LPI 107.2 - Automate system administration tasks by scheduling jobs

Curs 2021 - 2022

ASIX M01-ISO 107 Administrative Tasks

Automate system administration tasks by scheduling jobs

Description

Key concepts:

- ❏ Manage cron and at jobs.
- ❏ Configure user access to cron and at services.
- ❏ Understand systemd timer units.

Commands and files:

- ❏ /etc/cron.{d,daily,hourly,monthly,weekly}/
- ❏ /etc/at.deny
- ❏ /etc/at.allow
- ❏ /etc/crontab
- ❏ /etc/cron.allow
- ❏ /etc/cron.deny
- ❏ /var/spool/cron/
- ❏ crontab
- ❏ at
- ❏ atq
- ❏ atrm
- ❏ systemctl
- ❏ systemd-run

Scheduling jobs with cron

Cron, at and batch are the utilities to schedule tasks in a GU/Linux system.

Scheduling tasks that are required to run at regular intervals use cron. .The crond daemon “wakes up” every minute, examines all of the crontab files that are stored in memory, and determines if there is a particular command or script requiring execution. If such a command is found, the crond daemon launches that process. The tasks are executed in the Time Zone of the server (is not possible to personalize in the scripts).

In Linux there is also the anacron facility, which is suitable for systems that can be powered off (such as desktops or laptops). It can only be used by root. If the machine is off when the

anacron jobs must be executed, they will run the next time the machine is powered on. anacron is out of scope for the LPIC-1 certification.

The cron system consists on:

- the crond daemon. Execute periodically the tasks..
- the crontab file describing tasks: system crontab and user crontab.
- the crontab command to edit/view the users crontab file.

Crontab scope

- system crontab: /etc/crontab and /etc/cron.d
- users crontab: editing with crontab. Attention root crontab is a user's crontab (with root privileges) and is different from /etc/crontab (system crontab).

System contab

- /etc/crontab
- /etc/cron.d
- /etc/cron.daily /etc/cron.hourly /etc/cron.weekly /etc/cron.monthly

- The system crontab allows scheduling of system-wide tasks such as log rotations, backups, and system database updates. Root privileges.
- Can execute tasks for any user but only root can create them. (observe the user field). Used for global system tasks.

/etc/crontab

```
minute hour mday month wday user command
```

```
#1
# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# 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
```

- Components:
  - comments (starting with #)
  - shell variables (SHELL, PATH) that define the environment for the scheduled jobs when they run.
  - scheduled tasks, one per line

```
#2
```

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
0 23 * * * root /usr/root/bkup.sh
*/30 * * * * dev /usr/dev/ant_build.sh
```

User crontab

- Each user can have his own crontab, his own scheduled task lists.
- The tasks are executed in the user's name. Not possible execute in name of another user.
- The crontab line differ in no user field is accepted.

```
minute hour mday month wday command
```

```
#3
0 17 * * 5 /usr/bin/banner  "The Weekend Is Here!" > /dev/console
```

Crontab file format

The global crontab file /etc/crontab and the individual crontab files differ in the field user, but all the other fields are the same. Let's see the meaning of each field:

```
# Example of job definition: in /etc/crontab
# .---------------- 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
```

Field description:

Minute	Range 0 – 59
Hour	Range 0 – 23
Day-of-month	Range 1 – 31
Month	Range 1 – 12 (Jan-Dec)
Day-of-week	Range 0 – 7 (0 & 7 = Sunday)
Command	The command to be executed

It is possible to indicate sets and ranges using the following notation:

- A single value
- Multiple values such as 1,3,5 in the fifth field, which would mean Monday, Wednesday, Friday
- A range of values (such as 1 through 5 (1-5)) in the fifth field, which would mean Monday through Friday

- An asterisk * character means any or all values
- Step values using */2 menys each 2 minutes (in minutes field), */5 each 5 hours in an hours field.

Precaution:

- Remember, friday night after 24h is saturday!

Examples:

