新文件，然后向其中写入需要运行的命令和要定期执行的时间。

　　然后存盘退出。假设该文件为/tmp/test.cron。再后就是使用crontab命令来安装这个文件，使之成为该用户的crontab文件。键入：

　　crontab test.cron

　　这样一个crontab 文件就建立好了。可以转到/var/spool/cron目录下面查看，发现多了一个foxy文件。这个文件就是所需的crontab 文件。用more命令查看该文件的内容可以发现文件头有三行信息：

　　#DO NOT EDIT THIS FILE -edit the master and reinstall.

　　#（test.cron installed on Mon Feb 22 14:20:20 1999）

　　#（cron version --$Id:crontab.c，v 2.13 1994/01/17 03:20:37 vivie Exp $）

　　大概意思是：

　　#切勿编辑此文件——如果需要改变请编辑源文件然后重新安装。

　　#test.cron文件安装时间：14:20:20 02/22/1999

　　如果需要改变其中的命令内容时，还是需要重新编辑原来的文件，然后再使用crontab命令安装。

　　可以使用crontab命令的用户是有限制的。如果/etc/cron.allow文件存在，那么只有其中列出的用户才能使用该命令；如果该文件不存在但cron.deny文件存在，那么只有未列在该文件中的用户才能使用crontab命令；如果两个文件都不存在，那就取决于一些参数的设置，可能是只允许超级用户使用该命令，也可能是所有用户都可以使用该命令。

　　crontab命令的语法格式如下：

　　crontab [-u user] file

　　crontab [-u user]{-l -r -e}

　　第一种格式用于安装一个新的crontab 文件，安装 淳褪莊ile所指的文件，如果使用“-”符号作为文件名，那就意味着使用标准输入作为安装来源。

　　-u 如果使用该选项，也就是指定了是哪个具体用户的crontab 文件将被修改。如果不指定该选项，crontab 将默认是操作者本人的crontab ，也就是执行该crontab 命令的用户的crontab 文件将被修改。但是请注意，如果使用了su命令再使用crontab 命令很可能就会出现混乱的情况。所以如果是使用了su命令，最好使用-u选项来指定究竟是哪个用户的crontab文件。

　　-l 在标准输出上显示当前的crontab。

　　-r 删除当前的crontab文件。

　　-e 使用VISUAL或者EDITOR环境变量所指的编辑器编辑当前的crontab文件。当结束编辑离开时，编辑后的文件将自动安装。

　　[例7]

　　# crontab -l #列出用户目前的crontab。

　　10 6 \* \* \* date

　　0 /2 \* \* date

　　0 23-7/2，8 \* \* \* date

　　#

　　在crontab文件中如何输入需要执行的命令和时间。该文件中每行都包括六个域，其中前五个域是指定命令被执行的时间，最后一个域是要被执行的命令。每个域之间使用空格或者制表符分隔。格式如下：

　　minute hour day-of-month month-of-year day-of-week commands

　　第一项是分钟，第二项是小时，第三项是一个月的第几天，第四项是一年的第几个月，第五项是一周的星期几，第六项是要执行的命令。这些项都不能为空，必须填入。如果用户不需要指定其中的几项，那么可以使用\*代替。因为\*是统配符，可以代替任何字符，所以就可以认为是任何时间，也就是该项被忽略了。在表4- 1中给出了每项的合法范围。

　表4-1　指定时间的合法范围

时间 minute hour day-of-month month-of-year day-of-week

合法值 00-59 00-23 01-31 01-12 0-6 (0 is sunday)

　　这样用户就可以往crontab 文件中写入无限多的行以完成无限多的命令。命令域中可以写入所有可以在命令行写入的命令和符号，其他所有时间域都支持列举，也就是域中可以写入很多的时间值，只要满足这些时间值中的任何一个都执行命令，每两个时间值中间使用逗号分隔。

　除了数字还有几个个特殊的符号就是"\*"、"/"和"-"、","，\*代表所有的取值范围内的数字，"/"代表每的意思,"/5"表示每5个单位，"-"代表从某个数字到某个数字,","分开几个离散的数字。

