**Cron job scheduling**

**What is Cron?**

The [cron](https://en.wikipedia.org/wiki/Cron" \t "_blank) and at services enable sysadmins to schedule tasks to run at a specific time in the future. The at service specifies a one-time task that runs at a certain time. The cron service can schedule tasks on a repetitive basis, such as daily, weekly, or monthly.

**Common Uses**

The system times (i.e., the operating system time) on my many computers are set using the Network Time Protocol (NTP). While NTP sets the system time, it does not set the hardware time, which can drift. I use cron to set the hardware time based on the system time.

I also have a Bash program I run early every morning that creates a new "message of the day" (MOTD) on each computer. It contains information, such as disk usage, that should be current in order to be useful.

Many system processes and services, like [Logwatch](https://sourceforge.net/projects/logwatch/files/" \t "_blank), [logrotate](https://github.com/logrotate/logrotate" \t "_blank), and [Rootkit Hunter](http://rkhunter.sourceforge.net/), use the cron service to schedule tasks and run programs every day

The **crond** daemon is the background service that enables cron functionality.

The cron service checks for files in the **/var/spool/cron** and **/etc/cron.d** directories and the **/etc/anacrontab** file. The contents of these files define cron jobs that are to be run at various intervals. The individual user cron files are located in **/var/spool/cron**, and system services and applications generally add cron job files in the **/etc/cron.d** directory. The **/etc/anacrontab** is a special case that will be covered later in this article.

**Install crontab in centos 7**

<https://tecadmin.net/install-crontab-in-linux/>

**crontab -l**

**yum install cronie**

**start cron**

**[root@c1 ~]# systemctl start crond**

**[root@c1 ~]# systemctl enable crond**

**[root@c1 ~]# systemctl status crond**

When error comes as

crontab -u username -e

crontab -u root -e and save the file without writing anything

check for logs to see the status of job

cd /var/log

cat cron

Create first cron job and test the same

Creature a back up.sh write

#!/bin/bash

echo "this is cron job we need to access same" > test.txt

create file test.txt

33 19 \* \* \* /root/backup.sh

~

The path for root user is /var/spool/cron

|  |  |
| --- | --- |
| Directory | Description |
| /etc/cron.d/ | Put all scripts here and call them from /etc/crontab file. |
| /etc/cron.daily/ | Run all scripts once a day |
| /etc/cron.hourly/ | Run all scripts once an hour |
| /etc/cron.monthly/ | Run all scripts once a month |
| /etc/cron.weekly/ | Run all scripts once a week |

Daily job scheduling ::

1. mkdir /cachetmp
2. vi temp1.txt
3. vi temp2.txt
4. vi temp3.txt
5. /cachetmp

!#/bin/bash

#a sample script to show daily job schedule

# clean cache

CROOT="/cachetmp/"

# clean files every $days

DAYS=10

# okaay ,lets start cleaning]

find ${CROOT} -type -mtime +${DAYS} | args -r rm

#<https://www.gnu.org/software/findutils/manual/html_node/find_html/Deleting-Files.html>

#if directory is deleted confirm and get it back

if [ ! -d $CROOT ]

then

mkdir -p $CROOT

chown root:root ${CROOT}

fi

1. chmod +X clean.cache
2. touch -t 1909112009 tem\*

SHELL=/bin/bash

MAILTO=root@example.com

PATH=/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/root/bin:/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.211-2.6.17.1.el7\_6.x86\_64/bin

27 18 \* \* \* /sbin/hwclock --systohc

33 19 \* \* \* /root/backup.sh

19 22 \* \* \* root run-parts /etc/cron

## What is Crontab File

Crontab (cron table) is a text file that specifies the schedule of cron jobs. There are two types of crontab files. The system-wide crontab files and individual user crontab files.

Users’ crontab files are named according to the user’s name, and their location varies by operating systems. In Red Hat based distributions such as CentOS, crontab files are stored in the /var/spool/cron directory, while on Debian and Ubuntu files are stored in the /var/spool/cron/crontabs directory.

### Crontab Syntax and Operators

Each line in the user crontab file contains six fields separated by a space followed by the command to be run.

\* \* \* \* \* command(s)

- - - - -

| | | | |

| | | | ----- Day of week (0 - 7) (Sunday=0 or 7)

| | | ------- Month (1 - 12)

| | --------- Day of month (1 - 31)

| ----------- Hour (0 - 23)

------------- Minute (0 - 59)

The first five fields may contain one or more values, separated by a comma or a range of values separated by a hyphen.

* \* - The asterisk operator means any value or always. If you have the asterisk symbol in the Hour field, it means the task will be performed each hour.
* , - The comma operator allows you to specify a list of values for repetition. For example, if you have 1,3,5 in the Hour field, the task will run at 1 am, 3 am and 5 am.
* - - The hyphen operator allows you to specify a range of values. If you have 1-5 in the Day of week field, the task will run every weekday (From Monday to Friday).
* / - The slash operator allows you to specify values that will be repeated over a certain interval between them. For example, if you have \*/4 in the Hour field, it means the action will be performed every four hours. It is same as specifying 0,4,8,12,16,20. Instead of asterisk before the slash operator, you can also use a range of values, 1-30/10 means the same as 1,11,21.

## Linux Crontab Command

The crontab command allows you to install, [view](https://linuxize.com/post/how-to-list-cron-jobs-in-linux/) , or open a crontab file for editing:

* crontab -e - Edit crontab file, or create one if it doesn’t already exist.
* crontab -l - Display crontab file contents.
* crontab -r - Remove your current crontab file.
* crontab -i - Remove your current crontab file with a prompt before removal.
* crontab -u <username> - Edit other user crontab file. This option requires system administrator privileges.

