systemd/Timers

< Systemd

Timers are **systemd** unit files whose name ends in `.timer` that control `.service` files or events. Timers can be used as an alternative to **cron** (read **#As a cron replacement**). Timers have built-in support for calendar time events, monotonic time events, and can be run asynchronously.

Related articles

systemd

systemd/User

systemd FAQ

cron

Contents

Timer units

Timers are *systemd* unit files with a suffix of `.timer`. Timers are like other **unit configuration files** and are loaded from the same paths but include a `[Timer]` section which defines when and how the timer activates. Timers are defined as one of two types:

- **Realtime timers** (a.k.a. wallclock timers) activate on a calendar event, the same way that cronjobs do. The option `OnCalendar=` is used to define them.
- **Monotonic timers** activate after a time span relative to a varying starting point. They stop if the computer is temporarily suspended or shut down. There are number of different monotonic timers but all have the form: `On*Type*Sec=`. Common monotonic timers include `OnBootSec` and `OnActiveSec`.

For a full explanation of timer options, see the **systemd.timer(5) (https://jlk.fjfi.cvut.cz/arch/manpages/man/systemd.timer.5)**. The argument syntax for calendar events and time spans is defined in **systemd.time(7) (https://jlk.fjfi.cvut.cz/arch/manpages/m** an/systemd.time.7).

Service unit

For each `.timer` file, a matching `.service` file exists (e.g. `foo.timer` and `foo.service`). The `.timer` file activates and controls the `.service` file. The `.service` does not require an `[Install]` section as it is the *timer* units that are enabled. If necessary, it is possible to control a differently-named unit using the `Unit=` option in the timer's `[Timer]` section.

Management

To use a *timer* unit **enable** and **start** it like any other unit (remember to add the `.timer` suffix). To view all started timers, run:

```
$ systemctl list-timers
```

```
NEXT                          LEFT        LAST                          PASSED     UNIT                         ACTIVATES
Thu 2014-07-10 19:37:03 CEST  11h left    Wed 2014-07-09 19:37:03 CEST  12h ago    systemd-tmpfiles-clean.timer systemd-tmpfiles-clean.service
Fri 2014-07-11 00:00:00 CEST  15h left    Thu 2014-07-10 00:00:13 CEST  8h ago     logrotate.timer              logrotate.service
```

Note:

- To list all timers (including inactive), use `systemctl list-timers --all` .

- The status of a service started by a timer will likely be inactive unless it is currently being triggered.
- If a timer gets out of sync, it may help to delete its `stamp-*` file in `/var/lib/systemd/timers`. These are zero length files which mark the last time each timer was run. If deleted, they will be reconstructed on the next start of their timer.

Examples

A service unit file can be scheduled with a timer out-of-the-box. The following examples schedule `foo.service` to be run with a corresponding timer called `foo.timer`.

Monotonic timer

A timer which will start 15 minutes after boot and again every week while the system is running.

/etc/systemd/system/foo.timer

```
[Unit]
Description=Run foo weekly and on boot

[Timer]
OnBootSec=15min
OnUnitActiveSec=1w

[Install]
WantedBy=timers.target
```

Realtime timer

A timer which starts once a week (at 12:00am on Monday). When activated, it triggers the service immediately if it missed the last start time (option `Persistent=true`), for example due to the system being powered off:

```
/etc/systemd/system/foo.timer
```

```
[Unit]
Description=Run foo weekly

[Timer]
OnCalendar=weekly
Persistent=true

[Install]
WantedBy=timers.target
```

When more specific dates and times are required, `OnCalendar` events uses the following format:

```
DayOfWeek Year-Month-Day Hour:Minute:Second
```

An asterisk may be used to specify any value and commas may be used to list possible values. Two values separated by `..` indicate a contiguous range.

In the below example the service is run the first four days of each month at 12:00 PM, but *only* if that day is a Monday or a Tuesday.

```
OnCalendar=Mon,Tue *-*-01..04 12:00:00
```

To run a service on the first Saturday of every month, use:

```
OnCalendar=Sat *-*-1..7 18:00:00
```

More information is available in `systemd.time(7) (https://jlk.fjfi.cvut.cz/arch/manpages/man/systemd.time.7)`.

Tip:

- `OnCalendar` time specifications can be tested in order to verify their validity and to calculate the next time the condition would elapse when used on a timer unit file with the `calendar` option of the *systemd-analyze* utility. For example, one can use `systemd-analyze calendar weekly` or `systemd-analyze calendar "Mon,Tue *-*-01..04 12:00:00"`.
- Special event expressions like `daily` and `weekly` refer to *specific start times* and thus any timers sharing such calendar events will start simultaneously. Timers sharing start events can cause poor system performance if the timers' services compete for system resources. The `RandomizedDelaySec` option in the `[Timer]` section avoids this problem by randomly staggering the start time of each timer. See `systemd.timer(5) (https://jlk.fjfi.cvut.cz/arch/manpages/man/systemd.timer.5)`.