几个例子：

```sh

每天早上6点

0 6 \* \* \* echo "Good morning." >> /tmp/test.txt //注意单纯echo，从屏幕上看不到任何输出，因为cron把任何输出都email到root的信箱了。每两个小时

0 \*/2 \* \* \* echo "Have a break now." >> /tmp/test.txt晚上11点到早上8点之间每两个小时，早上八点

0 23-7/2，8 \* \* \* echo "Have a good dream：）" >> /tmp/test.txt每个月的4号和每个礼拜的礼拜一到礼拜三的早上11点

0 11 4 \* 1-3 command line1月1日早上4点

0 4 1 1 \* command line SHELL=/bin/bash PATH=/sbin:/bin:/usr/sbin:/usr/bin MAILTO=root //如果出现错误，或者有数据输出，数据作为邮件发给这个帐号 HOME=/        //使用者运行的路径,这里是根目录 # run-parts 01 \* \* \* \* root run-parts /etc/cron.hourly //每小时执行/etc/cron.hourly内的脚本 02 4 \* \* \* root run-parts /etc/cron.daily //每天执行/etc/cron.daily内的脚本 22 4 \* \* 0 root run-parts /etc/cron.weekly //每星期执行/etc/cron.weekly内的脚本 42 4 1 \* \* root run-parts /etc/cron.monthly //每月去执行/etc/cron.monthly内的脚本 大家注意"run-parts"这个参数了，如果去掉这个参数的话，后面就可以写要运行的某个脚本名，而不是文件夹名了。

这就是表示任意天任意月，其实就是每天的下午4点、5点、6点的5 min、15 min、25 min、35 min、45 min、55 min时执行命令。

5，15，25，35，45，55 16，17，18 \* \* \* command在每周一，三，五的下午3：00系统进入维护状态，重新启动系统。那么在crontab 文件中就应该写入如下字段：

　　00 15 \* \* 1，3，5 shutdown -r +5然后将该文件存盘为foxy.cron，再键入crontab foxy.cron安装该文件。

每小时的10分，40分执行用户目录下的innd/bbslin这个指令：

　　10，40 \* \* \* \* innd/bbslink每小时的1分执行用户目录下的bin/account这个指令：

　1 \* \* \* \* bin/account每天早晨三点二十分执行用户目录下如下所示的两个指令（每个指令以;分隔）：

20 3 \* \* \* （/bin/rm -f expire.ls logins.bad;bin/expire$#@62;expire.1st）

每年的一月和四月，4号到9号的3点12分和3点55分执行/bin/rm -f expire.1st这个指令，并把结果添加在mm.txt这个文件之后（mm.txt文件位于用户自己的目录位置）。

　　12,55 3 4-9 1,4 \* /bin/rm -f expire.1st$#@62;$#@62;mm.txt

```

crontab的正确用法(简短节省大家时间)命令： 1 man cron 2 man crontab 3 man 5 crontab ==> 中间有个5。命令"crontab -e": -------------------------------------------------------------------------------- PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/X11R6/bin #这是盲点，没想过要设，一次次的失望。［冒号分隔］ DISPLAY=:0.0 #加上吧，有冒号有句点。 # 分 钟 日 月 周 命令 1 23 1 8 \* shutdown -h +3 #1月8日23点1分关机(延时3分钟) 1,4,7 23 1 8 \* # 1分，4分，7分[1月8日23点] *\*/3 23 1 8 \* # 0,3,6,9...每隔3分钟（或每过3分钟，与每3分钟有点不同） [1-10]/3 23 1 8 \* # 1分，4分，7分[从1分起，逐次加3分钟，直到大于10==>这是 正确解法] ＠reboot shutdown -k now #这行能把你吓坏，一开机就关机（要是把"-k"换成"-h"的话）。 "@reboot"表示启动时执行。 ---------------------------------END-------------------------------------------- 另一个脚本： \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****\*\*\* [0-59]/5 23 \* \* \* shutdown -h now #每天晚上11点每隔5分钟关一次机，愿望很好，但是将不会执行，因为没设PATH变量，解决方法见下行。 [0-59]/5 23 \* \* \* /sbin/shutdown -h now #保证关得很死，给出了路径。 0 6 1 8 \* xmms /music/zhangchu/轻取.mp3 #明天早上6点钟放首歌叫我。但睡到7点钟还是没听到音乐，伤心失望。[DISPLAY=:0.0]。对本行的最后一点补充：现今99%主板都支持定时开机，没有闹钟的话可以叫电脑叫醒你，选项在BIOS电源里面［wake up by alarm]。***\*\****\*\****\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****\*\****\*END\****\*\****\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**\*\* 可以自由修改，最初发布于Linuxsir网站。

