**Job scheduling**

Job scheduling is a feature that allows a user to schedule a command for execution at a specified time in future. The execution of the command could be one time in future or at regular intervals based on a pre-determined time schedule. All this is taken care of by two daemons – **atd** and **crond**.

**Checking status of crond and atd daemons:**

REDHAT 5/6:

**# service crond status**

crond (pid 4050) is running...

**# service atd status**

atd (pid 4119) is running...

active at boot:

**# chkconfig crond on**

**# chkconfig atd on**

REDHAT 7:

# **systemctl status crond**

● crond.service - Command Scheduler

Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; vendor preset: enabled)

Active: active (running) since Thu 2016-11-17 03:08:14 EST; 2 months 29 days ago

Main PID: 4431 (crond)

CGroup: /system.slice/crond.service

└─4431 /usr/sbin/crond -n

# **systemctl status atd**

● atd.service - Job spooling tools

Loaded: loaded (/usr/lib/systemd/system/atd.service; enabled; vendor preset: enabled)

Active: active (running) since Thu 2016-11-17 03:08:10 EST; 2 months 29 days ago

Main PID: 2887 (atd)

CGroup: /system.slice/atd.service

└─2887 /usr/sbin/atd -f

active at boot:

# **systemctl enable crond**

# **systemctl enable atd**

**Controlling user access**

Which users can or cannot submit an **at** or **cron** job is controlled through files located in the /etc directory. For **at** job control, the **at.allow** and **at.deny** files are used, and that for **cron**, the **cron.allow** and **cron.deny** files are used. The table below shows various combinations and the impact on user access.

|  |  |  |
| --- | --- | --- |
| **at.allow / cron.allow** | **at.deny / cron.deny** | **Impact** |
| Exists, and contains user entries | Existence does not matter | All users listed in \*.allow files are permited |
| Exists, but empty | Existence does not matter | No users are permitted |
| Does not exist | Exists, and contains user entries | All users other than those listed in \*.deny files are permited |
| Does not exist | Exists, but empty | All users are permited |
| Does not exist | Does not exist | No users are permited |

**Cron log file**

All activities involving the **crond** daemon are logged to the **/var/log/cron** file. Information such as owner and start time for each invocation of **crontab** is captured. The file also keeps track of when the **crond** daemon was started, PID associated with it, spooled cron jobs, etc.

**# cat /var/log/cron**

Mar 23 09:30:21 john anacron[3981]: Job `cron.daily' terminated

Mar 23 09:30:21 john anacron[3981]: Job `cron.weekly' started

Mar 23 09:33:28 john anacron[3981]: Job `cron.weekly' terminated

Mar 23 09:33:28 john anacron[3981]: Normal exit (2 jobs run)

Mar 23 10:01:01 john crond[15228]: (root) CMD (run-parts /etc/cron.hourly)

Mar 28 10:18:56 john crond[4050]: (CRON) STARTUP (V5.0)

Mar 28 10:19:00 john anacron[4106]: Anacron 2.3 started on 2011-03-28

Mar 28 10:19:01 john anacron[4106]: Will run job `cron.daily' in 65 min.

Mar 28 10:19:01 john anacron[4106]: Will run job `cron.monthly' in 75 min.

Mar 28 10:19:01 john anacron[4106]: Jobs will be executed sequentially

**Scheduling an at job**

To understand scheduling an **at** job, consider the following examples. To schedule an **at** job to run the find command today at 11AM to look for all \*.vl files in the system and remove them:

**# at 11am**

at> find / -name \*.vl -exec rm -rf {} \;

at> <EOT>

job 1 at 2011-03-28 11:00

You have to press Ctrl+d to submit the **at** job. A file is created in the **/var/spool/at** directory, which will include all variable settings to be used when the job will actually run.

**#cat a00001014af000**

#!/bin/sh

# atrun uid=0 gid=0

# mail root 0

umask 22

...........................................

cd / || {

echo 'Execution directory inaccessible' >&2

exit 1

}

${SHELL:-/bin/sh} << `(dd if=/dev/urandom count=200 bs=1 2>/dev/null|LC\_ALL=C tr -d -c '[:alnum:]')`

find / -name \*.vl -exec rm -rf {} \;

**Listing and removing at jobs**

To list at jobs

**# at -l**

1 2011-03-28 11:00 a root

2 2011-03-28 11:00 a john

To remove a spooled at job:

**# at -l**

1 2011-03-28 11:00 a root

2 2011-03-28 11:00 a john

**# at -d 2** (2 is the job id specified by the at –l command)

**# at -l**

1 2011-03-28 11:00 a root

**Job scheduling using crontab**

Using the **crontab** command is the other method for scheduling tasks for execution in future. Unlike **at, crontab** executes jobs on a regular basis and at specified time defined in the master crontab file **/etc/crontab**. Crontab files for users are located in the **/var/spool/crontab** directory. Each user, who is allowed and has scheduled a cron job, has a file by his login name created in this directory. The third location where cron files can be stored is in **/etc/cron.d** directory. The **crond** daemon scans entries in **/etc/crontab** file and files at the two directory locations to determine job execution times. The daemon runs the commands or scripts at the specified time and puts a log entry in to the **/var/log/cron** file. The **crontab** command is used to edit, list and remove crontab files.

**Syntax of the user crontab file**

A crontab file has five fields for specifying day, date and time followed by the command to be run at that interval.

|  |
| --- |
| \*     \*   \*  \*   \*  command to be executed  -     -    -   -  -  |     |     |   |    |  |     |     |   |    +----- day of week (0 - 6) (Sunday=0)  |     |     |   +------- month (1 - 12)  |     |     +--------- day of month (1 - 31)  |     +----------- hour (0 - 23)  +------------- min (0 - 59) |

\* in the value field above means all legal values as in braces for that column.  
The value column can have a \* or a list of elements separated by commas. An element is either a number in the ranges shown above or two numbers in the range separated by a hyphen (meaning an inclusive range).  
**Notes**  
A.) Repeat pattern like /2 for every 2 minutes or /10 for every 10 minutes is not supported by all operating systems. If you try to use it and crontab complains it is probably not supported.

B.) The specification of days can be made in two fields: month day and weekday. If both are specified in an entry, they are cumulative meaning both of the entries will get executed.

**Scheduling a cron job**

To view scheduled cron jobs for current users:

**# crontab -l**

no crontab for root

To view scheduled cron jobs for a user on the system (only as root):

**# crontab -l -u john**

no crontab for john

To schedule a cron job use the crontab –e (edit) command. The root user can modify the contents of any other user’s crontab file. For example, do the following to edit the crontab file of user john as root:

**#crontab –u john –e**

no crontab for john - using an empty one

crontab: installing new crontab

**# crontab -u john -l**

30 08 10 06 \* /home/john/bla.sh

**30** – 30th Minute

**08** – 08 AM

**10** – 10th Day

**06** – 6th Month (June)

**\*** – Every day of the week

Some other cron jobs examples:

The following script takes an incremental backup twice a day every day. This example executes the specified incremental backup shell script (incremental-backup) at 11:00 and 16:00 on every day. The comma separated value in a field specifies that the command needs to be executed in all the mentioned time.

00 11,16 \* \* \* /home/john/bin/incremental-backup

* **00** – 0th Minute (Top of the hour)
* **11,16** – 11 AM and 4 PM
* **\*** – Every day
* **\*** – Every month
* **\*** – Every day of the week

Cron job everyday during working hours. This example checks the status of the database everyday (including weekends) during the working hours 9 a.m – 6 p.m

00 09-18 \* \* \* /home/ramesh/bin/check-db-status

* **00** – 0th Minute (Top of the hour)
* **09-18** – 9 am, 10 am,11 am, 12 am, 1 pm, 2 pm, 3 pm, 4 pm, 5 pm, 6 pm
* **\*** – Every day
* **\*** – Every month
* **\*** – Every day of the week

To remove the crontab file of john, use –r (remove) option :

**# crontab -u john -r**

**# crontab -u john -l**

no crontab for john

**What is anacron?**

**Anacron** is a service that checks at every system reboot for any **cron** and **at** scheduled jobs that were missed from running while the system was down. It checks its configuration file **/etc/anacrontab** and determines which jobs were not executed. It waits for a specified period of time and then executes them automatically. A typical /etc/anacrontab is displayed below:

**# cat /etc/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

1 65 cron.daily run-parts /etc/cron.daily

7 70 cron.weekly run-parts /etc/cron.weekly

30 75 cron.monthly run-parts /etc/cron.monthly

The first column represents the period in days, the second specifies the delay in minutes, the third identifies the job, the fourth lists the command that will execute contents of the scripts directory indicated in the fifth column. For each job, anacron checks whether the job was run in the past specified days (column 1) and executes it after waiting for the number of minutes (column 2) if it was not. Anacron may be run manually at the command prompt. For example, to run all the jobs that are scheduled in **/etc/anacrontab** and were missed:

**#anacron**