57 - Run jobs(programs/scripts) Regularly with `cron`

- **The `cron`/`crontab` service**
  `cron` executes specific commands in *`crontabs`* on a regular basis based on configuration created by root or users.
- **Types of crontabs**
  Crontabs are the configuration files read by the `crond` Daemon definning which jobs should be run when.
  - **Users crontabs**
    The users crontabs are created by the user issuing the command `crontab -e`.
    It is not recommended to edit the user crontabs using an editor directly instead of passing by the above command. Each user gets their own crontab file.
    Once written and saved, users crontabs are located in:
    `/var/spool/cron/tabs/`*`username`* (SuSE)
    `/var/spool/cron/crontabs/`*`username`* (RedHat, Debian, etc) **?**
    These crontab files contain only 6 fields.
  - **System wide crontab file: `/etc/crontab`**
    The file `/etc/crontab` is used as the main system crontab.
    In this file `crond` Daemon will execute the scheduled tasks and run them as the user specified for that task. For that an extra 'user' field (field 6) is used.
    Therefore this crontab file contains 7 fields.
  - **The `/etc/cron.d/` directory**
    This directory contains extra contabs of the same format as the `/etc/crontab` and will be recognized by `crond`.
  - **The `/etc/cron.{hourly,daily,weekly,monthl}` directories**
    These directories contain scripts that will be run regularly according to their names. They are not directly read by `crond` but are usually run from `/etc/crontab`.
  - **File format of the user contab**
    This file may contain the following types of entries:
    - Comments. These lines start with a '#'
    - Environment variables definitions. eg. MAILTO=michel
    - user crontab entries: made of 6 fields per line:

minute	hour	day of the month	month	day of the week	Command
`0-59`	`0-23`	`1-31`	`1-12`	`0-7`(1=monday)	`/home/michel/bin/myscript`
			jan	mon	
			feb	tue	
			et...	etc..	

  - **File format of the system crontabs**
    This file format applies to `/etc/crontab` and `/etc/cron.d/`*`xxxxxx`*
    This file may contain the following types of entries:
    - Comments. These lines start with a '#'
    - Environment variables definitions. eg. MAILTO=admin
    - user cron schedule entries: made of 7 fields per line:

minute	hour	day of month	month	day of the week	[username]	Command
`0-59`	`0-23`	`1-31`	`1-12`	`0-7`(1=monday)	`root`	`/usr/lib/cron/run-crons`
			jan	mon		
			feb	tue		
			et...	etc..		

Rules for CRONTAB files

- Cron jobs are checked every minute by the `crond` daemon.
- A Crontab file can be a system or a user crontab.
- System crontabs include the username field and the users crontabs don't.
- A crontab entry(line) will be either a comment, an environment setting or a cron command.
- A line starting with `#` is considered a comment.
- An environment setting is in the form: ***`VariableName = value`***
- Each cron entry can be very long but MUST exist on a single physical line.
- If the first character of a crontab line is a '`-`' then cron will not send a message to syslog each time the command is executed, otherwise it will.
- A **`*`** in a field means the command is executable on every instance possible of that field.
- Days of the week are numbers (0 to 7)and start on sunday(0 or 7)(monday=1).
  Names of the weekdays can also be used: `mon,tue,wed,thu,fri,sat,sun`
  Months can also be expressed with:
  `jan,eb,mar,apr,may,june,july,aug,sept,oct,nov,dec`
- '`/`' char. with a number, defines the steps size (default = 1).
  eg. `*/10` is every 10 Minutes `10-18/2` is from 10:00 to 18:00 at every 2 hours
- Hours field is using the range of 0-23
- Any output (incl. errors) from the scheduled jobs are sent to the local user's mail.
  - To avoid to generate any mail then redirect the output (STDERR + STDOUT) of commands to `/dev/null`. eg. `*/2 * * * * ping -c www.suse.de &> /dev/null`
  - or to avoid generating any mail for ALL of the processes started within this crontab: set the variable MAILTO to "". eg. `MAILTO=""`
- The jobs will be executed by `/bin/sh` or the content of the Variable SHELL only if:
  minute & hour & month & (day of the week or day of the month) are matching.

