

29 定时和延时执行，唯慢不破

更新时间：2019-07-23 15:14:24



“知识是一种快乐，而好奇则是知识的萌芽。

——培根”

内容简介

1. 前言
2. date 命令：调节时间
3. at 命令：延时执行一个程序
4. sleep 命令：休息一会
5. crontab 命令：定时执行程序
6. 总结

1. 前言

上一课多个终端和Terminator软件中，我们学习了 screen 命令和 Terminator 软件，现在你应该已经在操作终端方面很拿手了吧。

到目前为止，我们所运行的命令都是立即执行的。也就是我们按下回车键的那一刻，命令就开始执行了。

其实，在 Linux 中，命令还可以延时执行。这一课我们就来学习几个命令，可以帮助我们“稍后”执行程序。

虽说“天下武功，唯快不破”。
但有时，“慢”工也能出细活。

这一课新学的命令都涉及到时间的观念，所以我们先来学习有关系统时间的规格吧。

2. date 命令：调节时间

早期的课程中，我们已经介绍过 `date` 命令了。但那时我们只学习了 `date` 命令的最基本用法：输出当前时间。

`date` 命令的基本用法就是输入 `date`，然后回车：

```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ date
Sun May 19 10:54:58 CEST 2019
oscar@oscar-laptop:~$
```

输出信息的含义我们以前解释过了，再复习一次吧。从左到右依次是：

- Sun: Sunday 的缩写，表示“礼拜天”；
- May: 表示“五月”；
- 19: 19 日，也就是 5 月 19 日；
- 10:54:58: 10 点 54 分 58 秒；
- CEST: 所在时区。CEST 是 Central European Summer Time 的缩写，表示“欧洲中部夏令时间”，比世界标准时间（UTC）早两个小时的时区名称；
- 2019: 2019 年。

定制 `date` 的输出

`date` 命令其实挺强大的，它不仅可以输出当前时间，而且如果用 `man date` 来查看 `date` 命令的手册，就可以发现我们还可以自定义它的输出：可以选择输出哪部分信息，输出格式以及输出的顺序。

为了自定义 `date` 命令的输出，我们需要用到 `+` 号，后接其他的符号，表示不同的定制部分，这些信息最好都写在双引号间。我们来看几个例子，你就会使用了：

```
date "+%H"
```

```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ date "+%H"
13
oscar@oscar-laptop:~$
```

可以看到，输出是 13。

表示当前的小时数是 13，因为 H 是 hour 的首字母，表示“小时”。

再来一个复杂点的：

```
date "+%H:%M:%S"
```

```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ date "+%H:%M:%S"
13:01:16
oscar@oscar-laptop:~$
```

可以看到，在刚才 `+%H` 的基础上，添加了 `%M` 和 `%S`，分别表示分钟数和秒数。因为 `M` 是英语 `minute` 的首字母，表示“分钟”；`S` 是英语 `second` 的首字母，表示“秒”。

所以上命令用于显示当前的小时数，分钟数和秒数。我们用自定义的冒号来分隔三个部分信息。

当然了，我们也可以自定义其他的分隔字符，例如：

```
date "+%H时%M分%S秒"
```

```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ date "+%H时%M分%S秒"
13时02分25秒
oscar@oscar-laptop:~$
```

可以看到，经过定制，我们的显示变得更加人性化了。只有紧跟 `%` 号之后的符号会被解析，而其他如“时，分，秒”等信息则原样输出。

那我是怎么知道 `%M` 表示分钟呢？因为我看 `date` 命令的手册了啊。

因此，我也知道如何显示年份：

```
date "+现在是%Y年"
```

```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ date "+现在是%Y年"
现在是2019年
oscar@oscar-laptop:~$
```