我们来看一个超级用户的crontab文件：

 　　#Run the ‘atrun' program every minutes 　　#This runs anything that's due to run from ‘at'.See man ‘at' or ‘atrun'. 　　0,5,10,15,20,25,30,35,40,45,50,55 \* \* \* \* /usr/lib/atrun 　　40 7 \* \* \* updatedb 　　8,10,22,30,39,46,54,58 \* \* \* \* /bin/sync

## Linux定时任务Crontab命令详解

\* [Linux定时任务Crontab命令详解\_Linux教程\_Linux公社-Linux系统门户网站 ](http://www.linuxidc.com/Linux/2015-10/124478.htm)

crontab的语法

crontab [-u username] [-l

-e

-r]

选项与参数：

-u ：只有 root 才能进行这个任务，亦即帮其他使用者创建/移除 crontab 工作排程；

-e ：编辑 crontab 的工作内容

-l ：查阅 crontab 的工作内容

-r ：移除所有的 crontab 的工作内容，若仅要移除一项，请用 -e 去编辑

查询使用者目前的 crontab 内容:

crontab -l

## Linux的cron和crontab

Linux的cron和crontab - iTech - 博客园

 http://www.cnblogs.com/itech/archive/2011/02/09/1950226.html

 转自：http://blogold.chinaunix.net/u/31547/showart\_438018.html

一 cron

crond位于/etc/rc.d/init.d/crond 或 /etc/init.d 或 /etc/rc.d /rc5.d/S90crond,最总引用/var/lock/subsys/crond。

cron是一个linux下的定时执行工具（相当于windows下的scheduled task），可以在无需人工干预的情况下定时地运行任务task。由于cron 是Linux的service（deamon），可以用以下的方法启动、关闭这个服务：

/sbin/service crond start //启动服务

/sbin/service crond stop //关闭服务

/sbin/service crond restart //重启服务

/sbin/service crond reload //重新载入配置

你也可以将这个服务在系统启动的时候自动启动：

在/etc/rc.d/rc.local这个脚本的末尾加上：

/sbin/service crond start

现在cron这个服务已经在进程里面了，我们就可以用这个服务了。

二 crontab

crontab位于/usr/bin/crontab。

cron服务提供crontab命令来设定cron服务的，以下是这个命令的一些参数与说明：

crontab -u //设定某个用户的cron服务，一般root用户在执行这个命令的时候需要此参数

crontab -l //列出某个用户cron服务的详细内容

crontab -r //删除某个用户的cron服务

crontab -e //编辑某个用户的cron服务

比如说root查看自己的cron设置：crontab -u root -l

再例如，root想删除fred的cron设置：crontab -u fred -r

在编辑cron服务时，编辑的内容有一些格式和约定，输入：crontab -u root -e 进入vi编辑模式，编辑的内容一定要符合下面的格式：

\*/1 \* \* \* \* ls >> /tmp/ls.txt

这个格式的前一部分是对时间的设定，后面一部分是要执行的命令，如果要执行的命令太多，可以把这些命令写到一个脚本里面，然后在这里直接调用这个脚本就可以了，调用的时候记得写出命令的完整路径。时间的设定我们有一定的约定，前面五个\*号代表五个数字，数字的取值范围和含义如下：

分钟　（0-59）

小時　（0-23）

日期　（1-31）

月份　（1-12）

星期　（0-6）//0代表星期天

除了数字还有几个个特殊的符号就是"*\*"、"/"和"-"、","，\**代表所有的取值范围内的数字，"/"代表每的意思,"\*/5"表示每5个单位，"-"代表从某个数字到某个数字,","分开几个离散的数字。以下举几个例子说明问题：

每天早上6点

0 6 \* \* \* echo "Good morning." >> /tmp/test.txt //注意单纯echo，从屏幕上看不到任何输出，因为cron把任何输出都email到root的信箱了。