```
30 04 * * * /usr/bin/some_command
01,31 04,05 1-15 1,6 * /usr/bin/some_command
00 08-17 * * 1-5 /usr/bin/some_command
*/2 10-22 * * 1,3,5 /usr/bin/some_command
* * * * * /usr/bin/copia.sh  /etc 3 &>> /backups/backup_$(date +\%y\%m\%d).log
```

Variables

The crontab file can also contain variables, the `crond` utility does not read user initialization files when it executes commands. This means that creating variables like the `PATH` variable is very important.

Some of the most commonly-found variables are:

HOME=/home/user
PATH=/usr/local/bin:/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin
SHELL=/bin/bash
MAILTO=databaseadmin

The HOME directory indicates from which directory are commands executed. The PATH is necessary to locate the executable, the SHELL indicates which shell use (otherwise /bin/sh will be used) and MAILTO indicates to which mail will be sent the crontab report once finished the execution, containing the output generated.

The users (and root) can create their own variables in the crontab configuration files.

Crontab keywords

Special keywords with meaning for the crontab utility:

@reboot
    Run once, at startup
@yearly
@annually
    Run once a year. 0 0 1 1 *
@monthly
    Run once a month. 0 0 1 * *
@weekly
    Run once a week. 0 0 * * 0
@daily
@midnight
    Run once a day. 0 0 * * *
@hourly

Run once an hour. 0 * * * *

Equivalent examples

```
0 0 * * *  /home/sysadmin/bin/daily-backup
@daily  /home/sysadmin/bin/daily-backup
@midnight  /home/sysadmin/bin/daily-backup
```

Maintaining crontab files

User files

- crontab -e
- crontab -l
- crontab -r
- EDITOR
- /etc/crontab
- /var/spool/cron/<user>

Usual commands:

```
$ crontab -e
$ crontab -l
$ crontab -r
$ export EDITOR=vim
$ export EDITOR=geany
```

```
# crontab -u user -e
# crontab -u user -l
# crontab -u user -r
```

```
#3
$ echo $EDITOR

$ crontab -e
no crontab for ecanet - using an empty one
crontab: installing new crontab

$ crontab -l
*/5 * * * *  touch /tmp/file-$(date +"%Y-%m-%d")
@reboot date >> /tmp/reboot.log

$ ls /var/spool/cron/
ls: cannot open directory '/var/spool/cron/': Permission denied
```

```
# ls -l /var/spool/cron/
total 8
-rw-------. 1 ecanet ecanet 81 Nov  1 12:03 ecanet
-rw-------. 1 root   root   80 Apr 16  2021 root

# cat /var/spool/cron/ecanet
*/5 * * * *  touch /tmp/file-$(date +"%Y-%m-%d")
@reboot date >> /tmp/reboot.log
```

```
$ crontab -r

$ crontab -l
no crontab for ecanet
```

System files

- /etc/crontab
- /etc/cron.d
- /etc/cron.hourly
- /etc/cron.daily
- /etc/cron.weekly
- /etc/cron.monthly

The directories described above include files that are executed in the indicated period. Some programs in the installation process create their own files in the cron directories to ensure the execution of a periodic task.

```
#4
# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# 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
@reboot root date >> /var/log/reboot.log

# ls /etc/cron
cron.d/       cron.daily/   cron.deny     cron.hourly/  cron.monthly/ crontab
cron.weekly/

# ls /etc/cron.hourly/
0anacron

# ls /etc/cron.daily/
google-chrome

# ls /etc/cron.weekly/
98-zfs-fuse-scrub

# ls /etc/cron.monthly/

# ls /etc/cron.d
0hourly
```

Access to cron daemon

- systemctl start|stop|status|reload|restart crond
- /etc/cron.allow
- /etc/cron.deny

User root can select which user can or cannot use cron using the files /etc/cron.allow and /etc/cron.deny:

- If both the cron.allow and the cron.deny files do not exist, the default is to allow all users to use the crontab command.
- If only the cron.allow file exists, then only the users listed in the file can execute the crontab command.
- If only the cron.deny file exists, then all users listed in this file are denied the ability to execute the crontab command, and all other users are allowed to use the crontab command.
- If both the cron.allow and the cron.deny files exist, the cron.allow file applies and the cron.deny file is ignored. As only one of these files should exist, the presence of both files is typically due to a mistake made by the administrator.

- The rules apply to new crontab jobs, but if the crontab.allow or crontab.deny file changes after the user launches crontab tasks, they are not affected.

