Crontab：<https://linuxtools-rst.readthedocs.io/zh_CN/latest/tool/crontab.html>

### 命令

crontab [-u user] file

crontab [-u user] [ -e | -l | -r ]

-u user：用来设定某个用户的crontab服务；

file : file是命令文件的名字，表示将file作为crontab的任务列表文件并载入crontab；若命令行没有指定该文件，则会弹出一个vim编辑界面，你可以将你的输入载入到crontab；

-e : 编辑某个用户的crontab文件，若不指定用户，则表示编辑当前用户的crontab文件

-l : 显示某个用户的crontab文件，若不指定用户，则表示显示当前用户的crontab 文件内容；

-r : 从 /var/spool/cron目录中（Linux）|| /usr/lib/cron/tabs/用户名（Mac） 删除某个用户的crontab文件，如果不指定用户，则默认删除当前用户的crontab文件。

crontab命令用于安装、删除和列出所有定时任务列表。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。

每个用户的crontab文件真实保存在/var/spool/cron目录中（Linux）|| /usr/lib/cron/tabs/用户名（Mac）。在/var/spool/cron目录中（Linux）|| /usr/lib/cron/tabs/用户名（Mac） 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，假设用户yangwenju，需要创建自己的crontab文件，首先可以用编辑器建一个新文件，然后将需要运行的命令和定期执行的时间写入。，暂且保存为 /tmp/test.cron，执行：

crontab /tmp/test.cron

这样一个crontab文件就创建好了，转到/var/spool/cron目录中（Linux）|| /usr/lib/cron/tabs/用户名（Mac）目录查看，发现多了一个yangwenju文件，该文件就是用户yangwenju的crontab文件。该命令把test.cron中的内容导入到yangwenju的crontab文件中了。若要改变crontab文件内容，还是需要重新编辑原来的文件，再使用crontab命令安装。

### crontab服务的重启关闭，开启

Mac系统下

sudo /usr/sbin/cron start

sudo /usr/sbin/cron restart

sudo /usr/sbin/cron stop

sudo /usr/sbin/cron status

日志文件：linux下cron的日志文件位于：/var/log

查看crontab是否开启

**service crond status**

### 时间表示

分 时 日 月 星期 要运行的命令

* 第1列分钟0～59
* 第2列小时0～23（0表示子夜）
* 第3列日1～31
* 第4列月1～12
* 第5列星期0～7（0和7表示星期天）
* 第6列要运行的命令

\*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，表示一个集合

“/”——字符用来指定一个值的的增加幅度.比如在“秒”字段中设置为"0/15"表示"第0, 15, 30,和45秒"。而"5/15"则表示"第5, 20, 35,和50".

在'/'前加"\*"字符相当于指定从0秒开始.每个字段都有一系列可以开始或结束的数值。对于“秒”和“分”字段来说，其数值范围为0到59，对于“小时”字段来说其为0到23,对于“日”字段来说为1到31,而对于“月”字段来说为1到12。"/"字段仅仅只是帮助你在允许的数值范围内从开始"第n"的值。

例如：\* \* \*/3 \* \* command

会在每个月的1、4、7、10号执行

### 注意事项

* 1. 环境变量

有时我们创建了一个crontab，但是这个任务却无法自动执行，而手动执行这个任务却没有问题，这种情况一般是由于在crontab文件中没有配置环境变量引起的。

在crontab文件中定义多个调度任务时，需要特别注环境变量的设置，因为我们手动执行某个任务时，是在当前shell环境下进行的，程序当然能找到环境变量，而系统自动执行任务调度时，是不会加载任何环境变量的，因此，就需要在crontab文件中指定任务运行所需的所有环境变量，这样，系统执行任务调度时就没有问题了。

注意以下三点：

* 脚本中涉及文件路径时写全局路径；
* 脚本执行要用到java或其他环境变量时，通过source命令引入环境变量，如:

cat start\_cbp.sh

!/bin/sh

source /etc/profile

export RUN\_CONF=/home/d139/conf/platform/cbp/cbp\_jboss.conf

/usr/local/jboss-4.0.5/bin/run.sh -c mev &

* 当手动执行脚本OK，但是crontab死活不执行时,很可能是环境变量惹的祸，可尝试在crontab中直接引入环境变量解决问题。如:

0 \* \* \* \* . /etc/profile;/bin/sh /var/www/java/audit\_no\_count/bin/restart\_audit.sh

* 1. 注意清理系统用户的邮件日志

每条任务调度执行完毕，系统都会将任务输出信息通过电子邮件的形式发送给当前系统用户，这样日积月累，日志信息会非常大。

可以设置忽略日志输出：

0 \*/3 \* \* \* /usr/local/apache2/apachectl restart >/dev/null 2>&1

“/dev/null 2>&1”表示先将标准输出重定向到/dev/null，然后将标准错误重定向到标准输出，由于标准输出已经重定向到了/dev/null，因此标准错误也会重定向到/dev/null，这样日志输出问题就解决了。

* 1. 系统级任务调度和用户级任务调度

系统级任务调度主要完成系统的一些维护操作，用户级任务调度主要完成用户自定义的一些任务，可以将用户级任务调度放到系统级任务调度来完成（不建议这么做），但是反过来却不行，root用户的任务调度操作可以通过”crontab –uroot –e”来设置，也可以将调度任务直接写入/etc/crontab文件，需要注意的是，如果要定义一个定时重启系统的任务，就必须将任务放到/etc/crontab文件，即使在root用户下创建一个定时重启系统的任务也是无效的。

* 1. 其他

新建的cron job 至少2分钟后才会执行，如果重启cron则会马上执行；

* 1. flock 避免同时起多个进程

比如1分钟执行一次，但任务1分钟并没有执行完，此时下一分钟会再起一个进程。

flock 文件锁实现任务锁定，解决冲突

flock [-sxon][-w #] file [-c] command

-s, --shared: 获得一个共享锁

-x, --exclusive: 获得一个独占锁

-u, --unlock: 移除一个锁，通常是不需要的，脚本执行完会自动丢弃锁

-n, --nonblock: 如果没有立即获得锁，直接失败而不是等待

-w, --timeout: 如果没有立即获得锁，等待指定时间

-o, --close: 在运行命令前关闭文件的描述符号。用于如果命令产生子进程时会不受锁的管控

-c, --command: 在shell中运行一个单独的命令

-h, --help 显示帮助

-V, --version: 显示版本

文件锁使用独占锁，如果锁定则失败不等待。参数为-xn

|  |
| --- |
| 50 9 \* \* \* flock **-xn** /home/work/hfsfd3/current/changelog/crontab/serviceTodoSendQx.js **-c** 'cd /home/work/hfsfd3/current/changelog && NODE\_ENV=test ~/.nvm/versions/node/v10.16.3/bin/node crontab/serviceTodoSendQx.js >> logs/serviceTodoSendQx.log 2>&1' |

死锁：

进程不释放锁，导致下次的脚本运行无法进行

|  |
| --- |
| \* \* \* \* \* **timeout -s SIGINT 10**  flock -xn /tmp/lock -c "~/.local/node.js/current/bin/node /home/work/test.js >> /home/work/test.log" |

timeout [OPTION] DURATION COMMAND [ARG]..

OPTION:

-s : 后跟string or number, SIGINT 或 9

DURATION : 超时时间，默认单位秒

### crontab 坑

* 1. \* \*/1 \* \* \*

看似是每小时执行一次，但实际上是**每分钟**执行一次！

因为分钟要求的是每分钟执行，而小时却要求每一个小时执行，这2个分明是冲突的时间策略。最终以分钟为准，所以它是每分钟执行一次。