第3章　后台执行命令

当你在终端或控制台工作时，可能不希望由于运行一个作业而占住了屏幕，因为可能还有更重要的事情要做，比如阅读电子邮件。对于密集访问磁盘的进程，你可能希望它能够在每天的非负荷高峰时间段运行。为了使这些进程能够在后台运行，也就是说不在终端屏幕上运行，有几种选择方法可供使用。

在本章中我们将讨论：

- 设置crontab文件，并用它来提交作业。
- 使用at命令来提交作业。
- 在后台提交作业。
- 使用nohup命令提交作业。

名词解释：

cron　系统调度进程。可以使用它在每天的非高峰负荷时间段运行作业，或在一周或一月中的不同时段运行。

At　at命令。使用它在一个特定的时间运行一些特殊的作业，或在晚一些的非负荷高峰时间段或高峰负荷时间段运行。

&　使用它在后台运行一个占用时间不长的进程。

Nohup　使用它在后台运行一个命令，即使在用户退出时也不受影响。

3.1　cron和crontab

cron是系统主要的调度进程，可以在无需人工干预的情况下运行作业。有一个叫做crontab的命令允许用户提交、编辑或删除相应的作业。每一个用户都可以有一个 crontab文件来保存调度信息。可以使用它运行任意一个 shell脚本或某个命令，每小时运行一次，或一周三次，这完全取决于你。每一个用户都可以有自己的 crontab文件，但在一个较大的系统中，系统管理员一般会禁止这些文件，而只在整个系统保留一个这样的文件。系统管理员是通过cron.deny和cron.allow这两个文件来禁止或允许用户拥有自己的 crontab文件。

3.1.1　crontab的域

为了能够在特定的时间运行作业，需要了解 crontab文件每个条目中各个域的意义和格式。下面就是这些域：

第1列　　分钟1～59
第2列　　小时1～23（0表示子夜）
第3列　　日1～31
第4列　　月1～12
第5列　　星期0～6（0表示星期天）
第6列　　要运行的命令

下面是crontab的格式：

分<>时<>日<>月<>星期<>要运行的命令

其中<>表示空格。

Crontab文件的一个条目是从左边读起的，第一列是分，最后一列是要运行的命令，它位于星期的后面。

在这些域中，可以用横杠 -来表示一个时间范围，例如你希望星期一至星期五运行某个作业，那么可以在星期域使用 1-5来表示。还可以在这些域中使用逗号“ , ”，例如你希望星期一和星期四运行某个作业，只需要使用 1,4来表示。可以用星号 *来表示连续的时间段。如果你对某个表示时间的域没有特别的限定，也应该在该域填入 *。该文件的每一个条目必须含有 5 个时间域，而且每个域之间要用空格分隔。该文件中所有的注释行要在行首用 #来表示。

3.1.2 crontab条目举例

这里有crontab文件条目的一些例子：

```
30 21* * * /apps/bin/cleanup.sh
```

上面的例子表示每晚的21:30运行/apps/bin目录下的cleanup.sh。

```
45 4 1,10,22 * * /apps/bin/backup.sh
```

上面的例子表示每月1、10、22日的4:45运行/apps/bin目录下的backup.sh。

```
10 1 * * 6,0 /bin/find -name "core" -exec rm {} \;
```

上面的例子表示每周六、周日的1:10运行一个find命令。

```
0,30 18-23 * * * /apps/bin/dbcheck.sh
```

上面的例子表示在每天18:00至23:00之间每隔30分钟运行/apps/bin目录下的dbcheck.sh。

```
0 23 * * 6 /apps/bin/qtrend.sh
```

上面的例子表示每星期六的11:00pm运行/apps/bin目录下的qtrend.sh。

你可能已经注意到上面的例子中，每个命令都给出了绝对路径。当使用 crontab运行shell脚本时，要由用户来给出脚本的绝对路径，设置相应的环境变量。记住，既然是用户向 cron提交了这些作业，就要向cron提供所需的全部环境。不要假定cron知道所需要的特殊环境，它其实并不知道。所以你要保证在 shell脚本中提供所有必要的路径和环境变量，除了一些自动设置的全局变量。