Crontab commands:

`crontab -e`	to create/edit a user crontab(scheduling) file (incl. root) The crontab program will save this file under the user's name in `/var/spool/cron/tabs/` directory. eg. `/var/spool/cron/tabs/joe`
`crontab -l`	Displays user's crontab file.
`crontab -r`	Deletes user's crontab file.
`crontab -e -u` *`username`*	Working on a user's crontab file (need to be root)

Access control of cron scheduling service.

The crontab services can be restricted to certain users and blocked for all others.
This control is done via the files `/etc/cron.allow` and `/etc/cron.deny`.

- If `cron.allow` exists then all users are NOT allowed to use the cron service except the ones listed in `cron.allow`. `cron.deny` if it exists is then ignored.

- If `cron.allow` doesn't exist and the `cron.deny` does exist, then all users are allowed to use the cron service except the ones listed in `cron.deny`.

Examples of user crontabs

- **Range of numbers** are expressed with a '**-**' between numbers:

eg.

```
DISPLAY=:0
0  8-18 * * * /usr/X11R6/bin/xmessage "Take a break" & sleep 10 ;\
killall xmessage
```

Displays the message **"Take a break"** every hour from 8 to 18 hours every day.
The message will appear, wait for 10 seconds and be killled.

- **Commands can be grouped together** inside **( )** for redirections of standard output.

eg.

```
0,15,30,45  *  *  *  *  (echo -n "---";date) > /dev/console
```

- **Standard input entries** can be expressed via the '**%**' in the command.
  Any additional '**%**' in the command line produces a new line (like **/n** in bash)
  use **\%** to enter a litteral '**%**':

eg.

```
30 11 31 12 * /usr/bin/wall%Happy new Year!%Lets meet and enjoy!
```

- **Range Interval settings** using '**/**' character.
  When a range of time-time or date-date is given, an interval between repetition can be set with the '**/**' character.

eg.

```
30 6-16/4 * * * /usr/X11R6/bin/xmessage -display :0 "Take medicine"
```

Gets a message **"Take Medicine"** every 4 hours(**/4**) on the half hour (**30**)
This means at : 6:30 , 10:30, 14:30, 16:30 every day

- Other simple examples:

eg1.

```
-0   1    *    *    1-5  updatedb
```

Runs the updatedb program every day, monday to friday at 01:00, and doesn't report to syslog ('**-**' at start of line)

eg2.

```
0,10,20,30     12   *    *    1,2   updatedb
```

Runs the updatedb program every monday and tuesday at: 12:00,12:10,12:20,12:30

eg3: (Username entry is only in /etc/crontab or in /etc/cron.d/*)

```
*/10 *    *    *    *    root ping -c1 www.suse.de
```

Sends a ping once every 10 minutes (***/10**)to www.suse.de as user **root**

The anacron service.

Anacron is a similar service like cron, except that its frequency is expressed in days and not in minutes. Unlike cron, it does not assume that the machine is running continuously. Hence, it can be used on machines that aren't running 24 hours a day, to control daily, weekly, and monthly jobs that are usually controlled by cron.
When executed, Anacron reads a list of jobs from a configuration file, normally /etc/anacrontab. This file contains the list of jobs that Anacron controls. Each job entry specifies a period in days, a delay in minutes, a unique job identifier, and a shell command.
For each job, Anacron checks whether this job has been executed in the last *n* days, where *n* is the period specified for that job. If not, Anacron runs the job's shell command, after waiting for the number of minutes specified as the delay parameter. After the command exits, Anacron records the date in a special timestamp file for that job, so it can know when to execute it again. Only the date is used for the time calculations. The hour is not used.

The file /etc/anacrontab

The file /etc/anacrontab describes the jobs controlled by anacron.
Its lines can be of four kinds:
- job-description lines, environment assignments, empty lines or comments.
- Environment assignments, blank line sand comments are the same as for crontab.

