System and Network Engineering - Lecture 8

$ Scheduling tasks (cron, anacron, at)

Scheduling tasks - Cron jobs

- **Cron** - is the common name for the service to schedule tasks such as regular backups, system updates etc. The cron daemon (cron.service) is the background service that enables cron functionality.

```
saitanov@linuxPC:~$ systemctl status cron
● cron.service - Regular background program processing daemon
     Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2022-10-16 00:48:46 +04; 39min ago
       Docs: man:cron(8)
   Main PID: 639 (cron)
      Tasks: 1 (limit: 4610)
     Memory: 812.0K
        CPU: 20ms
     CGroup: /system.slice/cron.service
             └─639 /usr/sbin/cron -f -P
```

Cron

The cron daemon checks for special files called "**crontabs**" as follows:

1. `/var/spool/cron/crontabs/` - individual user tasks
   - `$crontab -e` - put tasks here under current user
   - `$crontab -l` - display current tasks for the user in std output
   - `$crontab -u <username>` - run crontab under specific user (require root privileges)

```
saltanov@linuxPC:~$ sudo crontab -u test -l
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

30 0 * * * /usr/bin/echo "Hello!"
saltanov@linuxPC:~$ sudo ls -al /var/spool/cron/crontabs/
total 12
drwx-wx--T 2 root crontab 4096 Oct 16 02:28 .
drwxr-xr-x 3 root root    4096 Aug  9 15:48 ..
-rw------- 1 test crontab  481 Oct 16 02:28 test
```

Cron

The cron daemon checks for special files called "**crontabs**" as follows:

2. `/etc/cron.d/` - placement for system services and applications that will put crontabs there

```
saltanov@linuxPC:/etc/cron.d$ ll
total 28
drwxr-xr-x   2 root root  4096 Aug 28 16:19 ./
drwxr-xr-x 130 root root 12288 Oct 16 01:14 ../
-rw-r--r--   1 root root   219 Oct  9  2021 anacron
-rw-r--r--   1 root root   201 Jan  9  2022 e2scrub_all
-rw-r--r--   1 root root   102 Mar 23  2022 .placeholder
```

3. `/etc/crontab` - system-wide tasks, usually only used by root user or daemons to configure system wide jobs.

```
saltanov@linuxPC:/etc/cron.d$ ll /etc/crontab
-rw-r--r-- 1 root root 1136 Mar 23  2022 /etc/crontab
```

Cron

To configure system wide jobs with the *crontab* file:

- First, the environment must be defined. **SHELL**=/bin/bash. If the SHELL line is omitted, cron will use the default, which is **sh**
- **PATH**=/where/the/executable. If the PATH variable is omitted, no default will be used and file location to be executed will need to have an absolute path
- **HOME**=/where/the/app. If HOME is omitted, cron will use the invoking user home directory. Some programs require additional files to be read for execution and use $HOME env.
- **MAILTO**="" value can be empty, root or contains particular email where to send notifications

Cron

By the initial design *crontab* file will run as follows:

- `/etc/cron.*/ {cron.hourly|cron.daily|cron.weekly|cron.monthly}` - in the `/etc/crontab` file, cron will run scripts timely in accordance with the directories.
- Scripts should be defined as executables without *.sh extension as they will be processed by `run-parts` command that takes directory as the argument where those executables are

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

Crontab syntax

```
* * * * * command to be executed
- - - - -
| | | | |
| | | | ----- Day of week (0 - 7) (Sunday=0 or 7)
| | | ------- Month (1 - 12)
| | --------- Day of month (1 - 31)
| ----------- Hour (0 - 23)
------------- Minute (0 - 59)
```

- For system-wide tasks in `/etc/crontab` file USERNAME field should be defined

```
1 2 3 4 5 USERNAME /path/to/command arg1 arg2
```

- If no user is specified, the job is run as the user that owns the crontab file, generally is root.

Crontab syntax

- You can use special strings to provide scheduled time

Special string	Meaning
@reboot	Run once, at startup.
@yearly	Run once a year, “0 0 1 1 *”.
@annually	(same as @yearly)
@monthly	Run once a month, “0 0 1 * *”.
@weekly	Run once a week, “0 0 * * 0”.
@daily	Run once a day, “0 0 * * *”.
@midnight	(same as @daily)
@hourly	Run once an hour, “0 * * * *”.

Cron

Limit access to the crontab executable (cron jobs)

- Based on existence of /etc/cron.allow and /etc/cron.deny, user is allowed or denied to edit the crontab in below sequence:
  - If cron.allow exists - only users listed into it can use crontab
  - If cron.allow does not exist - all users except the users listed into cron.deny can use crontab
  - If neither of the file exists - only the root can use crontab
  - If a user is listed in both cron.allow and cron.deny - that user can use crontab.

```
saltanov@linuxPC:/etc/cron.d$ cat /etc/cron.deny
test
test2

saltanov@linuxPC:/etc/cron.d$ sudo su - test -c "crontab -e"
You (test) are not allowed to use this program (crontab)
See crontab(1) for more information
```

Cron

Creating backup installed cron jobs entries:

- `$crontab -l > /backup/cron/users.current.backup` - current user
- `$crontab -u <username> -l >/backup/cron/users.userName.backup` - another user

Cron jobs log file:

- You can check status and verify your passed cron job by looking in the log file with using `journalctl`
  - `$crontab -u <username> -l >/backup/cron/users.<username>.backup` - another user
- Analyze manually in the log
  - `$cat /var/log/syslog | grep cron`
  - Setting up separately `cron.log` file in `/var/log/cron.log`
    - edit `/etc/rsyslog.d/50-default.conf` configuration file

Cron - common questions

1. How to execute a Linux cron job every second using Crontab?

   Answ: Can not. In cron, the minimum unit you can specify is minute.