每两个小时

0 \*/2 \* \* \* echo "Have a break now." >> /tmp/test.txt

晚上11点到早上8点之间每两个小时，早上八点

0 23-7/2，8 \* \* \* echo "Have a good dream：）" >> /tmp/test.txt

每个月的4号和每个礼拜的礼拜一到礼拜三的早上11点

0 11 4 \* 1-3 command line

1月1日早上4点

0 4 1 1 \* command line

每次编辑完某个用户的cron设置后，cron自动在/var/spool/cron下生成一个与此用户同名的文件，此用户的cron信息都记录在这个文件中，这个文件是不可以直接编辑的，只可以用crontab -e 来编辑。cron启动后每过一份钟读一次这个文件，检查是否要执行里面的命令。因此此文件修改后不需要重新启动cron服务。

三 编辑/etc/crontab配置文件

cron的系统级配置文件位于/etc/crontab。

cron服务每分钟不仅要读一次/var/spool/cron内的所有文件，还需要读一次/etc/crontab配置文件,因此我们配置这个文件也能运用 cron服务做一些事情。用crontab -e进行的配置是针对某个用户的，而编辑/etc/crontab是针对系统的任务。此文件的文件格式是：

SHELL=/bin/bash

PATH=/sbin:/bin:/usr/sbin:/usr/bin

MAILTO=root      //如果出现错误，或者有数据输出，数据作为邮件发给这个帐号

HOME=/    //使用者运行的路径,这里是根目录

**# run-parts**

01 \* \* \* \* root run-parts /etc/cron.hourly //每小时执行/etc/cron.hourly内的脚本

02 4 \* \* \* root run-parts /etc/cron.daily //每天执行/etc/cron.daily内的脚本

22 4 \* \* 0 root run-parts /etc/cron.weekly //每星期执行/etc/cron.weekly内的脚本

42 4 1 \* \* root run-parts /etc/cron.monthly //每月去执行/etc/cron.monthly内的脚本

大家注意"run-parts"这个参数了，如果去掉这个参数的话，后面就可以写要运行的某个脚本名，而不是文件夹名了。

四 实例

--------------------------------------

基本格式 : [参数间必须使用空格隔开]

\*　　\*　　\*　　\*　　\*　　command

分　时　日　月　周　命令

第1列表示分钟1～59 每分钟用\*或者 \*/1表示

第2列表示小时1～23（0表示0点）

第3列表示日期1～31

第4列表示月份1～12

第5列标识号星期0～6（0表示星期天）

第6列要运行的命令

crontab文件的一些例子：

30 21 \* \* \* /usr/local/etc/rc.d/lighttpd restart

上面的例子表示每晚的21:30重启lighttpd 。

45 4 1,10,22 \* \* /usr/local/etc/rc.d/lighttpd restart

上面的例子表示每月1、10、22日的4 : 45重启lighttpd 。

10 1 \* \* 6,0 /usr/local/etc/rc.d/lighttpd restart

上面的例子表示每周六、周日的1 : 10重启lighttpd 。

0,30 18-23 \* \* \* /usr/local/etc/rc.d/lighttpd restart

上面的例子表示在每天18 : 00至23 : 00之间每隔30分钟重启lighttpd 。

0 23 \* \* 6 /usr/local/etc/rc.d/lighttpd restart

上面的例子表示每星期六的11 : 00 pm重启lighttpd 。

\* \*/1 \* \* \* /usr/local/etc/rc.d/lighttpd restart

每一小时重启lighttpd

\* 23-7/1 \* \* \* /usr/local/etc/rc.d/lighttpd restart

晚上11点到早上7点之间，每隔一小时重启lighttpd

0 11 4 \* mon-wed /usr/local/etc/rc.d/lighttpd restart

每月的4号与每周一到周三的11点重启lighttpd

0 4 1 jan \* /usr/local/etc/rc.d/lighttpd restart

一月一号的4点重启lighttpd

五 特殊用法

@hourly /usr/local/www/awstats/cgi-bin/awstats.sh

使用 @hourly 對應的是 0 \* \* \* \*, 還有下述可以使用:

string            meaning

------           -------

@reboot        Run once, at startup.

@yearly         Run once a year, "0 0 1 1 \*".

@annually      (same as @yearly)

@monthly       Run once a month, "0 0 1 \* \*".

@weekly        Run once a week, "0 0 \* \* 0".

@daily           Run once a day, "0 0 \* \* \*".

@midnight      (same as @daily)

@hourly         Run once an hour, "0 \* \* \* \*".