如果cron不能运行相应的脚本，用户将会收到一个邮件说明其中的原因。

3.1.3 crontab命令选项

crontab命令的一般形式为：

```
Crontab [-u user] -e -l -r
```

其中：

-u 用户名。

-e 编辑crontab文件。

-l 列出crontab文件中的内容。

-r 删除crontab文件。

如果使用自己的名字登录，就不用使用 -u选项，因为在执行 crontab命令时，该命令能够

知道当前的用户。

3.1.4 创建一个新的crontab文件

在考虑向cron进程提交一个 crontab文件之前，首先要做的一件事情就是设置环境变量EDITOR。cron进程根据它来确定使用哪个编辑器编辑 crontab文件。99%的UNIX和LINUX用户都使用vi，如果你也是这样，那么你就编辑 $HOME目录下的.profile文件，在其中加入这样一行：

```
EDITOR=vi; export EDITOR
```

然后保存并退出。

不妨创建一个名为<user>cron的文件，其中<user>是用户名，例如， davecron。在该文件中加入如下的内容。

```
# (put your own initials here) echo the date to the console every
# 15 minutes between 6pm and 6am
0,15,30,45 18-06  *  *  *  /bin/echo `date` > /dev/console
```

保存并退出。确信前面5个域用空格分隔。

在上面的例子中，系统将每隔 15分钟向控制台输出一次当前时间。如果系统崩溃或挂起，从最后所显示的时间就可以一眼看出系统是什么时间停止工作的。在有些系统中，用 tty1来表示控制台，可以根据实际情况对上面的例子进行相应的修改。

为了提交你刚刚创建的crontab文件，可以把这个新创建的文件作为 cron命令的参数：

```
$ crontab davecron
```

现在该文件已经提交给cron进程，它将每隔15分钟运行一次。

同时，新创建文件的一个副本已经被放在 /var/spool/cron目录中，文件名就是用户名（即，dave）。

3.1.5 列出crontab文件

为了列出crontab文件，可以用：

```
$ crontab -l
# (crondave installed on Tue May  4 13:07:43 1999)
# DT: echo the date to the console every 30 minutes
 0,15,30,45 18-06  *  *  *  /bin/echo `date` > /dev/tty1
```

你将会看到和上面类似的内容。可以使用这种方法在 $HOME目录中对crontab文件做一备份：

```
$ crontab -l > $HOME/mycron
```

这样，一旦不小心误删了crontab文件，可以用上一节所讲述的方法迅速恢复。

3.1.6 编辑crontab文件

如果希望添加、删除或编辑 crontab文件中的条目，而EDITOR环境变量又设置为vi，那么就可以用vi来编辑crontab文件，相应的命令为：

```
$ crontab -e
```

可以像使用vi编辑其他任何文件那样修改 crontab文件并退出。如果修改了某些条目或添

加了新的条目，那么在保存该文件时， cron会对其进行必要的完整性检查。如果其中的某个域出现了超出允许范围的值，它会提示你。

我们在编辑crontab文件时，没准会加入新的条目。例如，加入下面的一条：

```
#  DT: delete core files, at 3.30am on 1,7,14,21,26 days of each month
30 3  1,7,14,21,26  *  *  /bin/find -name "core" -exec rm {} \;
```

现在保存并退出。最好在 crontab文件的每一个条目之上加入一条注释，这样就可以知道它的功能、运行时间，更为重要的是，知道这是哪位用户的作业。

现在让我们使用前面讲过的crontab -l命令列出它的全部信息：

```
$ crontab -l
# (crondave installed on Tue May  4 13:07:43 1999)
# DT: echo the date to the console every 30 minutes
0,15,30,45 18-06  *  *  *  /bin/echo `date` > /dev/tty1
#  DT: delete core files, at 3.30am on 1,7,14,21,26 days of each
# month
31 3  1,7,14,21,26  *  *  /bin/find -name "core" -exec rm {} \;
```