```
Example 1: 100 users, only five should be able to use the crontab command.

In this scenario, create the /etc/cron.allow file and place the five users who are
allowed to use the crontab command in this file.

Do not create a /etc/cron.deny file.
```

```
Example 2: 100 users, only five should be denied from using the crontab command.

In this scenario, create the /etc/cron.deny file and place the five users who are denied
access to the crontab command in this file.

Do not create a /etc/cron.allow file.
```

```
Example 3: 100 users, none should be able to use the crontab command.

In this scenario, do not create a /etc/cron.deny, but do create an empty /etc/cron.allow
file to specify that no users have permission to access to crontab.
```

```
#5
# vim /etc/cron.deny

# cat /etc/cron.deny
ecanet

$ crontab -e
You (ecanet) are not allowed to use this program (crontab)
See crontab(1) for more information

$ crontab -l
You (ecanet) are not allowed to use this program (crontab)
See crontab(1) for more information
```

```
#6
# systemctl  status crond
● crond.service - Command Scheduler
     Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2021-11-01 11:10:54 CET; 1h 24min ago
   Main PID: 1082 (crond)
      Tasks: 1 (limit: 8916)
     Memory: 1.4M
        CPU: 140ms
     CGroup: /system.slice/crond.service
             └─1082 /usr/sbin/crond -n

Nov 01 11:10:54 localhost.localdomain crond[1082]: (CRON) INFO (RANDOM_DELAY will be scaled with factor 13% if used.)
Nov 01 11:10:54 localhost.localdomain crond[1082]: (CRON) INFO (running with inotify support)
Nov 01 12:01:01 localhost.localdomain CROND[7133]: (root) CMD (run-parts /etc/cron.hourly)
```

```
Nov 01 12:01:01 localhost.localdomain anacron[7142]: Anacron started on 2021-11-01
Nov 01 12:01:01 localhost.localdomain anacron[7142]: Will run job `cron.daily' in 11 min.
Nov 01 12:01:01 localhost.localdomain anacron[7142]: Jobs will be executed sequentially
Nov 01 12:01:01 localhost.localdomain run-parts[7144]: (/etc/cron.hourly) finished 0anacron
Nov 01 12:12:01 localhost.localdomain anacron[7142]: Job `cron.daily' started
Nov 01 12:12:01 localhost.localdomain anacron[7142]: Job `cron.daily' terminated
Nov 01 12:12:01 localhost.localdomain anacron[7142]: Normal exit (1 job run)
```

Watch logs: syslog / journald

```
#7
# journalctl -u crond | tail -10
Nov 01 11:10:54 localhost.localdomain crond[1082]: (CRON) INFO (running with inotify
support)
Nov 01 12:01:01 localhost.localdomain CROND[7133]: (root) CMD (run-parts
/etc/cron.hourly)
Nov 01 12:01:01 localhost.localdomain anacron[7142]: Anacron started on 2021-11-01
Nov 01 12:01:01 localhost.localdomain anacron[7142]: Will run job `cron.daily' in 11
min.
Nov 01 12:01:01 localhost.localdomain anacron[7142]: Jobs will be executed sequentially
Nov 01 12:01:01 localhost.localdomain run-parts[7144]: (/etc/cron.hourly) finished
0anacron
Nov 01 12:12:01 localhost.localdomain anacron[7142]: Job `cron.daily' started
Nov 01 12:12:01 localhost.localdomain anacron[7142]: Job `cron.daily' terminated
Nov 01 12:12:01 localhost.localdomain anacron[7142]: Normal exit (1 job run)
Nov 01 13:01:01 localhost.localdomain CROND[12135]: (root) CMD (run-parts
/etc/cron.hourly)