还有其他各种输出的定制。参考手册，你可以自由发挥。

用 `date` 修改系统时间

`date` 命令还可以修改系统时间。是的，你没有看错。

修改系统时间需要使用 `root` 身份，因此我们可以这样做：

```
sudo date 10121430
```

```
oscar@oscar-laptop:~$ sudo date 10121430
[sudo] password for oscar:
Sat Oct 12 14:30:00 CEST 2019
oscar@oscar-laptop:~$ date
Sat Oct 12 14:30:01 CEST 2019
oscar@oscar-laptop:~$
```

date 命令后接的参数可以是多种形式的，此处的 10121430 表示“10 月 12 日 14 点 30 分”，没有指定年份和秒数，所以年份和秒数不变。

再次用 date 命令输出当前系统时间，可以看到已经改为了

```
Sat Oct 12 14:30:01 CEST 2019
```

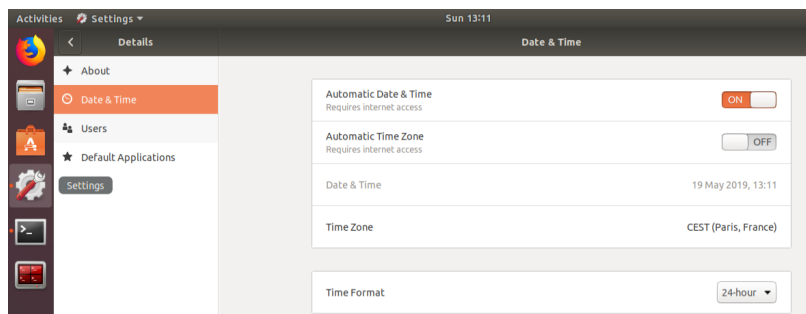
表示“星期六 十月 12日 14 点 30 分 01 秒 欧洲中部时间 2019 年”。

当然了，你可以用同样的方式，再修改回原来的系统时间。

当然，你也可以设定系统时间和网络的时间同步。

在 Settings（“设置”） -> Details（“细节”） -> Date & Time（“日期和时间”） 里，可以看到“Automatic Date & Time”（“自动的日期和时间（因为与互联网同步）”）默认是 ON（“开启”）的状态。

目前我的 Time Zone（时区）是选择的 CEST（Paris, France）（“巴黎，法国”）。如果在国内，那一般是 Beijing, China（北京，中国）吧。



你可以也把“Automatic Time Zone”（“自动的时区”注：与互联网同步）从 OFF（“关闭”）改为 ON 状态。Linux 系统会和指定的时间服务器同步，就不用担心时间不对了。当然，需要在联网的情况下才可以。

3. at 命令：延时执行一个程序

你想要延时执行一个程序（所有的命令说到底都是程序）吗？没问题。我们可以用 at 命令来设定一个程序的执行时间。

at 是英语“在...时刻”的意思。

注意：at 命令只能让程序执行一次。

如果你要定时重复执行程序，那就要用 crontab 命令，本课的最后一节我们会学到。

也许你的 Ubuntu 里没有 at 命令，可以用

```
sudo apt install at
```

来安装。

at 命令有几种用法，我们先来看第一种：

在指定时刻执行程序

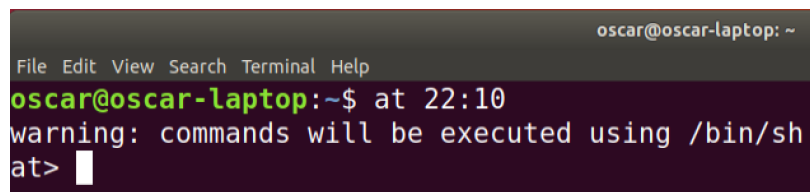
在这种用法下，at 命令的使用顺序如下：

1. 先用 at 命令后接想要程序执行的确定时刻
2. 再输入你想要在以上指定时刻执行的命令

例如：

```
at 22:10
```

按回车，

A terminal window titled 'oscar@oscar-laptop: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'oscar@oscar-laptop:~\$'. The user enters 'at 22:10'. The terminal shows a warning: 'warning: commands will be executed using /bin/sh'. The prompt changes to 'at>' with a cursor.