完！

## 定时任务crontab如何实现每秒执行？ - 脚本小娃子 - 博客园

https://www.cnblogs.com/shengulong/p/9099992.html

linux crontab 命令，最小的执行时间是一分钟。如需要在小于一分钟内重复执行，可以有两个方法实现。

方法一：crontab -l内容如下，则每10秒执行一次/home/fdipzone/php/crontab/tolog.php

\* \* \* \* \* php /home/fdipzone/php/crontab/tolog.php

\* \* \* \* \* sleep 10; php /home/fdipzone/php/crontab/tolog.php

\* \* \* \* \* sleep 20; php /home/fdipzone/php/crontab/tolog.php

\* \* \* \* \* sleep 30; php /home/fdipzone/php/crontab/tolog.php

\* \* \* \* \* sleep 40; php /home/fdipzone/php/crontab/tolog.php

\* \* \* \* \* sleep 50; php /home/fdipzone/php/crontab/tolog.php

方法二：脚本**crontab.sh**内容如下：

[复制代码](javascript:void(0);)

#!/bin/bash

step=2 #间隔的秒数，不能大于60

for (( i = 0; i < 60; i=(i+step) )); do

$(php '/home/fdipzone/php/crontab/tolog.php')

sleep $step

done

exit 0

[复制代码](javascript:void(0);)

　　crontab -l内容如下：

# m h dom mon dow command

\* \* \* \* \* /home/fdipzone/php/crontab/crontab.sh

如上两步实现了每2秒执行一次/home/fdipzone/php/crontab/crontab.sh

# Ssh

## 1.使用Xshell生成key,免密码登录linux

https://blog.csdn.net/fireofjava/article/details/40624215

我们通常在Xshell使用命令ssh user@ip远程登录linux，这样会提示我们输入密码比较麻烦，想免密码登录的话，我们可以生成相应的key，然后放到远程服务器上，下面是具体的操作。

到这里我们就生成的公钥和私钥了，我们把公钥id\_ras\_1024.pub上传到服务器上的/home/wsuser/.ssh下面，注意这里的wsuser是我登录的用户名。

接着远程登录到目标机器上，执行下面的命令

cat id\_ras\_1024.pub >> authorized\_keys

接下来在Xshell里面新建会话，

这样配置了以后就可以直接登录远程机器了。

## 02.Pseudo-terminal will not be allocated because stdin is not a terminal

ssh无法登录,提示Pseudo-terminal will not be allocated because stdin is not a terminal. - 飘飘雪 - 博客园 https://www.cnblogs.com/wangcp-2014/p/6691445.html

当远程通过ssh登录主机时出现Pseudo-terminal will not be allocated because stdin is not a terminal. 错误

字面意思是伪终端将无法分配，因为标准输入不是终端。

所以需要增加-t -t参数来强制伪终端分配，即使标准输入不是终端。

to force pseudo-tty allocation even if stdin isn't a terminal.

参考样例如下:

ssh -tt user1@host1 -p 9527

## 03.Connection closed by remote host

ssh连接提示 "Connection closed by remote host" - Aaron-Ma的专栏 - CSDN博客 https://blog.csdn.net/mjm26/article/details/52242398/

解决方法一. 把SSH连接数改大

修改服务器上的这个文件：/etc/ssh/sshd\_config 找到这行：

**# MaxSessions 10**

去掉前面的"#" 并把数字改大，最后重启sshd service sshd restart 然后重新连接即可.

解决方法二.  每次正常退出SSH连接

每次执行完命令后用输入"exit" 退出, 防止连接数过多.

解决方法三. 重启

如果连接不上远程主机可以在空间服务商的网页控制面板中重启服务

查看Linux服务器有多少个用户连接ssh的方法

命令行输入

who

如果想要断开某个ssh 用户 可以在命令行输入

ps aux |grep sshd

得到pid 再用命令关闭那个pid:

kill -111

**---------------------**

作者：Aaron-Ma

来源：CSDN

原文：https://blog.csdn.net/mjm26/article/details/52242398/

版权声明：本文为博主原创文章，转载请附上博文链接！

## 06.ppk和pem证书互转

ppk和pem证书互转 - Eric.S - 博客园 https://www.cnblogs.com/oskb/p/8558650.html

首先你得去下载个putty

pem：通用证书格式

ppk：为putty下面的专有格式

pem->ppk

直接通过putty下的puttygen.exe 选的Load private key， 然后save private key就可以了

ppk->pem

这个也算是选择load private key 然后要选择菜单Conversions下面的Export SSH key

## ECDSA host key ip地址 for has changed and you have requested strict checking

ECDSA host key "ip地址" for has changed and you have requested strict checking. - - ITeye博客 http://1197757723.iteye.com/blog/2398053