3.1.7 删除crontab文件

为了删除crontab文件，可以用：

```
$ crontab -r
```

3.1.8 恢复丢失的crontab文件

如果不小心误删了crontab文件，假设你在自己的$HOME目录下还有一个备份，那么可以将其拷贝到/var/spool/cron/<username>，其中<username>是用户名。如果由于权限问题无法完成拷贝，可以用：

```
$ crontab <filename>
```

其中，<filename>是你在$HOME目录中副本的文件名。

我建议你在自己的$HOME目录中保存一个该文件的副本。我就有过类似的经历，有数次误删了crontab文件（因为r键紧挨在e键的右边…）。这就是为什么有些系统文档建议不要直接编辑crontab文件，而是编辑该文件的一个副本，然后重新提交新的文件。

有些crontab的变体有些怪异，所以在使用 crontab命令时要格外小心。如果遗漏了任何选项，crontab可能会打开一个空文件，或者看起来像是个空文件。这时敲 delete键退出，不要按<Ctrl-D>，否则你将丢失crontab文件。

3.2 at命令

at命令允许用户向cron守护进程提交作业，使其在稍后的时间运行。这里稍后的时间可能是指10min以后，也可能是指几天以后。如果你希望在一个月或更长的时间以后运行，最好还是使用crontab文件。

一旦一个作业被提交， at命令将会保留所有当前的环境变量，包括路径，不象 crontab，只提供缺省的环境。该作业的所有输出都将以电子邮件的形式发送给用户，除非你对其输出进行了重定向，绝大多数情况下是重定向到某个文件中。

和crontab一样，根用户可以通过 /etc目录下的at.allow和at.deny文件来控制哪些用户可以

使用at命令，哪些用户不行。不过一般来说，对 at命令的使用不如对 crontab的使用限制那么严格。

at命令的基本形式为：

```
at [-f script] [-m -l -r] [time] [date]
```

其中，

-f script　是所要提交的脚本或命令。

-l　列出当前所有等待运行的作业。 atq命令具有相同的作用。

-r　清除作业。为了清除某个作业，还要提供相应的作业标识（ ID）；有些UNIX变体只接受atrm作为清除命令。

-m　作业完成后给用户发邮件。

time　at命令的时间格式非常灵活；可以是 H、HH.HHMM、HH:MM或H:M，其中H和M分别是小时和分钟。还可以使用 a.m.或p.m.。

date　日期格式可以是月份数或日期数，而且 at命令还能够识别诸如today、tomorrow这样的词。

现在就让我们来看看如何提交作业。

3.2.1 使用at命令提交命令或脚本

使用at命令提交作业有几种不同的形式，可以通过命令行方式，也可以使用 at命令提示符。一般来说在提交若干行的系统命令时，我使用 at命令提示符方式，而在提交 shell脚本时，使用命令行方式。

如果你想提交若干行的命令，可以在 at命令后面跟上日期/时间并回车。然后就进入了 at命令提示符，这时只需逐条输入相应的命令，然后按‘ <CTRL-D> ’退出。下面给出一个例子：

```
$ at 21:10
at> find / -name "passwd" -print
at> <EOT>
warning: commands will be executed using /bin/sh
job 1 at 1999-05-05 21:10
```

其中， <EOT>就是<CTRL-D>。在21:10系统将执行一个简单的 find命令。你应当已经注意到，我所提交的作业被分配了一个唯一标识 job 1。该命令在完成以后会将全部结果以邮件的形式发送给我。

下面就是我从这个邮件中截取的一部分：

```
Subject: Output from your job      1
/etc/passwd
/etc/pam.d/passwd
/etc/uucp/passwd
/tmp/passwd
/root/passwd
/usr/bin/passwd
/usr/doc/uucp-1.06.1/sample/passwd
```

下面这些日期/时间格式都是at命令可以接受的：

```
At 6.45am May12
At 11.10pm
At now + 1 hour
```

At 9am tomorrow
At 15:00 May 24
At now + 10 minutes – this time specification is my own favourite.