# journalctl -f
Nov 01 13:14:06 localhost.localdomain crontab[12701]: (ecanet) REPLACE (ecanet)
Nov 01 13:14:06 localhost.localdomain crontab[12701]: (ecanet) END EDIT (ecanet)
Nov 01 13:14:11 localhost.localdomain audit[12757]: USER_ACCT pid=12757 uid=1001
auid=1001 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
msg='op=PAM:accounting grantors=pam_access,pam_unix,pam_localuser acct="ecanet"
exe="/usr/bin/crontab" hostname=? addr=? terminal=cron res=success'
Nov 01 13:14:11 localhost.localdomain audit[12757]: CRED_ACQ pid=12757 uid=1001
auid=1001 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
msg='op=PAM:setcred grantors=pam_env,pam_fprintd acct="ecanet" exe="/usr/bin/crontab"
hostname=? addr=? terminal=cron res=success'
Nov 01 13:14:11 localhost.localdomain crontab[12757]: (ecanet) LIST (ecanet)
Nov 01 13:14:19 localhost.localdomain NetworkManager[926]: <info>  [1635768859.7515] Nov
01 13:14:29 localhost.localdomain audit[12767]: USER_ACCT pid=12767 uid=1001 auid=1001
ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='op=PAM:accounting
grantors=pam_access,pam_unix,pam_localuser acct="ecanet" exe="/usr/bin/crontab"
hostname=? addr=? terminal=cron res=success'
Nov 01 13:14:29 localhost.localdomain audit[12767]: CRED_ACQ pid=12767 uid=1001
auid=1001 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
msg='op=PAM:setcred grantors=pam_env,pam_fprintd acct="ecanet" exe="/usr/bin/crontab"
hostname=? addr=? terminal=cron res=success'
Nov 01 13:14:29 localhost.localdomain crontab[12767]: (ecanet) BEGIN EDIT (ecanet)
Nov 01 13:14:30 localhost.localdomain systemd[1850]: Started VTE child process 12773
launched by geany process 12769.
```

At and Batch jobs

For scheduling one-time tasks, the at and batch commands are more useful. There are two prerequisites for running the at command: the at package must be installed (install if necessari, for example: `sudo apt-get install at`, or `sudo dnf install atd`), and atd service must be running.

- at time

- /var/spool/at
- interactive: commands + ^d
- at -f fie-tasks time
- at -f file-tasks -m time
- atq (list, query at tasks) (at -l)
- atrm (delete at tasks) (at -d)

At time specification examples:

[assume the current date/time is 1st Mar 2025, 8:00 a.m]

midnight	12:00 a.m. 2nd Mar 2025
noon	12:00 p.m .1st Mar 2025
tomorrow	8:00 a.m. 2nd Mar 2025
next week	8:00 a.m. 8th Mar 2025
1630	4:30 p.m. 1st Mar 2025
4:30 PM Mar 20	4:30 p.m. 20th Mar 2025
now + 2 hours	10:00 a.m .1st Mar 2025
now + 7 days	8:00 a.m .8th Mar 2025
HH:MM MMDD[CC]YY	specific date and time
HH:MM MM/DD/[CC]YY	specific date and time
HH:MM [CC]YY-MM-DD	specific date and time
time month-name day	specific time and date

Examples

```
$ at now + 5 min
$ at 10am Jul 31
$ at 4pm + 3 days
$ at 1am tomorrow
$ at 23:00 03032022
at 23:00 2022-03-10
```

The at and batch command can be used interactive, introduce one command per line and finalize with ^d.