终端会显示 `at>`，提示你输入要在 22 点 10 分执行的命令。

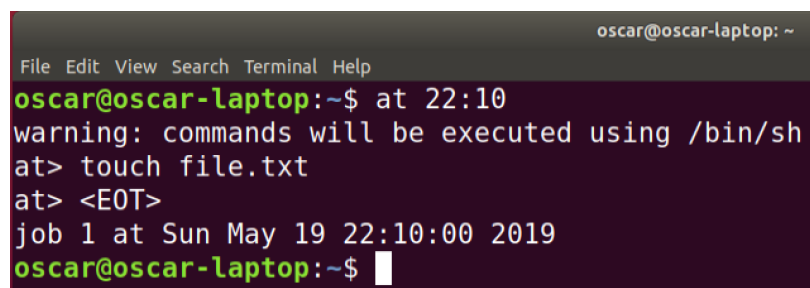
我们可以输入

```
touch file.txt
```

表示我们想要在 22 点 10 分创建一个文件，名叫 file.txt。

然后，回车。at 命令会继续显示 `at>`，提示你输入在指定时刻想要执行的其他命令。你可以继续输入，也可以就此打住。

那么怎么结束输入并退出 at 命令呢？可以使用 Ctrl + D 组合键，at 会显示，是“End Of Transmission”（表示“传输结束”）的缩写。

A terminal window showing the continuation of the previous session. The prompt is 'at>'. The user enters 'touch file.txt'. The prompt changes to 'at>' again. The user presses Ctrl+D, which is represented as '<EOT>'. The terminal shows 'job 1 at Sun May 19 22:10:00 2019' and then returns to the shell prompt 'oscar@oscar-laptop:~\$'.

然后会打印出一句话：

```
job 1 at Sun May 19 22:10:00 2019
```

上面这句话表示：

- job: 英语“工作，任务”的意思，表示创建了一个任务；
- 1: 是 job 的编号。表示第 1 号任务；
- at: “在...时刻”，也正是 at 命令的作用所在；
- Sun: Sunday 的缩写。英语“星期日”的意思；
- May: 英语“五月”的意思；
- 19: 19 日。即 5 月 19 日；
- 22:10:00: 命令执行的时刻。22 点 10 分 00 秒，也就是我们用 at 22:10 指定的时间；
- 2019: 2019 年。

之后，到了 22 点 10 分，就会创建 file.txt 这个文件了。

如果说我们不想要在今天的 22 点 10 分执行指定命令，而想要在明天的 22 点 10 分执行，怎么做呢？可以这样：

```
at 22:10 tomorrow
```

tomorrow 是英语“明天”的意思。

那我要在 2019 年 12 月 10 日的 22 点 10 分执行呢？

```
at 22:10 12/10/19
```

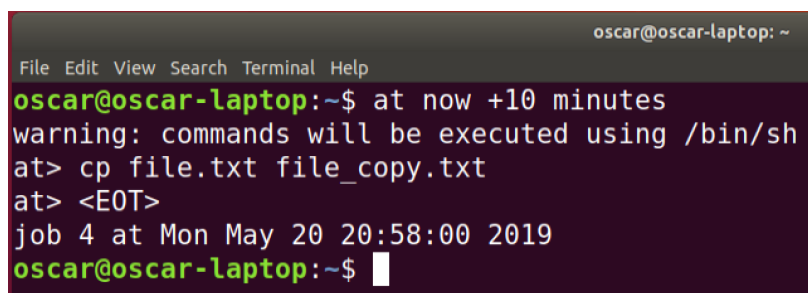
日期的格式是美国日期的格式，所以是 12/10/19，依次是“月/日/年”。

在指定间隔之后执行程序

at 还有第二种用法，就是在指定时间间隔之后执行程序。

例如，我要在 10 分钟之后执行指定程序：

```
at now +10 minutes
```



```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ at now +10 minutes  
warning: commands will be executed using /bin/sh  
at> cp file.txt file_copy.txt  
at> <EOT>  
job 4 at Mon May 20 20:58:00 2019  
oscar@oscar-laptop:~$
```

now 表示“现在”，+10 minutes 表示“10 分钟之后”，所以就是“现在开始的 10 分钟之后执行”。

这里我随便写了一个命令：

```
cp file.txt file_copy.txt
```

所以，job 2 被创建了，就是在现在开始的 10 分钟后会执行“拷贝 file.txt 文件到 file_copy.txt”。

当然了，不止 minutes 这个关键字可以使用，我们列出几乎所有可以使用的关键字：

- minutes: 表示“分钟”；
- hours: 表示“小时”；
- days: 表示“天”；
- weeks: 表示“星期”；
- months: 表示“月”；
- years: 表示“年”。