2. How to execute a Linux command after every reboot using cron?

   Answ: Using the @reboot cron keyword. This will execute the specified command once after the machine got booted every time. @reboot <command>

3. If there are few crons configured for a particular user and its password gets expired, will the cron jobs continue to run?

   Answ: No. The cron jobs will stop running.

4. If there are some crons configured for a user which are running fine, the root user put that particular user in /etc/cron.deny list. Will the crons already configured for the user continue to run?

   Answ: Yes, crons will continue to run but the user will not be able to edit, view or remove its crontab entries.

Anacron syntax

Daemon (anacron.service) - performs the same function as cron, but it adds the ability to run jobs that were skipped, such as if the computer was off.

- running jobs in accordance with the schedule (e.g. daily, weekly or monthly)

```
SHELL=/bin/sh
HOME=/root
LOGNAME=root
START_HOURS_RANGE=3-22
RANDOM_DELAY=30

# These replace cron's entries
1       5       cron.daily      run-parts --report /etc/cron.daily
7       10      cron.weekly     run-parts --report /etc/cron.weekly
@monthly        15      cron.monthly    run-parts --report /etc/cron.monthly

#period delay job-identifier command
@daily 10 example.daily /bin/bash /home/aron/bin/backup.sh
```

- There are 2 important additional parameters could be added:
  - *START_HOURS_RANGE=3-22 - variable sets the time frame, when the job could started*
  - *RANDOM_DELAY=30 - minutes will be added to the start up delay of the jobs.*

The syntax of anacron is as follows:

- **period** - is the frequency of the task execution, specified in days or as @daily, @weekly, or @monthly for once a day, week, or month, respectively. You can also use numbers: 1 for daily, 7 for weekly, 30 for monthly, and N for the number of days.
- **delay** - is the number of minutes to wait before executing the job.
- **job-id -** is the name for the job, as will be recorded in the log files.

Anacron

The scheme of the anacron work is the following:

- On Debian systemd-based systems, anacron daemon as defined in `/lib/systemd/system/anacron.service` will run jobs in accordance with the schedule configured in `/lib/systemd/system/anacron.timer` - this file provides systemd timer for anacron. By default the service is triggered hourly through systemd timer:

```
saltanov@linuxPC:/etc/cron.daily$ cat /lib/systemd/system/anacron.timer
[Unit]
Description=Trigger anacron every hour

[Timer]
OnCalendar=*-*-* 07..23:30
RandomizedDelaySec=5m
Persistent=true

[Install]
WantedBy=timers.target
```

Anacron

- to check status of scheduled tasks, anacron uses special file where timestamps are stored in `/var/spool/anacron/`. It compares timestamps with the local time and if they are not updated then anacron will read `/etc/anacrontab` configuration file to perform its scheduled tasks.

```
saltanov@linuxPC:/etc/cron.daily$ tree /var/spool/anacron/
/var/spool/anacron/
├── cron.daily
├── cron.monthly
└── cron.weekly
```

- Anacron will run jobs as they defined in the `/etc/anacrontab` file (e.g. daily, weekly or monthly)

```
SHELL=/bin/sh
HOME=/root
LOGNAME=root
START_HOURS_RANGE=3-22
RANDOM_DELAY=30

# These replace cron's entries
1       5       cron.daily      run-parts --report /etc/cron.daily
7       10      cron.weekly     run-parts --report /etc/cron.weekly
@monthly        15      cron.monthly    run-parts --report /etc/cron.monthly

#period delay job-identifier command
@daily 10 example.daily /bin/bash /home/aron/bin/backup.sh
```

Anacron

- One of the first task will be as defined by the script in `0anacron` file in `/etc/cron.{daily,weekly,monthly}` directories. `run-parts` first, test anacron executable and on the success run anacron to update the timestamps for daily tasks that it was run

```
saltanov@linuxPC:/etc/cron.daily$ find /etc/cron.{daily,monthly,weekly} -name 0anacron
/etc/cron.daily/0anacron
/etc/cron.monthly/0anacron
/etc/cron.weekly/0anacron
```

```
saltanov@linuxPC:/etc/cron.daily$ cat 0anacron
#!/bin/sh
#
# anacron's cron script
#
# This script updates anacron time stamps. It is called through run-parts
# either by anacron itself or by cron.
#
# The script is called "0anacron" to assure that it will be executed
# _before_ all other scripts.

test -x /usr/sbin/anacron || exit 0
anacron -u cron.daily
```

Anacron/cron/at

- In case anacron is not installed in the system, scheduled jobs (daily, weekly, monthly) can be run by cron as shown below (in default configuration)

```
saltanov@linuxPC:/etc/cron.d$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
# You can also override PATH, but by default, newer versions inherit it from the environment
#PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# 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
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

- If you want to run the job just one time (not repeatedly), use **at** command. The **at** utility reads commands from standard input and executes them at a later time
  - `$at 09:00 -f /home/script.sh`