Cron and at jobs

The cron daemon

- Crond is the cron deamon. It is responsible for executing commands on a specified period of time
- It starts when the system boots
- It uses configuration files to determine which commands to run and on which schedule
- Configuration files are placed in /var/spool/cron (they are called crontabs for Cron Tables). Each file is named after the login name of the user who created it.
- The crond daemon runs in the context of the user who created the crontab file
- It uses the sh (BOURNE shell) by default to execute commands. It can be configured to use other shells like bash or ksh

The cron configuration

- You have to use the crontab command to edit it because it notified the cron daemon with the changes and lets it recalculate the time schedule
- It keeps a log file in `/var/log/cron`, which contains the commands that were executed with their time. It's important to consult this file if a cron job did not run on time
- Lines are ignored (commented out) by placing a pound sign (#) at line start
- Each lines has six fields representing the following:
  - Minute – 0 to 59
  - Hour – 0 to 23
  - Day – 1 to 31
  - Month – 1 to 12
  - Weekday – 0 to 6
  - Command/script

Time fields

- A star means *every*. So a star in the hour field means every hour.
- A range can be specified by placing a dash between two integers. For example 1-5 in the weekday field means everyday from Monday through Friday.
- You can specify discrete values by separating them by a comma. For example, 1,10,20 in the day of month field means that days 1, 10, and 20 of the month
- If a weekday and a day of month were specified, the condition will be satisfied on each one of them.

The command field

- You can put a shell script or just a normal shell command in this field
- The environment file (.bash_profile or .profile) does not get read when the command is executed
- For this reason, you can execute the environment file before attempting to run an scripts shells
- The percent sign (%) represents newline in the command line field. For example `mail -s hello ahmed % Welcome to the team` is inputting "Welcome" to the mail command as it's input
- The PATH variable is not available as well, so you can add it manually to the crontab file like this:
  `PATH=/bin:/sbin:/usr/bin:/usr/sbin:/home/ahmed`

The crontab command

- The `crontab -e` command starts editing the cron file using the editor specified in your environment, indicated by the $EDITOR environment variable
- Using `crontab -l` lists the contents of the current crontab file to the standard output.
- If you want to completely remove the crontab file, use `crontab -r`
- The superuser can manage crontab files of other users by specifying the username after the –u command line argument. For example `crontab -e -u ahmad` will let root edit the crontab file of user ahmad.

Cron permissions

- Who can use cron can be specified in `/etc/cron.allow` and `/etc/cron.deny` file
- The `cron.allow` file is checked first. It contains a list of login names who are allowed to use cron. If this file exists, only people mentioned in this file can use crontab.
- The cron.deny is checked only if `cron.allow` file does not exist, the `cron.deny` file is checked. Login names listed in this file are not allowed to use crontab.
- An empty `cron.allow` file denies anyone from using cron. If you want to disable it you have to delete it or rename it.
- Notice that cron permissions are managed by presence of a crontab file in the `/var/spool/cron` directory. That is, if a file with the name of a user is placed in this directory, cron will execute it regardless of whether or not the user is allowed to use cron.

Crontab keywords

- Crontab enables you to use keywords to facilitate time selection. They can be summarized as follows:
  - @yearly: once per year, equivalent 0 0 1 1 *
  - @monthly: once per month , equivalent to 0 0 1 * *
  - @ daily: once per day , equivalent to 0 0 * * *
  - @hourly: once every hour, equivalentto 0 * * * *
  - @reboot: at system startup

Redirect the command output mail

- The crond daemon send an e-mail to the user who initiated the cron job by default, but this can be configured using the MAIL variable inside the crontab file
- For example, to redirect all cron mail to user root add the following line to the crontab file:
  - `MAILTO="root"`
- You can completely disable mail sending by setting the MAIL variable to an empty variable like this:
  `MAILTO=""`

The `at` command

- While cron is used to execute commands on a periodic, repetitive basis, `at` is used to execute commands once at a specific date and time.
- Time and date from command execution can be specified either exclusively like `1 am December 25`, or relative to the current time like `now + 10 min`
- The `at` command parses the date/time string you input and raises a "Garbled Time" error when it fails to parse it
- When you issue an at command you enter into an interactive shell where you can add the commands you want to execute and press CTRL-D when finished

Managing `at` jobs

- To see all the pending at jobs, use the at –l or atq commands
- The output of these commands is the job number, the scheduled execution time and the owner username
- To delete an at job, use `at -d` or `atrm` followed by the job number
- If you want at to accept a script file as the job, use –f. For example
  `at -f myscript.sh now + 30 min`
- This form of at will make the job run even if you are not logged in to the server (same like nohup)
- You can use batch (a form of at command) to execute a job when the CPU load falls below 1.5. You can use it the same way you use at but without adding a scheduled time

Examples of at date and time formats

- Here are some examples and shortcuts that can be used with the at command:
  - `noon`: today at 12:00 PM
  - `midnight`: tomorrow at 12:00 AM
  - `teatime`: today at 4:00 PM
  - `tomorrow`: 24 hours from now
  - `noon/midnight/teatime tomorrow`: same as their respective meanings but tomorrow rather than today
  - `next week`: 7 days from now (at the same time as now)
  - `next wednesday`: the first Wednesday from now at the same time
  - `fri/saturday`...etc: the respective day of the week at the same time as now
- You can also use now followed by a duration that ranges from minutes to years. For example:
  - `now + 1 minute/hour/day/week/month/year`

The at command permissions

- It is determined by /etc/at.allow and /etc/at.deny files
- If the at.allow file exists, only users in this file are allowed to use the at command
- If the at.allow does not exist, the at.deny file will be checked (if it does exist). Users mentioned in this file are not allowed to use the at command.
- The superuser is always allowed to run the at command