例如：

```
at now +7 weeks
```

表示在距今 7 个星期之后执行。

atq 和 atrm 命令：列出和删除正在等待执行的 at 任务

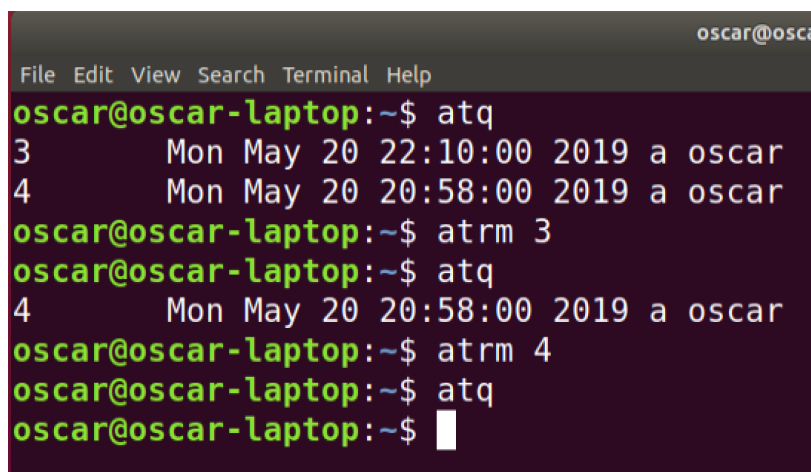
每次我们用 at 命令指定延时执行的命令，at 都会为其分配一个 job 编号，比如我们上面的两个例子，分别被分配了 3 和 4 的编号。3 号是 `touch file.txt`，4 号是 `cp file.txt file_copy.txt`。

因为 1 和 2 已经被我昨天创建的两个 at 任务占用了，所以今天新建的两个 at 任务的编号是 3 和 4。

atq 命令可以列出正等待执行的 at 任务。q 是英语 queue 的首字母，表示“队列”，会列出 at 命令的任务队列。

atrm 命令可以删除正在等待执行的 at 任务。rm 是英语 remove 的首字母，remove 表示“删除”。后接 at 任务的编号。例如 1, 2, 3, 4... 这样的编号。

我们来实际操作一下：

A terminal window with a dark background and light green text. The prompt is 'oscar@oscar-laptop:~\$'. The user enters 'atq', and the output shows two tasks: task 3 scheduled for Mon May 20 22:10:00 2019, and task 4 scheduled for Mon May 20 20:58:00 2019. The user then enters 'atrm 3', and the prompt returns. The user enters 'atq' again, and the output shows only task 4. The user then enters 'atrm 4', and the prompt returns. Finally, the user enters 'atq' again, and the prompt returns without any output, indicating no tasks are left in the queue.

```
oscar@oscar-laptop:~$ atq
3      Mon May 20 22:10:00 2019 a oscar
4      Mon May 20 20:58:00 2019 a oscar
oscar@oscar-laptop:~$ atrm 3
oscar@oscar-laptop:~$ atq
4      Mon May 20 20:58:00 2019 a oscar
oscar@oscar-laptop:~$ atrm 4
oscar@oscar-laptop:~$ atq
oscar@oscar-laptop:~$
```

可以看到，我们用 atq 命令列出了 at 任务，有两个，就是我们上面设定的两个延时任务，分别会在 22 点 10 分和 20 点 58 分执行。

然后我们用 atrm 一次性删除了这两个任务，因此就不会执行任何任务了。

4. sleep 命令：休息一会

其实，我们可以用分号隔开多个命令，使之一个接一个执行。这跟之前学过的管道不一样，管道是前一个命令的输出作为后一个命令的输入。用分号隔开的各个命令并没有关联。例如：

```
touch file.txt ; rm file.txt
```

上面用分号隔开的两句命令的作用：创建 file.txt 文件，然后删除之。

我们在两句命令之前可以插入一定的暂停等待时间，用 sleep 命令。

sleep 是英语“睡觉，睡眠”的意思。

```
touch file.txt ; sleep 10 ; rm file.txt
```

上面的三句命令分别表示：

- touch file.txt : 创建文件 file.txt;
- sleep 10 : 暂停 10 秒;
- rm file.txt : 删除 file.txt。

默认地，sleep 后面的数值表示秒数。但我们也可以指定其表示分钟或小时或天：

- m: minute 的缩写，表示“分钟”;
- h: hour 的缩写，表示“小时”;
- d: day 的缩写，表示“天”。

例如：

```
touch file.txt ; sleep 15m ; rm file.txt
```

上面的三句命令会依次执行：创建 file.txt 文件，暂停 15 分钟，删除 file.txt 文件。

&& 和 || 符号

上面我们讲了 sleep 命令的用法，也提到了分号的作用：可以用于分隔多个命令，使多个命令可以写在一行里，然后依次执行。分号前的一个命令执行完，就会执行分号后的一个命令。但是分号前的命令执行成功与否并不会影响后面的命令。不管怎样，分号前后的命令都会执行。

