CRONTAB 视频学习笔记

1.

4.1.1 Crond 是什么？

Crond 是 linux 系统中用来定期执行命令或指定程序任务的一种服务。一般情况下，安装完操作系统之后，默认便会启动 Crond 任务调度服务，在我们前面的系统安装及开机启动优化设置中，我们也设置保留了 Crond 开机自启动。Crond 服务会定期（默认每分钟检查一次）检查系统中是否有要执行的任务工作。如果有，便会根据其预先设定的定时任务规则自动执行该定时任务工作。

2. 查看服务

Chkconfig –list

3.

4.1.3.3 linux 系统下定时任务软件种类

严格的说，linux 系统下的定时任务软件还真不少，例如：at，crontab，anacron。

◆at：适合仅执行一次就结束的调度命令，例如：某天晚上需要处理一个任务，仅仅是这一天的晚上，属于突发性的工作任务。要执行 at 命令，还需要启动一个名为 atd 的服务才行，在老男孩的工作中从来都不会有需求用这个。因此，建议大家不要深入研究了，到此我们讲解这里为止即可。

◆crontab：正如前面所说这个命令可以周期性的执行任务工作，例如：每五分钟做一次服务器时间同步。要执行 crontab 这个命令，也需要启动一个服务 crond 才行，这个命令是老男孩在生产工作中最常用到的命令，请大家务必掌握了。

◆anacron：这个命令主要用于非 7*24 小时开机的服务器准备的，anacron 并不能指定具体时间执行任务工作，而是以天为周期或者在系统每次开机后需要执行的任务工作。它会检测停机期间应该进行，但是并没有进行的 crontab 任务工作，并将该任务执行一遍。

提示：crontab：是生产工作中的重要的应用，其他的很少使用。

```
crontab: usage error: unrecognized option
usage:  crontab [-u user] file
        crontab [ -u user ] [ -i ] { -e | -l | -r }
                (default operation is replace, per 1003.2)
        -e      (edit user's crontab)
        -l      (list user's crontab)
        -r      (delete user's crontab)
        -i      (prompt before deleting user's crontab)
```

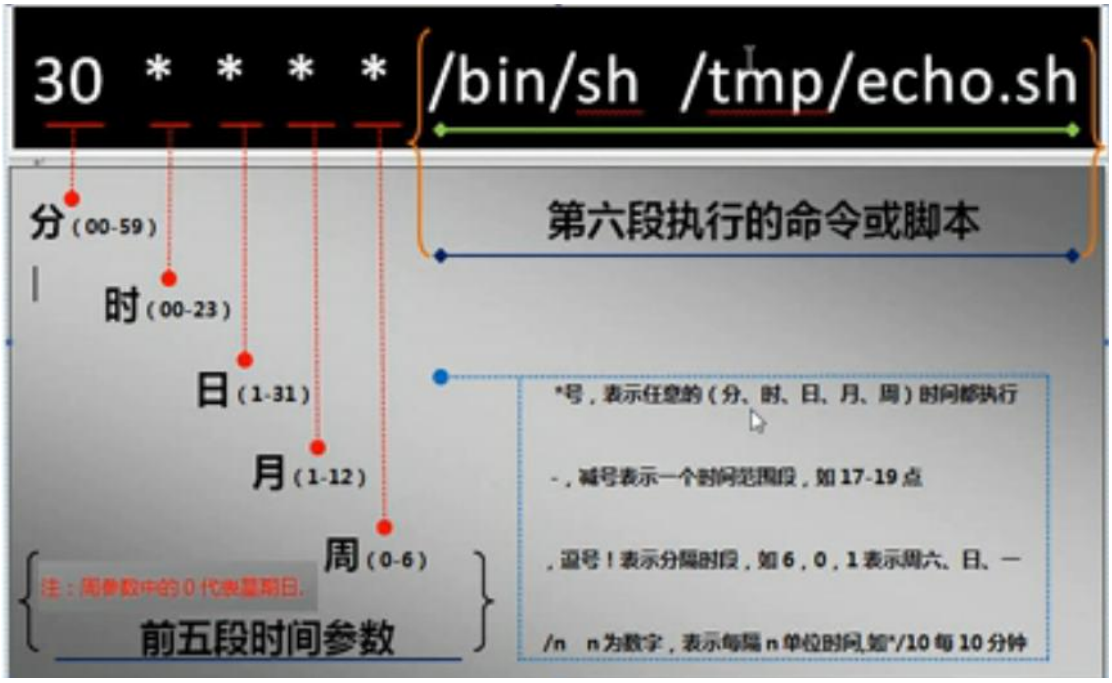
通过 crontab 我们可以在固定的间隔时间执行指定的系统指令或 shell script 脚本。时间间隔的单位可以是分钟、小时、日、月、周及以上的任意组合。crond 服务通过 crontab 命令可以很容易的实现周期性的日志分析或数据备份等运维场景工作。