Transient .timer units

One can use `systemd-run` to create transient `.timer` units. That is, one can set a command to run at a specified time without having a service file. For example the following command touches a file after 30 seconds:

```
# systemd-run --on-active=30 /bin/touch /tmp/foo
```

One can also specify a pre-existing service file that does not have a timer file. For example, the following starts the systemd unit named `someunit.service` after 12.5 hours have elapsed:

```
# systemd-run --on-active="12h 30m" --unit someunit.service
```

See systemd-run(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/systemd-run.1) for more information and examples.

As a cron replacement

Although **cron** is arguably the most well-known job scheduler, *systemd* timers can be an alternative.

Benefits

The main benefits of using timers come from each job having its own *systemd* service. Some of these benefits are:

- Jobs can be easily started independently of their timers. This simplifies debugging.
- Each job can be configured to run in a specific environment (see `systemd.exec(5) (https://jlk.fjfi.cvut.cz/arch/manpages/man/systemd.exec.5)`).
- Jobs can be attached to **cgroups**.
- Jobs can be set up to depend on other *systemd* units.
- Jobs are logged in the *systemd* journal for easy debugging.

Caveats

Some things that are easy to do with cron are difficult to do with timer units alone:

- Creation: to set up a timed job with *systemd* you need to create two files and run `systemctl` commands, compared to adding a single line to a crontab.
- Emails: there is no built-in equivalent to cron's `MAILTO` for sending emails on job failure. See the next section for an example of setting up a similar functionality using `OnFailure=`.

MAILTO

You can set up systemd to send an e-mail when a unit fails. Cron sends mail to `MAILTO` the job outputs to stdout or stderr, but many jobs are setup to only output on error. First you need two files: an executable for sending the mail and a *.service* for starting the executable. For this example, the executable is just a shell script using `sendmail`:

```
/usr/local/bin/systemd-email
-----------------------------
#!/bin/bash

/usr/bin/sendmail -t <<ERRMAIL
To: $1
From: systemd <root@$HOSTNAME>
Subject: $2
Content-Transfer-Encoding: 8bit
Content-Type: text/plain; charset=UTF-8

$(systemctl status --full "$2")
ERRMAIL
```

Whatever executable you use, it should probably take at least two arguments as this shell script does: the address to send to and the unit file to get the status of. The *.service* we create will pass these arguments:

```
/etc/systemd/system/status-email-user@.service
-----------------------------
[Unit]
Description=status email for %i to user

[Service]
Type=oneshot
ExecStart=/usr/local/bin/systemd-email address %i
User=nobody
Group=systemd-journal
```

Where `user` is the user being emailed and `address` is that user's email address. Although the recipient is hard-coded, the unit file to report on is passed as an instance parameter, so this one service can send email for many other units. At this point you can **start** `status-email-user@dbus.service` to verify that you can receive the emails.

Then simply **edit** the service you want emails for and add `OnFailure=status-email-user@%n.service` to the `[Unit]` section. `%n` passes the unit's name to the template.

Note:

- If you set up SSMTP security according to **SSMTP#Security** the user `nobody` will not have access to `/etc/ssmtp/ssmtp.conf`, and the `systemctl start status-email-user@dbus.service` command will fail. One solution is to use `root` as the User in the `status-email-user@.service` unit.
- If you try to use `mail -s somelogs address` in your email script, `mail` will fork and systemd will kill the mail process when it sees your script exit. Make the mail non-forking by doing `mail -Ssendwait -s somelogs address`.

Using a crontab

Several of the caveats can be worked around by installing a package that parses a traditional crontab to configure the timers. systemd-cron-next (https://aur.archlinux.org/packages/systemd-cron-next/)AUR and systemd-cron (https://aur.archlinux.org/packages/systemd-cron/)AUR are two such packages. These can provide the missing `MAILTO` feature.

Also, like with crontabs, a unified view of all scheduled jobs can be obtained with `systemctl`. See #Management.

See also

- systemd.timer(5) (https://jlk.fjfi.cvut.cz/arch/manpages/man/systemd.timer.5)
- Fedora Project wiki page (https://fedoraproject.org/wiki/Features/SystemdCalendarTimers) on *systemd* calendar timers
- Gentoo wiki section (https://wiki.gentoo.org/wiki/Systemd#Timer_services) on *systemd* timer services
- **systemd-cron-next** — tool to generate timers/services from crontab and anacrontab files

  https://github.com/systemd-cron/systemd-cron-next || systemd-cron-next (https://aur.archlinux.org/packages/systemd-cron-next/)AUR

- **systemd-cron** — provides systemd units to run cron scripts; using *systemd-crontab-generator* to convert crontabs

  https://github.com/systemd-cron/systemd-cron || `systemd-cron (https://aur.archlinux.org/packages/systemd-cron/)`AUR

Retrieved from "https://wiki.archlinux.org/index.php?title=Systemd/Timers&oldid=509259"