我们来学习两个很有用的符号：&& 和 ||（两竖）。

这两个在编程语言里一般称为“逻辑与”和“逻辑或”符号，在命令行里又会起到什么作用呢？

&& 及 || 和分号一样，用于分隔两个命令，使得命令依次执行，貌似和分号类似，但是有区别。

简单说来，就是：

- &&: && 号前的命令执行成功，才会执行后面的命令。
- ||: || 号前的命令执行失败，才会执行后面的命令。
- 分号: 不论分号前的命令执行成功与否，都执行分号后的命令。前后命令之间没有相关性。

这三个符号非常有用，可以提高我们命令行的效率和丰富程度。

5. crontab 命令：定时执行程序

crontab 命令是 Linux 中很常用也很强大的一个命令。我们可以用它来定时执行程序。

前面我们学过的 at 命令，只能执行某个（或某几个）命令一次。

但是 crontab 却可以重复执行命令。例如：每小时，每分钟，每天，每星期等等。

安装 crontab

一般来说，Ubuntu 下是默认安装了 crontab 程序的。不过有的 Linux 发行版可能没装 crontab。

crontab 的安装，举 yum 系列和 apt 系列两个例子来看看吧：

在 CentOS（Fedora 下也类似）中安装 Crontab

```
sudo yum install vixie-cron crontabs # 安装 Crontab
chkconfig crond on                  # 设为开机自启动
service crond start                  # 启动
```

在 Debian（Ubuntu 是 Debian 一族的）中安装 Crontab

```
sudo apt install cron                # 大部分情况下 Debian 都已安装
service cron restart 或者 restart cron # 重启 crontab
```

Ubuntu 下启动、停止和重启 crontab：

```
service cron start    # 启动
service cron stop     # 停止
service cron restart  # 重启
```

前期配置工作

在学习 crontab 之前，我们需要先做一些配置。我们来修改 .bashrc 这个文件，之前的课程我们已经学习过了，这是 Bash 这个 Shell（简单地说就是控制我们当前终端的程序）的配置文件。

也没什么太大的修改，就是想让 Nano 这个文本编辑器成为我们默认的文本编辑器。因为 Ubuntu 系统一般默认的文本编辑器是 vi，而 vi 是比较难的文本编辑器，我们之后的课程会专门学习 vi 和 vim。我们目前只会 Nano 这个文本编辑器，所以暂时不去碰 vi。

我们要做的就是把这一句话加入 .bashrc 文件：

```
export EDITOR=nano
```

用我们学过的重定向的知识，我们可以这样做：

```
echo "export EDITOR=nano" >> ~/.bashrc
```

这样，就把 `export EDITOR=nano` 这句话追加到了家目录下的 .bashrc 文件最后。

```
File Edit View Search Terminal Help
oscar@oscar-laptop: ~
oscar@oscar-laptop:~$ echo "export EDITOR=nano" >> ~/.bashrc
oscar@oscar-laptop:~$ tail .bashrc
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
export EDITOR=nano
oscar@oscar-laptop:~$
```

可以看到，`export EDITOR=nano` 这句话被追加到了 `.bashrc` 文件的末尾。editor 是英语“编辑器”的意思。

运行以下命令使改动立即生效，不然须要重开一个终端或者重新登录才能生效。

```
source ~/.bashrc
```

OK，准备工作做好了，我们来认识 `crontab` 吧。

crontab 是什么呢？

`crontab` 其实是一个命令，用来读取和修改名为 `crontab` 的文件。这个 `crontab` 文件包含了你要定时执行的程序列表，也包含了执行的时刻。

实际上，有两个命令，一个叫 `crontab`，一个叫 `cron`。`crontab` 用于修改 `crontab` 文件，`cron` 用于实际执行定时的程序。