文件	说明
/etc/cron.deny	该文件中所列用户不允许使用 crontab 命令
/etc/cron.allow	该文件中所列用户允许使用 crontab 命令，优先于/etc/cron.deny。
/var/spool/cron/	所有用户 crontab 文件默认都存放在此目录,文件名以用户名命名

4.2.5 指令的使用格式

默认情况下，当用户建立定时任务规则后，该规则记录对应的配置文件会存在于/var/spool/cron 中，其 crontab 文件对应的文件名与用户名一致，如：root 用户的定时任务配置文件为/var/spool/cron/root。

Crontab 定时任务的书写格式也很简单，一共分为 6 段（空格分隔，系统的定时任务则/etc/crontab 分为 7 段），前五段为时间设定段，第六段为所要执行的命令或脚本段。



*	*号，表示任意时间都，也是“每”的意思，举例：如 00 23 * * * cmd 表示每月每周每日的 23:00 都执行 cmd 任务。
-	减号，表示分隔符，表示一个时间范围段，如 17-19 点，每小时的 00 分执行任务。00 17-19 * * * cmd。就是 17,18,19 点整点分别执行的意思。
,	逗号，表示分隔时段的意思。30 17,18,19 * * * /bin/sh /scripts/oldboy.sh 表示。每天 17、18 和 19 点的半点时刻执行/scripts/oldboy.sh 脚本。也可以和“-”结合使用，例如：30 3-5,17-19 * * * /bin/sh /scripts/oldboy.sh
/n	n 代表数字，即“每隔 n 单位时间”，例如：每 10 分钟执行一次任务可以写成*/10 * * * * cmd, 其中，*/10,*的范围是 0-59，因此，也可以写成 0-59/10。

windows 的时间同步：

linux 系统的时间同步方法

手动同步：

```
[root@oldboy ~]# /sbin/ntpdate time.windows.com
 5 May 12:00:42 ntpdate[2844]: step time server 65.55.21.23 offset -41960.869371 sec
```

定时任务：

```
[root@centos ~]# echo '#time sync by oldboy at 2010-2-1' >>/var/spool/cron/root
[root@centos ~]# echo '*/5 * * * * /sbin/ntpdate time.windows.com >/dev/null 2>&1' >>/var/spool/cron/root
```

#→这个命令其实就是写一个定时任务，相当于执行 crontab -e 然后加入内容：*/5 * * * * /usr/sbin/ntpdate time.windows.com >/dev/null 2>&1 保存退出。有关 crontab 定时任务后文有详细讲解，大家不必在这里纠缠。

补充一个定时清理优化任务，find /var/spool/clientmqueue/ -type f -mtime +30|xargs rm -f

① 30 3,12 * * * /bin/sh /scripts/oldboy.sh

在本例中，第一列为 30，表示 30 分钟，第二列为 3,12，这代表 3 点及 12 点，此定时任务的意思是**每天凌晨 3 点和中午 12 点的半点时刻**（或描述为每天凌晨 3：30 和中午 12：30）执行/scripts/oldboy.sh 脚本。