**Other scheduling tricks**

Now let's do some things that are a little more interesting than these basics. Suppose you want to run a particular job every Thursday at 3 p.m.:

**00 15 \* \* Thu /usr/local/bin/mycronjob.sh**

This line runs**mycronjob.sh**every Thursday at 3 p.m.

Or, maybe you need to run quarterly reports after the end of each quarter. The cron service has no option for "The last day of the month," so instead you can use the first day of the following month, as shown below. (This assumes that the data needed for the reports will be ready when the job is set to run.)

**02 03 1 1,4,7,10 \* /usr/local/bin/reports.sh**

This cron job runs quarterly reports on the first day of the month after a quarter ends.

The following shows a job that runs one minute past every hour between 9:01 a.m. and 5:01 p.m.

**01 09-17 \* \* \* /usr/local/bin/hourlyreminder.sh**

I have encountered situations where I need to run a job every two, three, or four hours. That can be accomplished by dividing the hours by the desired interval, such as **\*/3** for every three hours, or **6-18/3** to run every three hours between 6 a.m. and 6 p.m. Other intervals can be divided similarly; for example, the expression **\*/15** in the minutes position means "run the job every 15 minutes."

**\*/5 08-18/2 \* \* \* /usr/local/bin/mycronjob.sh**

This cron job runs every five minutes during every hour between 8 a.m. and 5:58 p.m.

One thing to note: The division expressions must result in a remainder of zero for the job to run. That's why, in this example, the job is set to run every five minutes (08:05, 08:10, 08:15, etc.) during even-numbered hours from 8 a.m. to 6 p.m., but not during any odd-numbered hours. For example, the job will not run at all from 9 p.m. to 9:59 a.m.

I am sure you can come up with many other possibilities based on these examples.

### Limiting cron access

Regular users with cron access could make mistakes that, for example, might cause system resources (such as memory and CPU time) to be swamped. To prevent possible misuse, the sysadmin can limit user access by creating a **/etc/cron.allow** file that contains a list of all users with permission to create cron jobs. The root user cannot be prevented from using cron.

By preventing non-root users from creating their own cron jobs, it may be necessary for root to add their cron jobs to the root crontab. "But wait!" you say. "Doesn't that run those jobs as root?" Not necessarily. In the first example in this article, the username field shown in the comments can be used to specify the user ID a job is to have when it runs. This prevents the specified non-root user's jobs from running as root. The following example shows a job definition that runs a job as the user "student":

**04 07 \* \* \* student /usr/local/bin/mycronjob.sh**

### cron.d

The directory **/etc/cron.d** is where some applications, such as [SpamAssassin](http://spamassassin.apache.org/" \t "_blank) and [sysstat](https://github.com/sysstat/sysstat" \t "_blank), install cron files. Because there is no spamassassin or sysstat user, these programs need a place to locate cron files, so they are placed in **/etc/cron.d**.

The **/etc/cron.d/sysstat** file below contains cron jobs that relate to system activity reporting (SAR). These cron files have the same format as a user cron file.

***# Run system activity accounting tool every 10 minutes*  
\*/10 \* \* \* \* root /usr/lib64/sa/sa1 1 1  
*# Generate a daily summary of process accounting at 23:53*  
53 23 \* \* \* root /usr/lib64/sa/sa2 -A**

The sysstat package installs the**/etc/cron.d/sysstat**cron file to run programs for SAR.

The sysstat cron file has two lines that perform tasks. The first line runs the **sa1** program every 10 minutes to collect data stored in special binary files in the **/var/log/sa** directory. Then, every night at 23:53, the **sa2** program runs to create a daily summary.

1. he crond service runs the cron job specified in **/etc/cron.d/0hourly**.

***# Run the hourly jobs*  
SHELL=/bin/bash  
PATH=/sbin:/bin:/usr/sbin:/usr/bin  
MAILTO=root  
01 \* \* \* \* root run-parts /etc/cron.hourly**

The contents of**/etc/cron.d/0hourly**cause the shell scripts located in**/etc/cron.hourly**to run.

1. The cron job specified in **/etc/cron.d/0hourly** runs the **run-parts** program once per hour.
2. The **run-parts** program runs all the scripts located in the **/etc/cron.hourly** directory.
3. The **/etc/cron.hourly** directory contains the **0anacron** script, which runs the anacron program using the **/etdc/anacrontab** configuration file shown here.

***# /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**

The contents of**/etc/anacrontab**file runs the executable files in the**cron.[daily|weekly|monthly]**directories at the appropriate times.

1. The anacron program runs the programs located in **/etc/cron.daily** once per day; it runs the jobs located in **/etc/cron.weekly** once per week, and the jobs in **cron.monthly** once per month. Note the specified delay times in each line that help prevent these jobs from overlapping themselves and other cron jobs.

Instead of placing complete Bash programs in the **cron.X** directories, I install them in the **/usr/local/bin** directory, which allows me to run them easily from the command line. Then I add a symlink in the appropriate cron directory, such as **/etc/cron.daily**.

The anacron program is not designed to run programs at specific times. Rather, it is intended to run programs at intervals that begin at the specified times, such as 3 a.m. (see the **START\_HOURS\_RANGE** line in the script just above) of each day, on Sunday (to begin the week), and on the first day of the month. If any one or more cycles are missed, anacron will run the missed jobs once, as soon as possible.

<https://stackoverflow.com/questions/29612386/cron-job-use-for-running-hadoop-program-in-linux>

<https://zedar.gitbooks.io/spark-hadoop-notes/content/spark_job_running_in_cron.html>

<https://docs.ansible.com/ansible/latest/modules/cron_module.html>