```
#8
# systemctl  is-active atd
inactive

# dnf -y install atd
## apt-get install at

# systemctl  start atd

# systemctl  is-active atd
active

[# systemctl enable atd]
```

```
#9
# at now+1min
warning: commands will be executed using /bin/sh
at> touch /tmp/file
at> echo "hola" > /tmp/hola.txt
at> <EOT>
job 1 at Mon Nov  1 12:47:00 2021

# ls /var/spool/at/
```

```
a00001019fff43  .SEQ            spool/

# ls -l /var/spool/at/
total 8
-rwx------. 1 root root 3934 Nov  1 12:47 a00002019fff45
drwx------. 2 root root 4096 Nov  1 12:47 spool

# cat /var/spool/at/a00002019fff45
#!/bin/sh
# atrun uid=0 gid=0
# mail ecanet 0
umask 22
SHELL=/bin/bash; export SHELL
HISTCONTROL=ignoredups; export HISTCONTROL
HISTSIZE=1000; export HISTSIZE
HOSTNAME=localhost.localdomain; export HOSTNAME
GUESTFISH_OUTPUT=\\e[0m; export GUESTFISH_OUTPUT
PWD=/root; export PWD
LOGNAME=root; export LOGNAME
MODULESHOME=/usr/share/Modules; export MODULESHOME
MANPATH=:; export MANPATH
XAUTHORITY=/root/.xauthr9ij0O; export XAUTHORITY
GUESTFISH_RESTORE=\\e[0m; export GUESTFISH_RESTORE
HOME=/root; export HOME
GUESTFISH_PS1=\\[\\e[1\;32m\\]\>\<fs\>\\[\\e[0\;31m\\]\ ; export GUESTFISH_PS1
MOZ_GMP_PATH=/usr/lib64/mozilla/plugins/gmp-gmpopenh264/system-installed; export MOZ_GMP_PATH
MODULEPATH_modshare=/usr/share/modulefiles:1:/usr/share/Modules/modulefiles:1:/etc/modulefiles:1;                  export
MODULEPATH_modshare
LESSOPEN=\|\|/usr/bin/lesspipe.sh\ %s; export LESSOPEN
USER=root; export USER
MODULES_RUN_QUARANTINE=LD_LIBRARY_PATH; export MODULES_RUN_QUARANTINE
LOADEDMODULES=; export LOADEDMODULES
SHLVL=1; export SHLVL
GUESTFISH_INIT=\\e[1\;34m; export GUESTFISH_INIT
XDG_DATA_DIRS=/root/.local/share/flatpak/exports/share:/var/lib/flatpak/exports/share:/usr/local/share:/usr/share;  export
XDG_DATA_DIRS
PATH=/root/.local/bin:/root/bin:/usr/share/Modules/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin; export PATH
MODULEPATH=/etc/scl/modulefiles:/usr/share/Modules/modulefiles:/etc/modulefiles:/usr/share/modulefiles; export MODULEPATH
MAIL=/var/spool/mail/root; export MAIL
MODULES_CMD=/usr/share/Modules/libexec/modulecmd.tcl; export MODULES_CMD
cd /root || {
          echo 'Execution directory inaccessible' >&2
          exit 1
}
${SHELL:-/bin/sh} << 'marcinDELIMITER69c990cc'
touch /tmp/file
echo "hola" > /tmp/hola.txt
marcinDELIMITER69c990cc

# atq
3       Mon Nov  1 12:52:00 2021 a root
```

- at -f file time
- atq
- atrm

```
#10
# cat > atjobs
echo "hola" >> /tmp/hola.txt
touch /tmp/file

# at -f atjobs now + 1min
warning: commands will be executed using /bin/sh
job 4 at Mon Nov  1 12:56:00 2021

# at -f atjobs now + 2min
warning: commands will be executed using /bin/sh
job 5 at Mon Nov  1 12:57:00 2021

# at -f atjobs now + 3min
warning: commands will be executed using /bin/sh
job 6 at Mon Nov  1 12:58:00 2021

# atq
4       Mon Nov  1 12:56:00 2021 a root
5       Mon Nov  1 12:57:00 2021 a root
6       Mon Nov  1 12:58:00 2021 a root

# atrm 5
```

```
# atq
6       Mon Nov  1 12:58:00 2021 a root
[note*: task 4 not appear because has finished ]
```

- /etc/at.allow
- /etc/at.deny

Identical to the cron daemon the root user can control who can use the at and batch utilities using the files at.allow and at.deny. The rules mentioned for cron access are applicable for at access also:

- Attention! If both files do not exist it may not work. Create at least one of them, one empty at.deny. .
- If only the at.allow file exists, then only the users listed in the file can execute the at and batch commands.
- if only the at.deny file exists, then all users listed in this file are denied the ability to execute the at and batch commands, and all other users are allowed to execute the at and batch commands.
- If both the at.allow and the at.deny files exist, the at.allow file applies and the at.deny file is ignored. As only one of these files should exist, the presence of both files is typically due to a mistake made by the administrator.

- if a user has been denied access to the at and batch commands, this does not prevent any previously created at or batch entries from running. If the user already had at or batch entries, the administrator needs to remove those entries.