② 30 */6 * * * /bin/sh /scripts/oldboy.sh

在本例中，第一列为 30，表示 30 分钟，第二列*/6 代表每 6 个小时，也相当于 6,12,18,24 的作用。此定时任务的意思是每隔 6 个小时的半点时刻执行/scripts/oldboy.sh 脚本任务。

③ 30 8-18/2 * * * /bin/sh /scripts/oldboy.sh

在本例中，其中的第一列为 30，表示 30 分钟，第二列 8-18/2 代表在 早晨 8 点到 下午 18 点之间每隔 2 小时，也相当于 8,10,12,14,16,18 单独列出的作用。

那么，此定时任务的意思就是早晨 8 点到 下午 18 点之间每隔 2 小时的 30 分时刻执行/scripts/oldboy.sh 脚本任务。

④ 30 21 * * * /application/apache/bin/apachectl graceful

上面的例子表示每晚的 21:30 重启 apache。

⑤ 45 4 1,10,22 * * /application/apache/bin/apachectl graceful

上面的例子表示每月 1、10、22 日的凌晨 4：45 分重启 apache。

⑥ 10 1 * * 6,0 /application/apache/bin/apachectl graceful

上面的例子表示每周六、周日的凌晨 1：10 分重启 apache。

⑦ 0,30 18-23 * * * /application/apache/bin/apachectl graceful

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。

提示：最后一次执行任务是 23:30 分

⑧ 00 */1 * * * /application/apache/bin/apachectl graceful

上面的例子表示每隔一小时整点重启 apache

提示：以上结果是不规范的，也是不对的。大家想想为什么？

以上的定时任务的第一列为*，表示每分都执行任务即表示晚上 23 点到早上 7 点之间，每小时的每分都重启 apache，很可怕吧。

⑨ 00 11 * 4 1-3 /application/apache/bin/apachectl graceful

上面的例子表示 4 月的每周一到周三的 11 点重启 apache

强调：周和日尽量不要同时用，否则可能达不到想要的效果。

例：每周五，5 月 5 日上午 9:00 去老男孩培训上课。

00 09 5 5 5 任务

```
tar zcf backup_$(date +%F-%H-%M).tar.gz /server/scripts
```

4.3.1.2 为定时任务规则加必要的注释

加必要注释：写定时任务规则时尽可能的加上注释（最好是英文注释），这是个好的习惯和规范。如：什么人，什么时间，因为谁（需求方），做了什么事。如果这些都标记清楚了，这样其他的运维人员（当然是你的同事）可以很容易理解任务的信息，从而提升工作效率。

4.3.1.3 执行 shell 脚本任务前加/bin/sh

执行定任务时，如果是执行脚本，请尽量在脚本前面带上/bin/sh 命令，否则有可能因为忘了为脚本设定执行权限(x)，从而以为 OK 了， 结果无法完成任务，这样就“杯具”了。

- 4.3.1.4 在指定用户下执行相关定时任务。

需要 root 权限执行的任务可以登陆到 root 用户下然后设置，如果不需要 root 权限，可以登陆到普通用户下（也可以直接在 root 下 crontab -u oldboy -e 的写法直接设置），然后设置。这里要特别注意不同用户的环境变量问题，如果是调用了系统环境变量（如生产场景中 java 程序的定时任务），最好在程序脚本中将用到的环境变量重新 export 下（下文有案例）。

平时工作中尽量多用 crontab -e 和 crontab -l 去编辑和查看定时任务，因为，会有语法检查。

如果给 1000 台服务器同时添加系统时间同步就不可能一台台登录修改。那么此时就会用分发工具或者批量运维脚本（脚本内容就是 echo “定时任务规则” >>/var/spool/cron/root）。