其实，`cron` 这个单词来源于希腊语 $\chi\rho\acute{o}\nu\omicron\varsigma$ (chronos)，原意是“时间”。

`crontab` 命令如何使用呢？

有三个参数要了解：

- `-e`：修改 `crontab` 文件
- `-l`：显示 `crontab` 文件
- `-r`：删除 `crontab` 文件

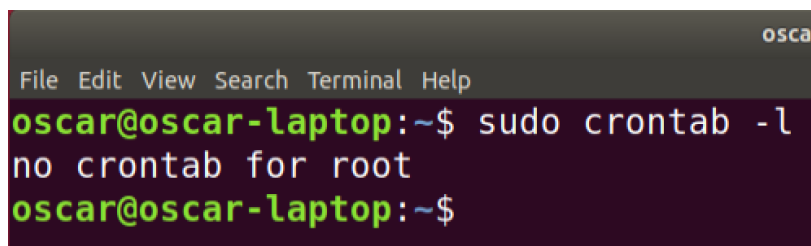
我们首先尝试显示 `crontab` 文件：

```
crontab -l
```

```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ crontab -l
no crontab for oscar
oscar@oscar-laptop:~$
```

可以看到，显示了“no crontab for oscar”，表示“用户 `oscar` 暂时没有任何 `crontab` 文件”。

其实，每个用户有自己的 crontab 文件，比如 root 用户也有自己的 crontab 文件。可以用 `sudo crontab -l` 来看：



```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ sudo crontab -l
no crontab for root
oscar@oscar-laptop:~$
```

可以看到，root 用户暂时也没有 crontab 文件。

既然我们的用户暂时还没有 crontab 文件，那么我们就来创建咯。

可以用：

```
crontab -e
```

命令，之前我们说过 `crontab -e` 用于修改 crontab 文件。那既然文件不存在，就新建一个，用什么编辑器编辑此 crontab 文件呢？就是用我们指定的 Nano 文本编辑器。

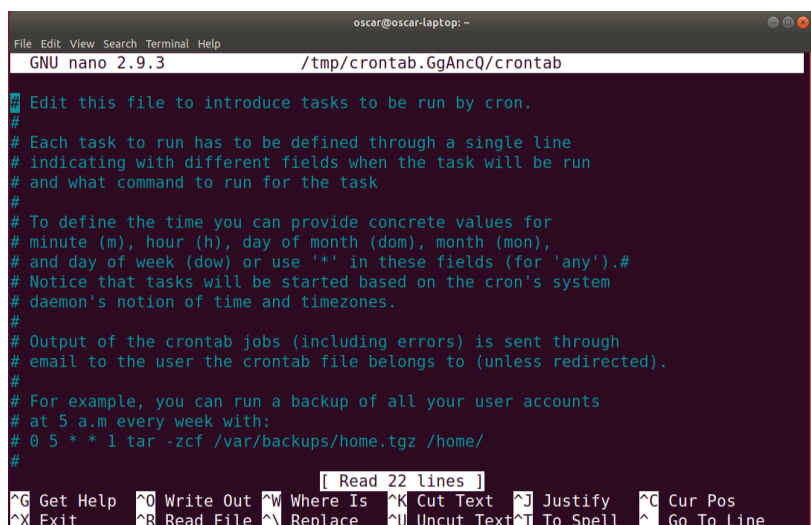
如果你之前正确配置了 `.bashrc` 文件，也就是在文件最后添加了 `export EDITOR=nano` 这一行的话，那么运行 `crontab -e` 命令之后，会用 Nano 编辑器来打开你的 crontab 文件。如果没有正确配置 `.bashrc`，那么将会由默认的 vi 编辑器来打开。但是暂时我们还没有讲过 vi 怎么用，所以你可以退出 vi。输入 q，然后回车。