```
#11
# vim /etc/at.deny
# cat /etc/at.deny
ecanet
pere

$ at now+1
You do not have permission to use at.
```

Batch jobs

The batch command is used to schedule one-time tasks. However, instead of specifying an execution time, batch commands are executed as soon as the system's load average drops below 0.8 (80% of CPU usage). The default value of 0.8 can be changed by using the –l option to the atd command.

- batch (interactive)
- batch -f file-tasks
- uptime , top , htop

```
#12
$ uptime
 13:04:28 up  1:53,  1 user,  load average: 0.36, 0.54, 0.55

$ top
```

```
top - 13:04:55 up  1:54,  1 user,  load average: 0.38, 0.53, 0.54
Tasks: 256 total,   1 running, 255 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.5 us,  1.6 sy,  0.0 ni, 94.0 id,  0.2 wa,  0.3 hi,  0.2 si,  0.0 st
MiB Mem :   7468.9 total,   1403.5 free,   2418.6 used,   3646.8 buff/cache
MiB Swap:   7628.0 total,   7628.0 free,      0.0 used.   3995.2 avail Mem
<Enter> to resume, filters: none
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1881 ecanet    20   0 1376560  79164  46200 S   4.7   1.0   3:14.54 Xorg
   2071 ecanet    20   0 4656452 192276 103988 S   3.7   2.5   3:25.43 gnome-shell
   5049 ecanet    20   0  613956  50800  36700 S   3.7   0.7   0:20.87 gnome-terminal-
```

```
#13
# batch -f atjobs
batch accepts no parameters

# batch < atjobs
warning: commands will be executed using /bin/sh
job 7 at Mon Nov  1 13:07:00 2021

# batch
warning: commands will be executed using /bin/sh
at> who > /tmp/users.log
at> <EOT>
job 8 at Mon Nov  1 13:07:00 2021

# atq
8       Mon Nov  1 13:07:00 2021 b root
```

Systemd Timer Units

The creation of cron dates back to 1975, which makes it much older than systemd timers. While cron is more widely-available across systems, including non-Linux systems (such as BSD systems), systemd timers have some advantages from what has been learned from years and years of usage of cron:

- Systemd timers can be set to run in a specific environment.
- With systemd timers, dependencies can be configured for jobs, meaning that scheduled jobs can be set to be dependent on other systemd units.
- Systemd timer jobs are logged via the systemd journal, providing output in a centralized place and format.

Timers are essentially files that end in .timer, which fall under a category of files called systemd unit files. The .timer files will contain configuration information about the task to be executed by the systemd timer.

```
#14
# cat /usr/lib/systemd/system/logrotate.timer
[Unit]
Description=Daily rotation of log files
Documentation=man:logrotate(8) man:logrotate.conf(5)

[Timer]
OnCalendar=daily
AccuracySec=1h
Persistent=true

[Install]
WantedBy=timers.target
```

```
## greeting.timer example
[Unit]
Description=Displays greeting after boot

[Timer]
OnBootSec=10sec
Unit=greeting.service

[Install]
WantedBy=multi-user.target
```

Types of timers:

- monotonic
- realtime

monotonic

Systemd timer allows a job to be executed after an event has occurred. This type can be used to run a job when the system boots (OnBootSec option) or a systemd unit is active (OnActiveSec option).

realtime

Realtime timers work like cron and execute a job when a specified time has occurred. To create a realtime timer, the OnCalendar option should be used in the [Timer] section of the .timer file. The format of an OnCalendar time entry is: DayofWeek Year-Month-Day Hour:Minute:Second

A .timer unit file should correspond to a systemd service, which is a .service file with the same name. For example, the greeting.timer unit file created above should have a corresponding service file called greeting.service. When the systemd timer is activated, the .service systemd unit is executed.