要有注释，要有绝对路径，已经有输出重定向的不能加>

```
北京-李灿(361258724)    16:00:47
*/1 * * * * echo '+'>> /oldboy.log >/dev/null 2>&1
```

问题：

a. 没注释。

b. 定时任务规则如果是命令，一般要测试好加与不加>/dev/null 2>&1 行不行。这里加是不行的。

命令：

```
echo '#print a "+" every min by oldboy 21:55 2011-12-14' >>/var/spool/cron/root
echo '*/1 * * * * /bin/sh /tmp/echo.sh >/dev/null 2>&1' >>/var/spool/cron/root
```

#说明：以上方法均通过>>追加符号处理的，你们也可以通过 vi 编辑/var/spool/cron/root，或者通过 crontab -e 加入，效果是一样的，只不过没有语法检查，因此初学者应尽量用 crontab 命令完成。

1、* */2 * * * 这个任务规则分位上为*表示没分，这是错的。

2、/bin/sh 是执行脚本的，不是执行命令的。这是错的。

3、命令任务规则带%会导致任务无法执行。

所以带%的程序都应该放在脚本文件里或者转义

特殊说明：

1）$(date +%Y%m%d%H%M)为 date 命令的用法，表示按分钟打不同的压缩包，很重要 你可以执行 echo $(date +%Y%m%d%H%M); echo $(date %F)等。写法有$()或``反引号等，直接放在文件名上即可。

2）如果是按分打包，时间变量必须也要带日期并精确到分，否则，过了一定时间后包名就会循环覆盖了。

3）查看服务日志是个好习惯，是菜鸟通向高手的必经之路。

在执行任务时，某些在后台执行的任务，我们也可以用 nohup 或&

命令行例子：

nohup /server/scripts/oldboy.sh &

nohup 的使用是十分方便的，只需在要处理的命令前加上 nohup 即可，标准输出和标准错误缺省会被重定向到 nohup.out 文件中。一般我们可在结尾加上"&"来将命令同时放入后台运行，也可用">*filename* 2>&1"来更改缺省的重定向文件名。

更多可参考让进程在后台可靠运行的几种方法

http://www.ibm.com/developerworks/cn/linux/l-cn-nohup/

- 4.5.1 增加执行频率调试任务

1.在调试时，把任务执行频率调快一点，如：每分钟、每 5 分钟执行一次，或者比当前时间推迟 5 分钟以后，看能否执行，是不是按照你想象的去执行了，如果正常没问题了，在改成需要的任务的执行时间。

强调：有些任务是不允许频繁执行的，例如：定时往数据库里插入数据，这样的任务就要在测试机上测试好，然后正式线上出问题的机会就少了。

- 4.5.2 调整系统时间调试任务

2.用正确的执行任务的时间，设置完成后，可以修改下系统当前时间，改成任务执行时间的前几分钟来测试（或者重启定时任务服务）。如：定时任务 9：00 执行，我们可以把系统时间改成 8：55 分，然后观察是不是正确执行了，当前时间比任务时间要提前足够长，如果是生产服务器不要这样处理。

3.在脚本中加入日志输出，然后把输出打到指定的日志中，然后观察日志内容结果，看是否执行或正确执行。或象下面的内容把脚本结果定向到一个 log 文件里，重定向>即可，不需要>>追加，这样日志就不会一直变大，如/app/log.log。

```
#study task by oldboy at 20121213
00 9,14 * * 6,0 /bin/sh /server/scripts/oldboy.sh >/app/log.log 2>&1
```

4.5.4 注意一些任务命令带来的问题

4.注意：*/1 * * * * echo "==" >> /tmp/oldboy.log >/dev/null 2>&1 这是隐蔽的无法正确执行的任务配置，原因是前面多了>>，或者去掉结尾的>/dev/null 2>&1 。