再重新配置，直到运行 `crontab -e` 命令，终端用 Nano 打开 crontab 文件为止。

当然了，如果你已经会用 vi 或 vim 或 emacs，那么就不必配置用 Nano 了。

修改 crontab 文件

上面，我们运行了 `crontab -e` 命令。我们的 crontab 文件刚被创建，所以没什么实际内容，如下图：



```
oscar@oscar-laptop: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 /tmp/crontab.GgAncQ/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/
#

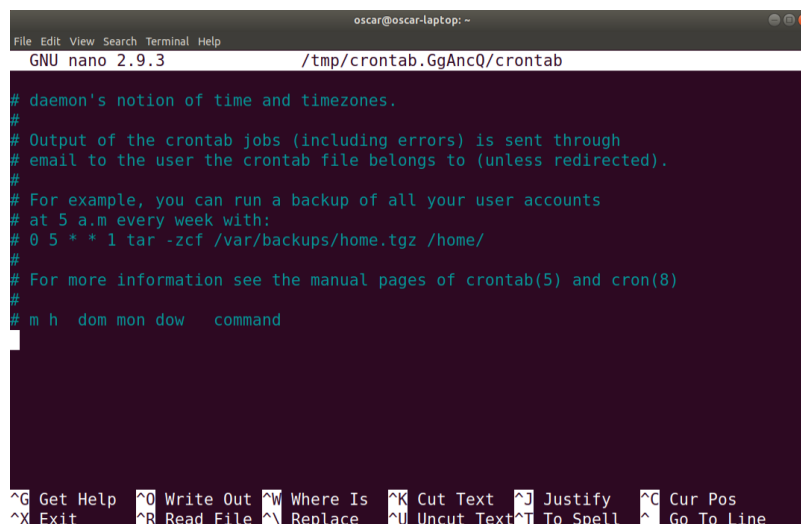
[ Read 22 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

可以看到，我们的 crontab 文件位于 `/tmp/crontab.GgAncQ/` 目录中（我承认，`crontab.GgAncQ` 这个名字很奇怪，不要在意它。你的目录名应该和我不一样）。

这个 crontab 文件目前都是 # 开头的注释，是说明 crontab 应该如何编辑的文档。

我们将光标移动到这个文件的最后一行，可以看到这样的一句注释的话：

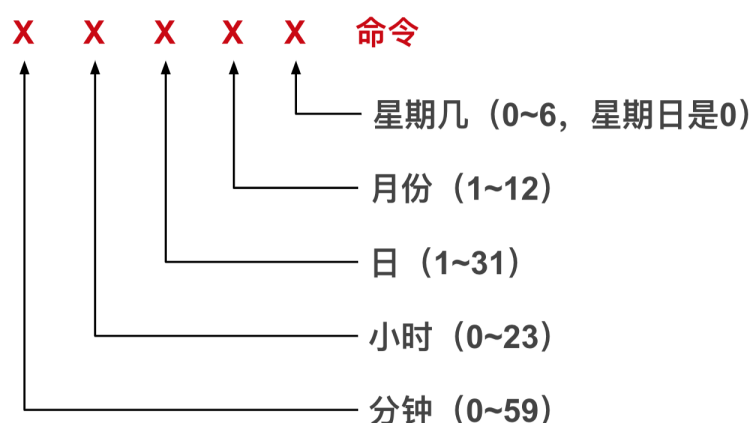
```
# m h dom mon dow command
```



其实，这句话给出了 crontab 中的每行指令的书写格式：

- m: minute 的缩写，表示“分钟”；
- h: hour 的缩写，表示“小时”；
- dom: day of month 的缩写，表示“一个月的哪一天”；
- mon: month 的缩写，表示“月份”；
- dow: day of week 的缩写，表示“星期几”；
- command: 英语“命令”的意思，表示需要定时执行的命令。

用下图表述：



所以每一行的写法很清楚：你须要先写定时是在什么时候，然后在最后写上定时执行什么命令。

理解 crontab 不是那么容易，但也并没有很难。我们会用好几个例子来帮助大家了解。

上图中用 X 表示的五个区域分别是“分钟，小时，日，月份，星期几”，这五个区域要么用数字加符号填充，要么写上一个星号 (*)，表示任何值。

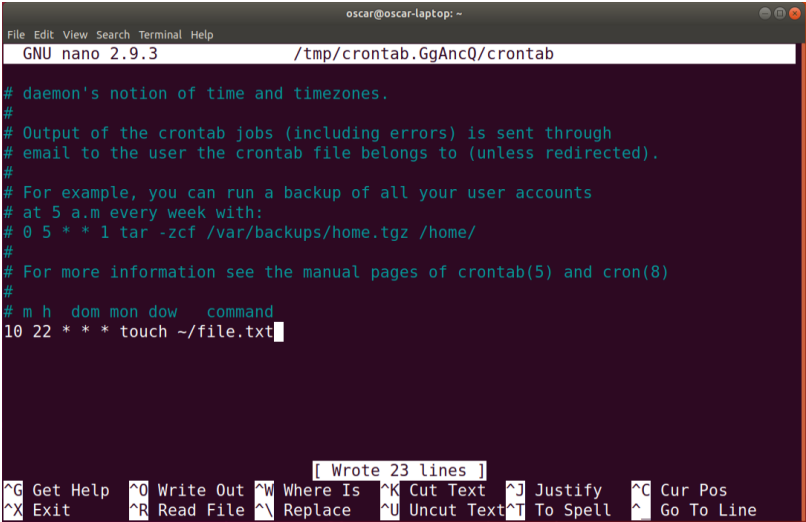
例如：

我希望每天的 22 点 10 分都在家目录下创建 file.txt 文件。可以在 crontab 文件里写入：