Linux SSH登录服务器报ECDSA host key "ip地址" for has changed and you have requested strict checking错误 - 专注于移动互联网 - CSDN博客 http://blog.csdn.net/ausboyue/article/details/52775281

丨版权说明 : 《Linux SSH登录服务器报ECDSA host key "ip地址" for  has changed and you have requested strict checking错误》于当前CSDN博客和乘月网属同一原创，转载请说明出处，谢谢。

Linux SSH命令用了那么久，第一次遇到这样的错误：ECDSA host key "ip地址" for  has changed and you have requested strict checking.记录下方便记忆。

解决方案：在终端上输入以下命令：

[plain] view plain copy

ssh-keygen -R "你的远程服务器ip地址"

目的是清除你当前机器里关于你的远程服务器的缓存和公钥信息，注意是大写的字母“R”。

原因分析：根据个人的情况，原因是我的云服务器重装了系统（清除了与我本地SSH连接协议相关信息），本地的SSH协议信息便失效了。SSH连接相同的ip地址时因有连接记录直接使用失效的协议信息去验证该ip服务器，所以会报错，使用上述命令便可以清除known\_hosts里旧缓存文件。

延伸：远程服务器的ssh服务被卸载重装或ssh相关数据（协议信息）被删除也会导致这个错误。

## ssh远程执行命令并自动退出

https://blog.csdn.net/fdipzone/article/details/23000201

ssh命令格式如下：

usage: ssh [-1246AaCfgKkMNnqsTtVvXxYy] [-b bind\_address] [-c cipher\_spec]

           [-D [bind\_address:]port] [-e escape\_char] [-F configfile]

           [-I pkcs11] [-i identity\_file]

           [-L [bind\_address:]port:host:hostport]

           [-l login\_name] [-m mac\_spec] [-O ctl\_cmd] [-o option] [-p port]

           [-R [bind\_address:]port:host:hostport] [-S ctl\_path]

           [-W host:port] [-w local\_tun[:remote\_tun]]

           [user@]hostname [command]

主要参数说明：

-l 指定登入用户

-p 设置端口号

-f 后台运行，并推荐加上 -n 参数

-n 将标准输入重定向到 /dev/null，防止读取标准输入

-N 不执行远程命令，只做端口转发

-q 安静模式，忽略一切对话和错误提示

-T 禁用伪终端配置

ssh 执行远程命令格式：

ssh [options][remote host][command]

假设远程服务器IP是192.168.110.34

例：查看远程服务器的cpu信息

ssh -l www-online 192.168.110.34 "cat /proc/cpuinfo"

www-online@onlinedev01:~$ ssh -l www-online 192.168.110.34 "cat /proc/cpuinfo"

www-online@192.168.110.34's password:

processor       : 0

vendor\_id       : GenuineIntel

cpu family      : 6

model           : 26

model name      : Intel(R) Xeon(R) CPU           E5506  @ 2.13GHz

stepping        : 5

cpu MHz         : 2128.000

cache size      : 4096 KB

fpu             : yes

fpu\_exception   : yes

cpuid level     : 11

wp              : yes

flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss syscall nx rdtscp lm constant\_tsc arch\_perfmon pebs bts rep\_good xtopology tsc\_reliable nonstop\_tsc aperfmperf pni ssse3 cx16 sse4\_1 sse4\_2 popcnt hypervisor lahf\_lm

bogomips        : 4256.00

clflush size    : 64

cache\_alignment : 64

address sizes   : 40 bits physical, 48 bits virtual

power management:

processor       : 1

vendor\_id       : GenuineIntel

cpu family      : 6

model           : 26

model name      : Intel(R) Xeon(R) CPU           E5506  @ 2.13GHz

stepping        : 5

cpu MHz         : 2128.000

cache size      : 4096 KB

fpu             : yes

fpu\_exception   : yes

cpuid level     : 11

wp              : yes

flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss syscall nx rdtscp lm constant\_tsc arch\_perfmon pebs bts rep\_good xtopology tsc\_reliable nonstop\_tsc aperfmperf pni ssse3 cx16 sse4\_1 sse4\_2 popcnt hypervisor lahf\_lm

bogomips        : 4260.80

clflush size    : 64

cache\_alignment : 64

address sizes   : 40 bits physical, 48 bits virtual

power management:

例：执行远程服务器的sh文件

首先在远程服务器的/home/www-online/下创建一个uptimelog.sh脚本

#!/bin/bash

uptime >> 'uptime.log'

exit 0

使用chmod增加可执行权限

chmod u+x uptimelog.sh

在本地调用远程的uptimelog.sh

ssh -l www-online 192.168.110.34 "/home/www-online/uptimelog.sh"

执行完成后,在远程服务器的/home/www-online/中会看到uptime.log文件，显示uptime内容

www-online@nmgwww34:~$ tail -f uptime.log

21:07:34 up 288 days,  8:07,  1 user,  load average: 0.05, 0.19, 0.31

例：执行远程后台运行sh

首先把uptimelog.sh修改一下,修改成循环执行的命令。作用是每一秒把uptime写入uptime.log

#!/bin/bash

while :

do

  uptime >> 'uptime.log'

  sleep 1

done

exit 0

我们需要这个sh在远程服务器以后台方式运行，命令如下：

ssh -l www-online 192.168.110.34 "/home/www-online/uptimelog.sh &"

www-online@onlinedev01:~$ ssh -l www-online 192.168.110.34 "/home/www-online/uptimelog.sh &"

www-online@192.168.110.34's password:

输入密码后，发现一直停住了，而在远程服务器可以看到，程序已经以后台方式运行了。

www-online@nmgwww34:~$ ps aux|grep uptimelog.sh

1007     20791  0.0  0.0  10720  1432 ?        S    21:25   0:00 /bin/bash /home/www-online/uptimelog.sh

原因是因为uptimelog.sh一直在运行，并没有任何返回，因此调用方一直处于等待状态。

我们先kill掉远程服务器的uptimelog.sh进程，然后对应此问题进行解决。

ssh 调用远程命令后不能自动退出解决方法

可以将标准输出与标准错误输出重定向到/dev/null，这样就不会一直处于等待状态。

ssh -l www-online 192.168.110.34 "/home/www-online/uptimelog.sh > /dev/null 2>&1 &"

www-online@onlinedev01:~$ ssh -l www-online 192.168.110.34 "/home/www-online/uptimelog.sh > /dev/null 2>&1 &"

www-online@192.168.110.34's password:

www-online@onlinedev01:~$

但这个ssh进程会一直运行在后台，浪费资源，因此我们需要自动清理这些进程。

实际上，想ssh退出，我们可以在ssh执行完成后kill掉ssh这个进程来实现。

首先，创建一个sh执行ssh的命令,这里需要用到ssh的 -f 与 -n 参数，因为我们需要ssh也以后台方式运行，这样才可以获取到进程号进行kill操作。

创建ssh\_uptimelog.sh，脚本如下

#!/bin/bash

ssh -f -n -l www-online 192.168.110.34 "/home/www-online/uptimelog.sh &" # 后台运行ssh

pid=$(ps aux | grep "ssh -f -n -l www-online 192.168.110.34 /home/www-online/uptimelog.sh" | awk '{print $2}' | sort -n | head -n 1) # 获取进程号

echo "ssh command is running, pid:${pid}"

sleep 3 && kill ${pid} && echo "ssh command is complete" # 延迟3秒后执行kill命令，关闭ssh进程，延迟时间可以根据调用的命令不同调整

exit 0

可以看到，3秒后会自动退出

www-online@onlinedev01:~$ ./ssh\_uptimelog.sh

www-online@192.168.110.34's password:

ssh command is running, pid:10141

ssh command is complete

www-online@onlinedev01:~$

然后查看远程服务器，可以见到uptimelog.sh 在后台正常执行。

www-online@nmgwww34:~$ ps aux|grep uptime

1007     28061  0.1  0.0  10720  1432 ?        S    22:05   0:00 /bin/bash /home/www-online/uptimelog.sh