4.5.5 注意环境变量导致的定时任务故障

5.在调试 java 程序任务的时候，注意环境变量，把环境变量的定义加到脚本里。

例：

```
[root@oldboy ~]# cat /scripts/resin/shell/Task.sh
#!/bin/bash
export JAVA_HOME=/application/jdk1.6
export PATH=$JAVA_HOME/bin:$PATH
export SH_HOME=/application/resin/webapps/ROOT/
export LIB=$SH_HOME/WEB-INF/lib
...省略部分...
定时任务：
00 9,14 * * * nohup /scripts/resin/shell/Task.sh & >/app/log.log 2>&1
```

1.export 变量问题

crontab 执行 shell 时只能识别为数不多的系统环境变量，普通变量是无法识别的，如果在编写的脚本中需要使用变量，最好使用 export 重新声明下该变量，脚本才能正常执行，例如：生产情况和 java 相关的服务任务和脚本。

2.任务路径问题

crontab 执行 shell 时，如果 shell 路径是相对路径或 shell 里含有相对路径，此时就会找不到任务路径，因此，在 shell 脚本中调用脚本或定时任务调用的脚本都要使用绝对路径。

范例：

```
* * * * * echo + >oldboy.log <==这是相对路径的写法。不规范。
```

3.脚本权限问题

要确保 crontab 的执行者有访问 shell 脚本所在目录并且执行此 shell 脚本的权限（可用 chmod 和 chown 修改脚本权限和所有者）。当然，最佳方法是执行脚本前加/bin/sh 执行测试下。在配置任务执行脚本时，可以省略当前用户配置，但最好带上/bin/sh，否则有可能因为忘了为脚本设定执行权限，而无法完成任务。本条是一个经验型的好习惯。

4.时间变量问题

“%”号在 crontab 任务中被认为是 newline，需要要用\来转义。crontab 任务命令中，如果有"date +%Y%m%d"，必须替换为："date +\%Y\%m\%d"，但写在脚本中就不需要了。这也是老师推荐用脚本文件的原因之一，如果是脚本文件，那么“%”就不需要转义了。

范例：

```
#tar comment by oldboy at 2012-11
*/1 * * * * cd /etc/ && tar zcvf /tmp/service_$(date +%Y%m%d%H%M).tar.gz ./services
```

6.定时任务加注释

写定时任务规则时要加上注释,这是个好习惯。如：什么人，什么时间，因为谁，做了什么事都标记清楚了，这样其他的维护人员就可以很容易理解。如老男孩于 2011-12-12 日在 oldboy 服务器上做了每 10 分钟同步时间的操作（如果是开发的任务可以写上需求人）。

范例：

7.使用脚本程序替代命令

使用脚本执行任务可以让我们少范错误，提升效率、规范，是个好习惯。定时任务中执行命令有一些限制，如时间变量问题，多个重定向命令混用问题等。

范例：

8.避免不必要的程序输出

在开发定时任务程序或脚本时，在调试好脚本程序后，应尽量把 DEBUG 及命令输出的内容信息屏蔽掉，如果还需要，可定向到指定日志文件里，以免产生多余的系统垃圾。

定时清理方法：

```
[root@oldboy ~]# echo "find /var/spool/clientmqueue/ -type f |xargs rm -f" >/server/scripts/del_sys_file.sh
    [root@oldboy ~]# cat /server/scripts/del_sys_file.sh
    find /var/spool/clientmqueue/  -type f |xargs rm -f
    [root@oldboy ~]# echo "00 00 * * * /bin/sh /server/scripts/del_sys_file.sh >/dev/null
2>&1" >>/var/spool/cron/root
    [root@oldboy ~]# echo "00 00 * * 0 /bin/sh /server/scripts/del_sys_file.sh >/dev/null
2>&1" >>/var/spool/cron/root
    [root@oldboy ~]# crontab -l|tail -2
    #del clientmqueue files by oldboy at 2010-09-26
    00 00 * * 0 /bin/sh /server/scripts/del_sys_file.sh >/dev/null 2>&1
```