```
10 22 * * * touch ~/file.txt
```

10 表示分钟，22 表示小时，其他三个区域是 *，最后的命令是 `touch ~/file.txt`。

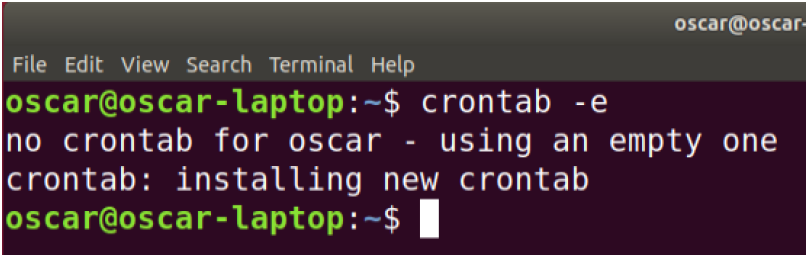
路径最好用绝对路径，因为你并不能确定 cron 命令执行这些语句的时候是在哪个目录。不过 ~ 就是代表当前用户的家目录了。



然后，保存并退出 Nano 编辑器。如果语法正确的话，crontab 会显示：

```
crontab: installing new crontab
```

意思是：“安装新的 crontab 文件”。



现在，既然已经设置好了。那么 file.txt 文件将会在每天的 22 点 10 分被创建于家目录下（如果 file.txt 文件不存在的话）。

我们看几个例子：

Crontab	意义
47 **** command	每个小时的 47 分都执行 command 命令，也就是 00 点 47 分, 01 点 47 分, 02 点 47 分等等
0 0 ** 1 command	每个礼拜一的凌晨都执行 command 命令
30 5 1-15 ** command	每个月的 1 ~ 15 日的 5 点 30 分都执行 command 命令
0 0 ** 1,3,4 command	每个礼拜一，礼拜三，礼拜四的凌晨都执行 command 命令
0 */2 *** command	每 2 个小时的整点（0，2，4，6，等等）都执行 command 命令
*/10 *** 1-5 command	每个礼拜一到礼拜五的每个 10 的倍数的分钟（0，10，20，30，等等）都执行 command 命令

你可以自己设计出很多很多不同的组合用法，来实现你的定时程序。

```
crontab -r
```

用于删除 crontab 文件。

```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ crontab -l  
# 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  
10 22 * * * touch ~/file.txt  
oscar@oscar-laptop:~$
```

可以看到，我们运行 `crontab -l`，此时我们有一个 `crontab` 文件，显示的内容就是我们之前用 Nano 编辑器填写的。

然后我们用 `crontab -r` 来删除 `crontab` 文件，则再用 `crontab -l` 命令时，就显示“no crontab for oscar”，就是又不存在 `crontab` 文件了，也就没有定时执行的任务了。

```
oscar@oscar-laptop:~$ crontab -r  
oscar@oscar-laptop:~$ crontab -l  
no crontab for oscar  
oscar@oscar-laptop:~$
```

小结

1. `date` 命令可以显示系统时间，可以按自定义格式显示，也可以修改系统时间。
2. `at` 命令可以延时执行程序。但是它只能执行一次。
3. 分号、`&&` 和 `||` 都可以用来连接多个命令：用于依次执行前后命令。
4. `sleep` 命令用于使前后两个命令执行之间间隔一定时间。
5. `crontab` 用于定时执行程序。例如：每天 17 点 12 分，每礼拜二和礼拜三在 12 点，每个月的 7 号，等等。我们用 `crontab -e` 来修改 `crontab` 程序。

今天的课就到这里，一起加油吧！

欢迎在这里发表留言，作者筛选后可公开显示



目前暂无任何讨论