查看uptime.log，每秒都有uptime数据写入。

www-online@nmgwww34:~$ tail -f uptime.log

22:05:44 up 288 days,  9:05,  1 user,  load average: 0.01, 0.03, 0.08

22:05:45 up 288 days,  9:05,  1 user,  load average: 0.01, 0.03, 0.08

22:05:46 up 288 days,  9:05,  1 user,  load average: 0.01, 0.03, 0.08

22:05:47 up 288 days,  9:05,  1 user,  load average: 0.01, 0.03, 0.08

22:05:48 up 288 days,  9:05,  1 user,  load average: 0.01, 0.03, 0.08

# Shell

## shell获取系统当前时间并格式化\_chenopodium的博客-CSDN博客

https://blog.csdn.net/sinat\_29100331/article/details/79091323

time=$(date "+%Y%m%d-%H%M%S")

or

time=$(date "+%Y-%m-%d %H:%M:%S")

......等各种自己想要的格式

echo "${time}"

上面两行简单的代码就是shell获取当前时间并按照自己想要的格式输出。

需要注意几点

date后面有一个空格，否则无法识别命令，shell对空格还是很严格的。

Y显示4位年份，如：2018；y显示2位年份，如：18。m表示月份；M表示分钟。d表示天，而D则表示当前日期，如：1/18/18(也就是2018.1.18)。H表示小时，而h显示月份(有点懵逼)。s显示当前秒钟，单位为毫秒；S显示当前秒钟，单位为秒。

## linux-检测远程端口是否打开\_房兰峰的博客-CSDN博客

<https://blog.csdn.net/qq_24784495/article/details/78183984>

常用telnet 110.101.101.101 80方式测试远程主机端口是否打开。

方法1.nmap ip -p port 测试端口

1.1 nmap ip 显示全部打开的端口

[root@localhost etc]# nmap 172.16.3.34 80

PORT STATE SERVICE

80/tcp open http

81/tcp open hosts2-ns

88/tcp open kerberos-sec

445/tcp open microsoft-ds

631/tcp open ipp

3031/tcp open eppc

5900/tcp open vnc

根据显示close/open确定端口是否打开。

方法2. nc -v host port

端口未打开返回状态为非0

一：telnet

此法常被用来检测是个远端端口是否通畅。

[root@oldboy ~]# telnet baidu.com 80

Trying 123.125.114.144...

Connected to baidu.com (123.125.114.144). #==>出现Connected表示连通了，说明百度的80端口开放的

Escape character is '^]'. #==>ctrl+]退出此地。

^]

telnet> quit

Connection closed.

如果写脚本通过telnet检查端口可以用下面的方法：

[root@localhost etc]#  echo -e "\n"|telnet baidu.com 80|grep Connected

Connection closed by foreign host.

Connected to baidu.com (123.125.114.144).

二：通过nmap来检查端口是否通畅

[root@localhost etc]# nmap 172.16.3.34 -p 80

Starting Nmap 5.51 ( http://nmap.org ) at 2017-08-12 22:31 EDT

Nmap scan report for 172.16.3.34

Host is up (0.00013s latency).

PORT STATE SERVICE

80/tcp open http

如果写脚本通过nmap检查端口可以用下面的方法：

   PORT\_COUNT=`nmap $ip\_add  -p $port|grep open|wc -l`

   [[ $PORT\_COUNT -ge 1 ]] && echo "$ip\_add $port is ok." || echo "$ip\_add $port is unknown."

三：通过nc命令检查

[root@localhost etc]# nc -w 5 172.16.3.34 80 && echo OK

OK

[root@localhost etc]# nc -w 5 172.16.3.34 80 && echo OK|| echo NO

OK

扩展-安装nmap

[root@localhost etc]# yum install nmap

## shell脚本中判断上一个命令是否执行成功 - 邯郸-小刀 - 博客园

<https://www.cnblogs.com/xzlive/p/9482204.html>

shell脚本中判断上一个命令是否执行成功

shell中使用符号“$?”来显示上一条命令执行的返回值，如果为0则代表执行成功，其他表示失败。

结合if-else语句实现判断上一个命令是否执行成功。

示例如下：

if [ $? -ne 0 ]; then

echo "failed"

else

echo "succeed"

fi

或者：

if [ $? -eq 0 ]; then

echo "succeed"

else

echo "failed"

fi

telnet baidu.com 80