cron Scheduled jobs

START HERE

crontab -e
Edit your crontab

@reboot *<cmd>*
Once at daemon start

crontab -l
List your crontab

MAILTO=""
Silence output mail

***/5 * * * *** *<cmd>*
Every 5 minutes

<cmd> **>>** *<log>* **2>&1**
Capture output

0 2 * * * *<cmd>*
Daily at 02:00

crontab -n *<file>*
Syntax check only

CONCEPTS

Cron wakes every minute and runs jobs whose five time fields match. Jobs run with a minimal environment and no terminal, so use absolute paths and set PATH yourself.

min	hour	dom	month	dow
0-59	0-23	1-31	1-12	0-7

Field order, left to right (not nested)

If both day-of-month and day-of-week are restricted (not *), the job runs when either matches, not both. 30 4 1,15 * 5 runs on the 1st, the 15th and every Friday.

MANAGING CRONTABS

crontab -e
Edit your crontab in $EDITOR

crontab -l
Show your installed crontab

crontab -r
Remove your crontab, no confirmation
No undo and it sits next to -e on the keyboard; use -i to be asked first.

crontab -i -r
Ask before removing

crontab *<file>*
Replace your crontab with a file

crontab -
Replace your crontab from stdin

crontab -n *<file>*
Dry run: check syntax only

crontab -u *<user>* **-l**
List another user's crontab (root)

crontab -u *<user>* **-e**
Edit another user's crontab (root)

crontab -l | crontab -
Reinstall as-is (round trip)

/etc/cron.d/*<file>*
System job file, has a user field

MANAGING CRONTABS (cont.)

/etc/cron.daily/
Scripts run by the system daily

/etc/cron.allow /etc/cron.deny
Who may use crontab
If cron.allow exists only listed users may use crontab and cron.deny is ignored; root is always allowed.

SCHEDULE SYNTAX

*** * * * *** *<cmd>*
Every minute

***/15 * * * *** *<cmd>*
Every 15 minutes

0 * * * * *<cmd>*
Top of every hour

30 4 * * * *<cmd>*
Every day at 04:30

0 9 * * 1-5 *<cmd>*
Weekdays at 09:00

0 0 1 * * *<cmd>*
First of the month at midnight

0 8,12,18 * * * *<cmd>*
List: 08:00, 12:00, 18:00

0 9-17/2 * * * *<cmd>*
Range with step: 9, 11, 13, 15, 17

0 6 * * mon *<cmd>*
Names allowed for month and weekday
Use the first three letters. The man page disallows lists and ranges of names, so use numbers there.

0 0 * * 0 *<cmd>*
Sundays (0 or 7)

0 3 1-7 * * ["$(date +\%a)" = Sun] && *<cmd>*
First Sunday of the month

SPECIAL STRINGS

@reboot
Once when the cron daemon starts
Runs when cron starts, which can be before network or other services are up.

@hourly
Same as 0 * * * *

SPECIAL STRINGS (cont.)

@daily / @midnight
Same as 0 0 * * *

@weekly
Same as 0 0 * * 0

@monthly
Same as 0 0 1 * *

@yearly / @annually
Same as 0 0 1 1 *

ENVIRONMENT AND OUTPUT

PATH=/usr/local/bin:/usr/bin:/bin
Set PATH at the top (no $PATH)
Values are not expanded: PATH=$HOME/bin:$PATH and ~ do not work.

SHELL=/bin/bash
Choose the shell for commands

MAILTO=*<addr>*
Mail job output to this address

MAILTO=""
Never mail output

<cmd> **>>** *<log>* **2>&1**
Append stdout and stderr to a log

<cmd> **> /dev/null 2>&1**
Discard all output

<cmd> **> /dev/null**
Keep errors, drop normal output

<cmd> **2>&1 | logger -t** *<tag>*
Send output to syslog

date +\%F
Escape % with a backslash
An unescaped % ends the command and the rest becomes stdin.

<cmd> **# note**
Trailing # is NOT a comment
The text is passed to the shell as part of the command line; only whole-line comments are safe in a crontab.

flock -n *<lock>* *<cmd>*
Skip if the previous run is still going

sleep $((RANDOM \% 60)); *<cmd>*
Random delay (needs SHELL=bash, escape %)

DEBUGGING

grep CRON /var/log/syslog
See cron run lines (needs rsyslog)
Newer Debian has no /var/log/syslog unless rsyslog is installed; use journalctl -u cron.

journalctl -u cron
Cron log on systemd (cron.service on Debian)
The unit is crond on RHEL and Fedora.

systemctl status cron
Is the daemon running?

env -i SHELL=/bin/sh PATH=/usr/bin:/bin *<cmd>*
Test with a cron-like environment

run-parts --test /etc/cron.daily
List what a cron.* dir would run

A job that works in your shell but fails in cron nearly always comes down to PATH, a relative path, a missing env variable or a % character.

For logging, dependencies and catching up after downtime, consider a systemd timer instead (systemctl list-timers).

RECIPES

Typical crontab header

```
SHELL=/bin/bash
PATH=/usr/local/bin:/usr/bin:/bin
MAILTO=you@example.com
# m  h  dom mon dow  command
15 3 * * *  /home/me/bin/backup.sh >> /home/me/backup.log 2>&1
```

Add one job without opening the editor

```
(crontab -l 2>/dev/null; echo '*/10 * * * * /home/me/bin/sync.sh') | crontab -
```

Non-overlapping job with a lock

```
*/5 * * * * flock -n /tmp/sync.lock /home/me/bin/sync.sh >> /home/me/sync.log 2>&1
```

Dated log files (note the escaped %)

```
0 1 * * * /home/me/bin/report.sh > /home/me/reports/$(date +\%F).txt 2>&1
```