Process Scheduling

cron

The **crond** service, launched at boot time, is responsible for executing certain scripts and commands at specific intervals. The crond service assumes that the system is online permanently. In the case of a downtime coinciding with a cronjob, the cronjob is simply not executed until the following period.

Under Red Hat/CentOS, every 60 seconds crond reads the system crontab file, **/etc/crontab**, any crontabs in **/etc/cron.d/** and all the user crontabs.

User crontabs are files named after the user who created them and can be found in **/var/spool/cron/**.

The crond service writes to a log file - **/var/log/cron**.

If a command or script produces an output, that output is sent to root by mail.

The **root** user can establish lists of users that can or cannot create their own crontabs by editing either the **/etc/cron.allow** or **/etc/cron.deny** files.

A typical system crontab under Red Hat/CentOS looks as follows:

crontab

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
```

Each line in a crontab starts with five columns:

Minutes	Hours	Day of the month	Month	Day of the week
(0-59)	(0-23)	(1-31)	(1-12)	(0|7-6)*

The following examples of values in those columns help explain how versatile a crontab can be:

Example	Description
An absolute value such as 10	In the *Minutes* column = 10 minutes after **each** hour
A series of values such as 2,6,8	In the *Month* column = February, June and August
A range such as 1-5	In the *Day of the week* column = From Monday through to Friday
The wildcard *	In the *Day of the month* column = Every day of the month
A regular interval such as 0-23/2	In the *Hour* column = Every two hours

Names of months or days of the week can be specified by name. Environment variables can be set in the crontab.

Special nicknames are supported :

- @reboot : Run once after reboot.
- @yearly : Run once a year, ie. “0 0 1 1 *”.
- @annually : Run once a year, ie. “0 0 1 1 *”.
- @monthly : Run once a month, ie. “0 0 1 * *”.
- @weekly : Run once a week, ie. “0 0 * * 0”.
- @daily : Run once * a day, ie. “0 0 * * *”.
- @hourly : Run once an hour, ie. “0 * * * *”.

As already stated, each authorized user can create a crontab. To check if a crontab exists, the user needs to execute the following command:

```
[trainee@centos ~]$ crontab -l
no crontab for trainee
```

In order to create a crontab, the user should use the following command:

```
$ crontab -e [Enter]
```

This command launches VI. Edit your crontab as follows:

crontab.trainee

```
* * * * * /bin/pwd > pwd.txt
```

Save and quit VI.

You will obtain a result similar to the following example:

```
[trainee@centos ~]$ crontab -e
no crontab for trainee - using an empty one
crontab: installing new crontab
```

The crontab that you have just created has been saved to disk in **/var/spool/cron/**.

You cannot view the contents of the file as a normal user. To do so you must become root:

```
[trainee@centos ~]$ su -
Password:
[root@centos ~]# cat /var/spool/cron/trainee
* * * * * /bin/pwd > pwd.txt
```

In order to allow or deny the right to a user to edit their crontab, root can edit either the **cron.allow** or **cron.deny** files. However, if root puts a user in the cron.deny file, any existing cronjobs will continue unless they are manually deleted:

anacron

The major drawback with cron is that it assumes the machine is up and running at all times. For this reason, anacron is now responsable for executing the contents of the following directories :

- /etc/cron.daily
- /etc/cron.weekly
- /etc/cron.monthly

Anacron reads a list of jobs from its configuration file **/etc/anacrontab**:

anacrontab

```
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days   delay in minutes   job-identifier   command
1   5    cron.daily        nice run-parts /etc/cron.daily
7   25    cron.weekly        nice run-parts /etc/cron.weekly
@monthly 45 cron.monthly        nice run-parts /etc/cron.monthly
```

This file contains two important columns at the beginning of each line:

Interval	Delay
period in days	delay in minutes

The period is the number of days that anacron waits before checking that the job has been executed. The delay in minutes is the delay from the moment that anacron checks after which the job will be run. Anacron is called upon boot.

Anacron records the date of execution in a special timestamp file for that job, so it can know when to execute it again. Only the date is used for the time calculations. The hour is not used :

```
[root@centos ~]# cat /var/spool/anacron/cron.daily
20131025
```

The files are stored in **/var/spool/anacron/**:

```
[root@centos ~]# ls -l /var/spool/anacron
total 12
-rw-------. 1 root root 9 Oct 25 18:08 cron.daily
-rw-------. 1 root root 9 Oct 25 18:46 cron.monthly
-rw-------. 1 root root 9 Oct 25 18:26 cron.weekly
```

at

As in the case of cron, root has the ability to control who can and cannot use the at command by editing one of the following:

- **/etc/at.allow**,
- **/etc/at.deny**.

If the at.allow file exists, only users in that file can use the at command.

Take the example of root creating two at jobs for the 31/12/2015 at 1pm and 2pm respectively:

```
[root@centos ~]# at 13:00 12/31/2015
```

```
at> pwd > /tmp/test1.atd
at> <EOT>
job 1 at 2015-12-31 13:00
[root@centos ~]# at 14:00 12/31/2015
at> free > /tmp/test2.atd
at> <EOT>
job 2 at 2015-12-31 14:00
```

The at files created can be found in **/var/spool/at/** :

```
[root@centos ~]# ls -l /var/spool/at
total 12
-rwx------. 1 root   root   2540 Oct 27 14:16 a00001017126d0
-rwx------. 1 root   root   2541 Oct 27 14:16 a000020171270c
drwx------. 2 daemon daemon 4096 Jan 30  2012 spool
```

Viewing the contents of the first file you will see something similar to the following example:

```
#!/bin/sh
# atrun uid=0 gid=0
# mail trainee 0
umask 22
HOSTNAME=centos.fenestros.loc; export HOSTNAME
SHELL=/bin/bash; export SHELL
HISTSIZE=1000; export HISTSIZE
QTDIR=/usr/lib/qt-3.3; export QTDIR
QTINC=/usr/lib/qt-3.3/include; export QTINC
USER=root; export USER
LS_COLORS=rs=0:di=01\;34:ln=01\;36:mh=00:pi=40\;33:so=01\;35:do=01\;35:bd=40\;33\;01:cd=40\;33\;01:or=40\;31\;01:
mi=01\;05\;37\;41:su=37\;41:sg=30\;43:ca=30\;41:tw=30\;42:ow=34\;42:st=37\;44:ex=01\;32:\*.tar=01\;31:\*.tgz=01\;
31:\*.arj=01\;31:\*.taz=01\;31:\*.lzh=01\;31:\*.lzma=01\;31:\*.tlz=01\;31:\*.txz=01\;31:\*.zip=01\;31:\*.z=01\;31
:\*.Z=01\;31:\*.dz=01\;31:\*.gz=01\;31:\*.lz=01\;31:\*.xz=01\;31:\*.bz2=01\;31:\*.tbz=01\;31:\*.tbz2=01\;31:\*.bz
=01\;31:\*.tz=01\;31:\*.deb=01\;31:\*.rpm=01\;31:\*.jar=01\;31:\*.rar=01\;31:\*.ace=01\;31:\*.zoo=01\;31:\*.cpio=
01\;31:\*.7z=01\;31:\*.rz=01\;31:\*.jpg=01\;35:\*.jpeg=01\;35:\*.gif=01\;35:\*.bmp=01\;35:\*.pbm=01\;35:\*.pgm=01
```

```
\;35:\*.ppm=01\;35:\*.tga=01\;35:\*.xbm=01\;35:\*.xpm=01\;35:\*.tif=01\;35:\*.tiff=01\;35:\*.png=01\;35:\*.svg=01
\;35:\*.svgz=01\;35:\*.mng=01\;35:\*.pcx=01\;35:\*.mov=01\;35:\*.mpg=01\;35:\*.mpeg=01\;35:\*.m2v=01\;35:\*.mkv=0
1\;35:\*.ogm=01\;35:\*.mp4=01\;35:\*.m4v=01\;35:\*.mp4v=01\;35:\*.vob=01\;35:\*.qt=01\;35:\*.nuv=01\;35:\*.wmv=01
\;35:\*.asf=01\;35:\*.rm=01\;35:\*.rmvb=01\;35:\*.flc=01\;35:\*.avi=01\;35:\*.fli=01\;35:\*.flv=01\;35:\*.gl=01\;
35:\*.dl=01\;35:\*.xcf=01\;35:\*.xwd=01\;35:\*.yuv=01\;35:\*.cgm=01\;35:\*.emf=01\;35:\*.axv=01\;35:\*.anx=01\;35
:\*.ogv=01\;35:\*.ogx=01\;35:\*.aac=01\;36:\*.au=01\;36:\*.flac=01\;36:\*.mid=01\;36:\*.midi=01\;36:\*.mka=01\;36
:\*.mp3=01\;36:\*.mpc=01\;36:\*.ogg=01\;36:\*.ra=01\;36:\*.wav=01\;36:\*.axa=01\;36:\*.oga=01\;36:\*.spx=01\;36:\
*.xspf=01\;36:; export LS_COLORS
MAIL=/var/spool/mail/root; export MAIL
PATH=/usr/lib/qt-3.3/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin; export PATH
PWD=/root; export PWD
LANG=en_US.UTF-8; export LANG
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass; export SSH_ASKPASS
HISTCONTROL=ignoredups; export HISTCONTROL
SHLVL=1; export SHLVL
HOME=/root; export HOME
LOGNAME=root; export LOGNAME
QTLIB=/usr/lib/qt-3.3/lib; export QTLIB
CVS_RSH=ssh; export CVS_RSH
LESSOPEN=\|/usr/bin/lesspipe.sh\ %s; export LESSOPEN
G_BROKEN_FILENAMES=1; export G_BROKEN_FILENAMES
XAUTHORITY=/root/.xauthuwdZK4; export XAUTHORITY
cd /root || {
     echo 'Execution directory inaccessible' >&2
     exit 1
}
${SHELL:-/bin/sh} << 'marcinDELIMITER14b80a14'
pwd > /tmp/test1.atd

marcinDELIMITER14b80a14
```

<note important> Notez que tous les jobs AT sont exécutés par **/bin/sh**. </note>

To delete that job you can use the at command:

```
[root@centos ~]# at -l
1	2015-12-31 13:00 a root
2	2015-12-31 14:00 a root
[root@centos ~]# at -d 1
[root@centos ~]# at -l
2	2015-12-31 14:00 a root
```

To execute several commands at the same time, it is simple to create a text file containing the commands and then to redirect the contents of the file to at's standard input:

```
[root@centos ~]# touch todo.list
[root@centos ~]# echo pwd > todo.list
[root@centos ~]# echo free >> todo.list
[root@centos ~]# echo who >> todo.list
[root@centos ~]# cat todo.list
pwd
free
who
[root@centos ~]# at 14:30 12/31/2015 < todo.list
job 3 at 2015-12-31 14:30
```

~~DISCUSSION:off~~