如果希望向at命令提交一个shell脚本，使用其命令行方式即可。在提交脚本时使用 -f选项。

```
$ at 3.00pm tomorrow -f /apps/bin/db_table.sh
warning: commands will be executed using /bin/sh
job 8 at 1999-05-06 15:00
```

在上面的例子中，一个叫做db_table.sh的脚本将在明天下午3:00运行。

还可以使用echo命令向at命令提交作业：

```
$ echo find /etc -name "passwd" -print | at now +1 minute
```

3.2.2 列出所提交的作业

一个作业被提交后，可以使用at -l命令来列出所有的作业：

```
$ at -l
2   1999-05-05 23:00 a
3   1999-05-06 06:00 a
4   1999-05-21 11:20 a
1   1999-05-06 09:00 a
```

其中，第一行是作业标识，后面是作业运行的日期 /时间。最后一列a代表at。还可以使用atq命令来完成同样的功能，它是 at命令的一个链接。当提交一个作业后，它就被拷贝到/var/spool/at目录中，准备在要求的时间运行。

```
$ pwd
/var/spool/at
$ ls
a0000200eb7ae4  a0000400ebd228  a0000800eb7ea4  spool
a0000300eb7c88  a0000500eb7d3c  a0000900eb7aaa
```

3.2.3 清除一个作业

清除作业的命令格式为：

```
atrm [job no]或at -r [job no]
```

要清除某个作业，首先要执行at -l命令，以获取相应的作业标识，然后对该作业标识使用at -r命令，清除该作业。

```
$ at -l
2   1999-05-05 23:00 a
3   1999-05-06 06:00 a
4   1999-05-21 11:20 a

$ atrm job 3
$ at -l
2   1999-05-05 23:00 a
4   1999-05-21 11:20 a
```

有些系统使用at-r [job no]命令清除作业。

3.3 &命令

当在前台运行某个作业时，终端被该作业占据；而在后台运行作业时，它不会占据终端。

可以使用&命令把作业放到后台执行。

该命令的一般形式为：

命令 &

为什么要在后台执行命令？因为当在后台执行命令时，可以继续使用你的终端做其他事情。适合在后台运行的命令有 find、费时的打印作业、费时的排序及一些 shell脚本。在后台运行作业时要当心：需要用户交互的命令不要放在后台执行，因为这样你的机器就会在那里傻等。

不过，作业在后台运行一样会将结果输出到屏幕上，干扰你的工作。如果放在后台运行的作业会产生大量的输出，最好使用下面的方法把它的输出重定向到某个文件中：

```
command >out.file 2>&1 &
```

在上面的例子中，所有的标准输出和错误输出都将被重定向到一个叫做 out.file 的文件中。当你成功地提交进程以后，就会显示出一个进程号，可以用它来监控该进程，或杀死它。

3.3.1 向后台提交命令

现在我们运行一个 find命令，查找名为“ srm.conf ”的文件，并把所有标准输出和错误输出重定向到一个叫作 find.dt的文件中：

```
$ find /etc -name "srm.conf" -print >find.dt 2>&1 &
[1] 27015
```

在上面的例子中，在我们成功提交该命令之后，系统给出了它的进程号 27015。

当该作业完成时，按任意键（一般是回车键）就会出现一个提示：

```
[1]+  Done        find /etc "srm.conf" -print
```

这里还有另外一个例子，有一个叫做 ps1的脚本，它能够截断和清除所有的日志文件，我把它放到后台去执行：

```
$ ps1 &
[2] 28535
```

3.3.2 用ps命令查看进程

当一个命令在后台执行的时候，可以用提交命令时所得到的进程号来监控它的运行。在前面的例子中，我们可以按照提交 ps1时得到的进程号，用 ps命令和grep命令列出这个进程：

```
$ ps x|grep 28305
28305  p1 S    0:00 sh /root/ps1
28350  p1 S    0:00 grep 28305
```