```
#15
# systemctl  list-timers
NEXT                         LEFT          LAST                         PASSED       UNIT                         ACTIVATES              >
Mon 2021-11-01 15:08:44 CET 1h 43min left Mon 2021-11-01 13:12:21 CET 13min ago    dnf-makecache.timer          dnf-makecache.service  >
Tue 2021-11-02 00:00:00 CET 10h left      n/a                          n/a          logrotate.timer              logrotate.service      >
Tue 2021-11-02 00:00:00 CET 10h left      n/a                          n/a          mlocate-updatedb.timer       mlocate-updatedb.servic>
Tue 2021-11-02 00:00:00 CET 10h left      n/a                          n/a          unbound-anchor.timer         unbound-anchor.service >
Tue 2021-11-02 11:26:21 CET 22h left      Mon 2021-11-01 11:26:21 CET 1h 59min ago systemd-tmpfiles-clean.timer systemd-tmpfiles-clean.>
Mon 2021-11-08 00:00:00 CET 6 days left   n/a                          n/a          fstrim.timer                 fstrim.service         >
6 timers listed.

# systemctl start|stop name.timer
# systemctl enable|disable name.timer
```

Transient jobs

- systemd-run
- systemctl list-timers

The systemd-run command can be used to run a command or execute a systemd .service unit. Is another way to schedule jobs, using systemd-run.

```
#16
# systemd-run --on-active=1h /bin/touch /tmp/fie
Running timer as unit: run-r5847ed43fa8e44bdbf40ce5cfdd19fa0.timer
Will run service as unit: run-r5847ed43fa8e44bdbf40ce5cfdd19fa0.service

# systemctl list-timers
NEXT                         LEFT          LAST                         PASSED      UNIT                                         ACTIVATES>
Mon 2021-11-01 14:29:39 CET 59min left    n/a                          n/a         run-r5847ed43fa8e44bdbf40ce5cfdd19fa0.timer run-r5847>
Mon 2021-11-01 15:08:44 CET 1h 38min left Mon 2021-11-01 13:12:21 CET 17min ago   dnf-makecache.timer                          dnf-makec>
Tue 2021-11-02 00:00:00 CET 10h left      n/a                          n/a         logrotate.timer                              logrotate>
Tue 2021-11-02 00:00:00 CET 10h left      n/a                          n/a         mlocate-updatedb.timer                       mlocate-u>
Tue 2021-11-02 00:00:00 CET 10h left      n/a                          n/a         unbound-anchor.timer                         unbound-a>
Tue 2021-11-02 11:26:21 CET 21h left      Mon 2021-11-01 11:26:21 CET 2h 3min ago systemd-tmpfiles-clean.timer              systemd-t>
Mon 2021-11-08 00:00:00 CET 6 days left   n/a                          n/a         fstrim.timer                                 fstrim.se>
7 timers listed.

# systemd-run --on-active=1h --unit=greeting.service

# systemd-run --on-calendar='2019-10-06 11:30' date
```

Cron Examples

Example 1: backup.sh

```
DIR=$1
RETENCIO=$2
OCUPACIO=$(du -s -k $DIR | cut -f1)
LLIURE=$(df | grep -w "/" | tr -s " " | cut -f4 -d" ")
if [ $LLIURE -le $OCUPACIO ]; then
        echo "Error $NOMPROG: no hi ha espai suficient per fer la còpia."
        exit 6
fi

# Procediment de còpia
tar -zcPf $DIRDESTI/$DIR-$(date +%y%m%d).tgz -C / $DIR 2>/dev/null
if [ $? -ne 0 ]; then
        echo "Error $NOMPROG: la còpia de seguretat de $DIR ha fallat" >> /dev/stderr
         exit 7
fi
echo "$NOMPROG: realitzada correctament la copia de seguretat de $DIR."

