RedHat 2

Day1
Self Study Task1

Nada Mohamed Ahmed Hassan Eleshmawy

Mansoura Open Source

Date : 11/2/2025

Comparison between At Vs Crontab Vs Systemd Timers Vs Anacron

At :

The `at` command is used to schedule a **one-time** job that runs at a specified time. It is useful when you need to execute a task in the future but only once.

How It Works

- Jobs are stored in `/var/spool/at/` until they are executed.

```
nada :>sudo ls /var/spool/at/
a0000101ba52b5  a0000601ba592a  spool
```

- **Once the job runs, it is removed from the queue**.
- If the system reboots before the job executes, it is **lost**.
- No built-in logging (unless manually redirected).

Example :

Run a command after 10 minutes:

echo "echo 'Hello, Nada!' > /tmp/message.txt" | at now + 10 minutes

```
nada :>echo "echo 'Hello, Nada' > /selfStudy/message.txt" | at now + 10 minutes
warning: commands will be executed using /bin/sh
job 7 at Wed Feb 12 20:19:00 2025
nada :>sudo ls /var/spool/at/
a0000101ba52b5  a0000601ba592a  a0000701ba592b  spool
```

View scheduled jobs:

atq

```
nada :>atq
1       Tue Feb 11 16:45:00 2025 a nada_mohamed2243
6       Wed Feb 12 20:18:00 2025 a nada_mohamed2243
7       Wed Feb 12 20:19:00 2025 a nada_mohamed2243
```

Remove a scheduled job:

atrm 6

```
nada :>atrm 6
nada :>atq
1       Tue Feb 11 16:45:00 2025 a nada_mohamed2243
7       Wed Feb 12 20:19:00 2025 a nada_mohamed2243
nada :>sudo ls /var/spool/at/
a0000101ba52b5  a0000701ba592b  spool
nada :>
```

Crontab :

The `cron` daemon allows users to schedule **recurring tasks** based on a predefined schedule. It's useful for automating daily, weekly, or monthly tasks.

```
* * * * *
```

* → Minute
* → Hour
* → Any day of the month
* → Any month
* → Any day of the week

How It Works

- Uses the `crontab` command to define jobs.
- Jobs are stored in `/var/spool/cron/` for each user.

```
nada :>sudo ls /var/spool/cron/
nada_mohamed2243
```

```
nada :>sudo cat /var/spool/cron/nada_mohamed2243
* * * * * ./cronScript.sh >> result
0 2 * * * echo "Hello, Nada!" > /tmp/message.txt
nada :>
```

- Cron jobs run at the exact specified time but **are missed if the system is off**.
- Logging is limited unless manually set up (e.g., redirecting output to a log file).

Example:

Edit the user's crontab:

crontab -e

Add a job that runs every day at **2 AM**:

0 2 * * * echo "Hello, Nada!" > /tmp/message.txt

0 → Minute

2 → Hour

* → Any day of the month

* → Any month

* → Any day of the week

```
nada :>crontab -e
crontab: installing new crontab
```

List scheduled jobs:

crontab -l

```
nada :>crontab -l
* * * * * ./cronScript.sh >> result
0 2 * * * echo "Hello, Nada!" > /tmp/message.txt
nada :>
```

Remove all cron jobs:

crontab -r

systemd timers :

systemd timers offer a powerful and **flexible** alternative to cron, allowing fine-grained control over scheduling. Unlike cron, they support:

- **Logging and debugging via journalctl**.
- **Dependency management** (e.g., ensuring a service runs before or after another).
- **Handling missed executions** when the system reboots.

How It Works

- Uses systemd services (.service) and timers (.timer).
- Jobs can be **oneshot** or **repeated**.
- Timers are configured under /etc/systemd/system/.
- Requires systemd (not available in minimal Linux distributions like Alpine).
- More complex to set up than cron.

Example

Create a systemd service

```
nada :>sudo vi /etc/systemd/system/hello.service
```

(/etc/systemd/system/hello.service):

```
[Unit]
Description=Say Hello

[Service]
Type=oneshot
ExecStart=/bin/echo "Hello, Nada!" >
selfStudy/messageTimer.txt
```

```
nada :>sudo vi /etc/systemd/system/hello.timer
```

Create a systemd timer (/etc/systemd/system/hello.timer):

```
[Unit]
Description=Run Hello Service Every 10 Minutes
```

```
[Timer]
OnCalendar=*:0/10
Persistent=true

[Install]
WantedBy=timers.target
```

Enable and start the timer:

`sudo systemctl enable hello.timer --now`

```
nada :>sudo systemctl enable hello.timer --now
Created symlink /etc/systemd/system/timers.target.wants/hello.timer → /etc/systemd/system/hello.timer.
nada :>
```

Check running timers:

`systemctl list-timers`

```
nada :>systemctl list-timers
NEXT                         LEFT          LAST                         PASSED    UNIT                         ACTIVATES                    >
Wed 2025-02-12 20:50:00 EET 5min left     -                            -         hello.timer                  hello.service
Thu 2025-02-13 00:00:00 EET 3h 15min left Wed 2025-02-12 20:02:25 EET 42min ago logrotate.timer              logrotate.service
Thu 2025-02-13 00:00:00 EET 3h 15min left Wed 2025-02-12 20:02:25 EET 42min ago mlocate-updatedb.timer       mlocate-updatedb.service
Thu 2025-02-13 20:17:27 EET 23h left      Wed 2025-02-12 20:17:27 EET 27min ago systemd-tmpfiles-clean.timer systemd-tmpfiles-clean.serv>
-                            -             Wed 2025-02-12 20:44:00 EET 29s ago   dnf-makecache.timer          dnf-makecache.service

5 timers listed.
Pass --all to see loaded but inactive timers, too.
lines 1-9/9 (END)
```

Note :

Systemd links **`hello.timer`** → **`hello.service`** because they share the **same base name**.

The timer schedules and automatically starts the service at the specified interval.

You can check logs with `journalctl -u hello.service`.

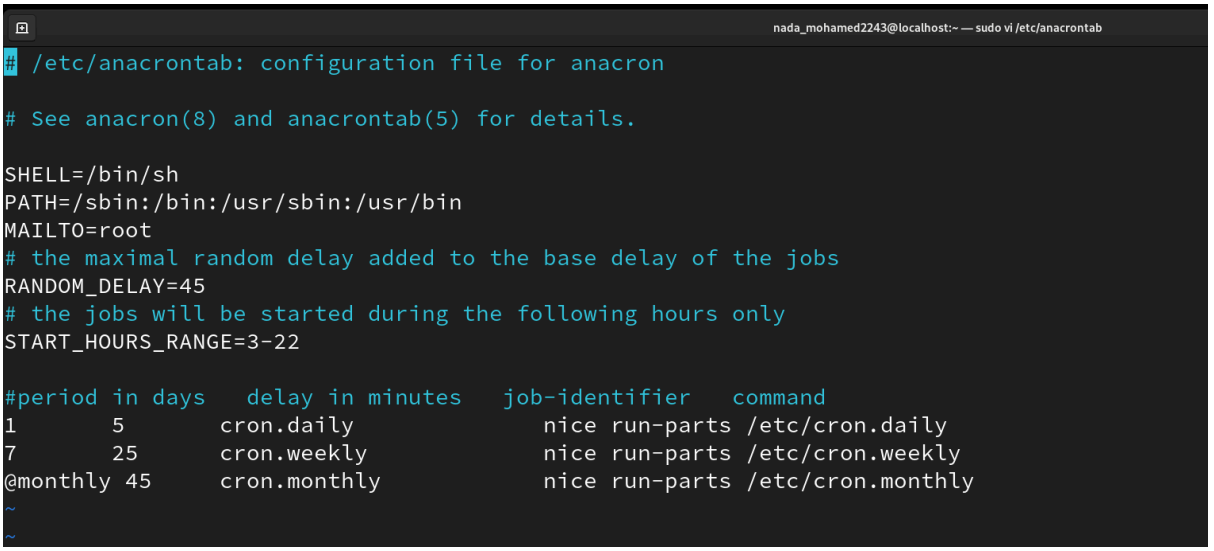
```
nada :>journalctl -u hello.service --no-pager --since "10 minutes ago"
Feb 12 21:20:01 localhost.localdomain systemd[1]: Starting Say Hello...
Feb 12 21:20:01 localhost.localdomain echo[4182]: Hello, Nada! > /selfStudy/messageTimer.txt
Feb 12 21:20:01 localhost.localdomain systemd[1]: hello.service: Deactivated successfully.
Feb 12 21:20:01 localhost.localdomain systemd[1]: Finished Say Hello.
nada :>
```

Anacron :

`anacron` is useful when you want **scheduled tasks to run even if the system was off** at the scheduled time. Unlike `cron`, `anacron` will **execute missed jobs** as soon as possible after boot.

How It Works

- Works only with **daily or less frequent jobs** (not for minute-based tasks).
- Runs scheduled jobs as soon as the system is powered on.
- Uses `/etc/anacrontab` for configuration.



- Does **not** support per-minute scheduling (only daily and above).
- Designed for **root** users (not for user-specific jobs).

Example

Edit `/etc/anacrontab` and add:

```
1 5 hello-job /bin/sh -c 'echo "Hello, Nada!" > /selfStudy/messageAnacron.txt'
```

1 → Runs **once per day**.

5 → Runs **5 minutes after system boot** (if missed).

hello-job → Job identifier (for logging).

/bin/sh -c 'echo "Hello, Nada!" > /selfStudy/messageAnacron.txt' → Runs the command using sh, which properly handles output redirection (>).

```
nada :>sudo anacron -d -f
[sudo] password for nada_mohamed2243:
Anacron started on 2025-02-12
Job `cron.daily' locked by another anacron - skipping
Job `cron.weekly' locked by another anacron - skipping
Will run job `hello-job' in 6 min.
Will run job `cron.monthly' in 46 min.
```