如果系统不支持ps x命令，可以用：

```
$ ps -ef |grep 28305
root 28305 21808   0 10:24:39  pts/2  0:00 sh ps1
root 21356 21808   1 10:24:46  pts/2  0:00 grep 28305
```

记住，在用ps命令列出进程时，它无法确定该进程是运行在前台还是后台。

3.3.3 杀死后台进程

如果想杀死后台进程可以使用 kill命令。当一个进程被放到后台运行时， shell会给出一个

进程号，我们可以根据这个进程号，用 kill命令杀死该进程。该命令的基本形式为：

```
kill -signal [process_number]
```

现在暂且不要考虑其中的各种不同信号；我们会在后面的章节对这一问题进行介绍。

在杀进程的时候，执行下面的命令 (你的进程号可能会不同)并按回车键。系统将会给出相应的信息告诉用户进程已经被杀死。

```
$ kill 28305
[1]+  Terminated              ps1
```

如果系统没有给出任何信息，告诉你进程已经被杀死，那么不妨等一会儿，也许系统正在杀该进程，如果还没有回应，就再执行另外一个 kill命令，这次带上一个信号选项：

```
$ kill -9 28305
[1] + Killed                  ps1 &
```

如果用上述方法提交了一个后台进程，那么在退出时该进程将会被终止。为了使后台进程能够在退出后继续运行，可以使用 nohup命令，下面我们就介绍这一命令。

3.4 nohup命令

如果你正在运行一个进程，而且你觉得在退出帐户时该进程还不会结束，那么可以使用 nohup命令。该命令可以在你退出帐户之后继续运行相应的进程。 Nohup就是不挂起的意思(no hang up)。

该命令的一般形式为：

```
nohup command &
```

3.4.1 使用nohup命令提交作业

如果使用 nohup命令提交作业，那么在缺省情况下该作业的所有输出都被重定向到一个名为nohup.out的文件中，除非另外指定了输出文件：

```
nohup command > myout.file 2>&1
```

在上面的例子中，输出被重定向到 myout.file文件中。

让我们来看一个例子，验证一下在退出帐户后相应的作业是否能够继续运行。我们先提交一个名为ps1的日志清除进程：

```
$ nohup ps1 &
[1] 179
$ nohup: appending output to `nohup.out`
```

现在退出该shell，再重新登录，然后执行下面的命令：

```
$ ps x |grep ps1
  179  ?   S N  0:01 sh /root/ps1
  506  p2 S     0:00 grep ps1
```

我们看到，该脚本还在运行。如果系统不支持 ps x命令，使用ps -ef|grep ps1命令。

3.4.2 一次提交几个作业

如果希望一次提交几个命令，最好能够把它们写入到一个 shell脚本文件中，并用 nohup命令来执行它。例如，下面的所有命令都用管道符号连接在一起；我们可以把这些命令存入一

个文件，并使该文件可执行。

```
cat /home/accounts/qtr_0499 | /apps/bin/trials.awk |sort|lp

$ cat > quarterend
cat /home/accounts/qtr_0499 | /apps/bin/trials.awk |sort|lp
<CTRL-D>
```

现在让它可执行：

```
$ chmod 744 quarterend
```

我们还将该脚本的所有输出都重定向到一个名为 qtr.out的文件中。

```
$ nohup ./quarterend > qtr.out 2>&1 &
[5] 182
```

3.5 小结

本章中所讨论的工具主要是有关后台运行作业的。有时我们必须要对大文件进行大量更改，或执行一些复杂的查找，这些工作最好能够在系统负荷较低时执行。

创建一个定时清理日志文件或完成其他特殊工作的脚本，这样只要提交一次，就可以每天晚上运行，而且无需你干预，只要看看相应的脚本日志就可以了。 Cron和其他工具可以使系统管理任务变得更轻松。