- Job-description lines are of the form:

  period delay job-identifier command

  - The period is specified in days
  - The delay is specified in minutes.
  - The job-identifier can contain any non-blank character, except slashes'/'. It is used to identify the job in Anacron messages, and as the name for the job's timestamp file.
  - The command can be any shell command.

eg.
```
1  5  cron.daily   nice run-parts --report /etc/cron.daily
7  10 cron.weekly  nice run-parts --report /etc/cron.weekly
30 15 cron.monthly nice run-parts --report /etc/cron.monthly
```

In this example, anacron :
- runs all the scripts contained in the directory /etc/cron.daily, (using default nice value) every day as soon as the day has changed after it has waited for 5 minutes (delay).
- runs all the scripts contained in the directory /etc/cron.weekly, every week as soon as the 7th day has come after it has waited for 10 minutes.
- runs all the scripts contained in the directory /etc/cron.monthly, every month as soon as the 30th day has come after it has waited for 15 minutes.

Files involved:

---------- Scheduling files ----------

`/ect/crontab` Main system crontab scheduling file checked every minute by cron daemon(`/usr/sbin/cron`)
Note: The syntax includes the USER field

`/etc/cron.d/` Directory where extra system scheduling files are stored
Note: The syntax includes the USER field

`/var/spool/cron/tabs/`*`username`* (SuSE)

`/var/spool/cron/crontabs/`*`username`* (RedHat,Debian,etc) **?**
Directory the scheduling files are stored and named as *username*
Note: The syntax should NOT include the USER field

---------- Scripts used by scheduling files ----------

`/usr/lib/cron/run-crons` (SuSE)
Script that will check and run the following regular sysadmin scripts. (**`/etc/cron.hourly/....)`**

`/etc/cron.hourly/` Directory where system admin **crontabs scripts** are located
`/etc/cron.daily/` "" "" ""
`/etc/cron.weekly/` "" "" ""
`/etc/cron.monthly/` "" "" ""

`/var/spool/cron/lastrun/` (SuSE)
? (RedHat,Debian,etc)
Directory where empty files are created having the same name as `cron.hourly` etc to reflect the time they were executed last.

---------- Authorization files ----------

`/var/spool/cron/allow` (SuSE)
`/ect/cron.allow` (RedHat,Debian,etc)
List of users that are allowed to create cron jobs.
If present: `deny` is ignored (If at all present) and all the users that are not listed here are NOT allowed.

`/var/spool/cron/deny` (SuSE)
`/ect/cron.deny` (RedHat,Debian,etc)
List of users that are **NOT** allowed to create cron jobs.
If present and `allow` is NOT present, then all users are allowed except the ones listed here.

Graphic interfaces programs to set the cron jobs:

kcron KDE crontab editor from SuSE 7.1 serie '`kpa`'

good:	- Very simple to use. - Good graphic interface. - Overview of all existing users jobs.
Not so good:	- Is limited to every 5 minutes instead of every minute. - No provision for repeat delays (eg. 5-15/2)

kcrontab KDE crontab editor from SuSE CD

good:	- Allows to enter manually the refined settings as desired as long as one knows the syntax.
Not so good:	- Not so intuitive - One must know the crontab syntax to use it.

Cromagnon crontab editor from: `http://www.andrews.edu/~aldy/cromagnon.html`

good:	- Allows increment of one minute
Not so Good:	- Does 24 hrs instead of 0 for midnight - Writes cron tables in some other places as standard - Project is been dropped for future developements

gat crontab editor from: `http://www.cs.duke.edu/~reynolds/gat/`

good:
- Hides the syntax from unexperienced user
- Has a test button (the only one who has that)

Not so good:
- accepts unacceptable entries
- cannot edit already existing jobs. Create or delete only

vcron crontab editor from: `http://www.linux-kheops.com/pub/vcron/vcronGB.html`

good:
- Elegantly hides the crontab syntax from user
- Produces at-jobs

Not so good:
- Doesn't interpret 2-20/6 syntax when read from sys.

tkt crontab editor from: `http://www.spin.net.au/~mich/tkcron`

good:
- helps with understanding the crontab syntax
- Understands 2-22/4 crontab syntax when read

Not so good:
- needs the command `crontab -l > +/.tct` to be done once to work for each user.

webmin module for crontab editing from `http://www.webmin.com`

good:
- Allows to test the job on the spot
- Http based for remote administration

not so good
- Need to install the whole package and run it via browser.