# Procediment de purga
NAME=$(basename $DIR)
find $DIRDESTI/ -name $NAME*.tgz -type f -mtime +$RETENCIO -delete
if [ $? -eq 0 ]; then
        echo "$NOMPROG: realitzada correctament la purga de les còpies de seguretat de
$DIR."
else
        echo "Error $NOMPROG: error eliminant les còpies de seguretat antigues de $DIR"
        exit 8
fi
exit 0
```

Example 2:

```
Configureu el cron de l'usuari root perquè es faci l'execució automàtica del shell
script copia.sh tenint en compte el següent:
    • Copieu el directori /etc i especifiqueu retenció de dades de 3.
    • Redirigiu la sortida estàndard i la d'error del procés a un fitxer anomenat
```

```
/backups/backup_aammdd.log (aammdd correspon als dos dígits de l'any, el mes i el dia del moment que es fa la còpia).
```

```
5 * * * * /usr/sbin/copia.sh  /etc 3 &>> /backups/backup_$(date +\%y\%m\%d).log
```

Example 3:

```
Planifiqueu amb el cron del sistema una tasca que deixi constància de cada vegada que el servidor es reinicialitza i guardi els resultats al fitxer reboot.log  al directori /var/log
```

```
@reboot root date >> /var/log/reboot.log
```

Example 4:

```
Planifiqueu amb el cron del sistema, els divendres a la nit just al primer minut de la matinada de dissabte, un reinici automàtic del sistema. Heu d'afegir la data i hora del reinici a un fitxer de registre anomenat /var/log/reboot.log. Documenteu el procés realitzat  indiqueu clarament quin usuari i quin fitxer utilitzeu.
```

```
/etc/crontab
1 0 * * 6 root date >> /var/log/reboot.log && /sbin/reboot
```

Example 5:

```
Planifiqueu amb el cron del vostre usuari una tasca que deixi constància cada cinc minuts en l'horari de 16h a 22h de dilluns a divendres de la quantitat de processos que s'estan executant. S'ha de guardar els resultats al fitxer processos.log  del vostre directori personal (a cada execució, afegeix els resultats al final del fitxer).
```

```
*/5   16-22   *   *   1-5    /usr/bin/who >> $HOME/usuaris.txt
```

Example 6:

```
lanifiqueu amb el cron del sistema que es realitzi una còpia de seguretat completa comprimida de totes les dades dels directoris personals dels usuaris (/home) al directori /backups en un únic fitxer anomenat backup_usuaris.tgz. Cal generar un fitxer de registre anomenat /var/log/backup_usuaris.log amb el llistat dels fitxers que s'han emmagatzemat només en l'última còpia. Aquest backup es realitza en una periodicitat setmanal cada diumenge a les dues de la matinada de la nit del dissabte al diumenge).
```

```
0   2   *   *   7 root   /bin/tar cvzf /backups/backup_usuaris.tgz -C /home/ > /var/log/backup_usuaris.log
```

Example Exercises

[cron]

1. Create a personal periodic task calculating the disc usage of the user's home directory, each day at 8 and at 16h from monday to friday. Generate the report adding the information in the file /tmp/du-report.log
2. List the cron tasks.
3. List the /var/spool/cron directory.
4. Create a system task every 2 hours from january to june from monday to friday reporting the disk free information. Add the information to the file /tmp/df-report.log.
5. List the cron tasks.

[at & batch]

6. Create a job 5 minutes from now generating the list of connected users to /tmp/who.log.
7. Create a job at a specified date and time listing the disk free information, users connected and memory report. Save the information at /tmp/report.log
8. List the scheduled jobs
9. Remove the last job.
10. Create a job whenever the system is available generating the file /tmp/tree.log containing all the directory tree.

[others]

11. Realitza els exercicis indicats a: 107.2 Automate system administration tasks by scheduling jobs
12. Realitza els exercicis indicats a: 107.2 Automate system administration tasks by scheduling jobs
13. Realitza els exercicis del Question-Topics 107.2.