

# LPIC-1 102-500 – Lesson 7

**107.2 Automate system  
administration tasks by scheduling  
jobs**



# The *cron* system

- The **cron** system is used for the periodic scheduling of commands based on the minute, hour, day of the month, month and day of the week.
- It uses the **crond** daemon (simply **cron** in debian) and different configuration files



# *cron* configuration files

- **/etc/crontab** # this is the basic configuration file which include commands for the periodic execution of commands on an hourly, daily, weekly and montly basis. We usually avoid setting schedules tasks in this file and we prefer the rest of the configuration methods
- **/etc/cron.d/** # in this directory we can create files with the same format as **/etc/crontab**. It is recommended to have a different file for each task.
- **/etc/cron.hourly/**, **/etc/cron.daily**, **/etc/cron.weekly**, **/etc/cron.monthly** # in these directories we have scripts to be executed on an hourly, daily, weekly and monthly basis. The exact times for the execution are defined in **/etc/crontab**.



# Format of the */etc/crontab* and */etc/cron.d/\** files

- # minute hour dayofmonth month dayofweek user command

45 7 3 \* \* root myscript

**minute:** minute of the hour. Accepts values from 0 – 59

**hour:** hour of the day. Values range from 0 – 23

**dayofmonth:** day of the month. Accepts values from 1 – 31

**month:** month. Accepts values from 1-12 or jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec

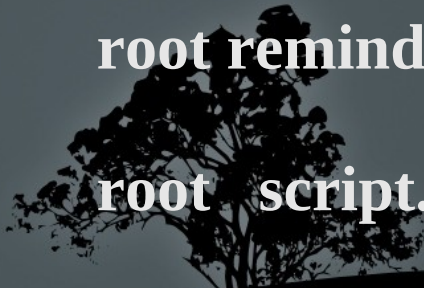
**dayofweek:** day of the week. Accepts values from 0 – 7 or sun - sat. 0 or 7 represents Sunday (sun), 1 is for Monday (mon), 2 for Tuesday (tue), 3 for Wednesday (wed), 4 for Thursday (thu), 5 for Friday (fri) and 6 for Saturday (sat).

- In the example above the **myscript** script will be executed every 3rd of each month at 7:45 am by the root user.



# More examples

```
# month      hour dayofmonth  month dayofweek user  command
# execute at every quarter of 6:00 and 18:00 every 5 days.
0,15,30,45  6,18 5,10,15,20,25,30 *      *      root  myscript
# identical time setting to the above but more readable
*/15        6,18 5-30/5 *      *      apache  apscript
# run backup every Friday
0           4     *      *      5      root  backup.sh
# Caution! This does not mean every last Friday of the month but every
# day between 24-31 and every Friday of the month!
0           4     24-31 *      5      root  backup.sh
# run every Friday of August at 6:30 in the morning
30         6     *      aug   fri   root  reminder.sh
# identical time setting to the above
30         6     *      8     5     root  reminder.sh
# run the command every minute!
*          *     *      *     *     root  script.sh
```



# Scheduling commands for all users with `crontab`

- The **crontab** command should not be confused with the `/etc/crontab` configuration file.
- It can be used to schedule tasks by each user individually
- It uses the default editor of the system (nano σε Debian) otherwise the editor set by the `$EDITOR` variable or the **select-editor** command.
- The file format is the same as `/etc/crontab` with the notable difference that we do not need to define the user who executes the task.
- The resultant configuration files for each user are saved under the `/var/spool/cron` directory (`/var/spool/cron/crontabs` in Debian)



# Scheduling commands for all users with `crontab`

- `$ crontab -e` # edit personal user crontab file
- | # | min | hour | dom | mon | dow | command   |
|---|-----|------|-----|-----|-----|---|
|   | 30  | 8    |     | 3   | 11  | * reminder_wifes_birthday.sh                                |
|   | 0   | 18   |     | *   | *   | 5 echo "yooopi\!"   mail -s "It\'s Friday" user@example.com |

## Options:

- `-l` # show the scheduled tasks of the active user
- `-r` # delete the personal crontab file of the active user and all the containing tasks.
- `-u user` # only the **root** user has the right to edit, view and delete cron tasks of other users



# Controlling *cron* usage rights with `/etc/cron.allow` and `/etc/cron.deny`

- The `/etc/cron.allow` and `/etc/cron.deny` files control which users can use the **cron** service.
- If none of the files exist then all users are allowed to use **cron** using the **crontab** command.
- If there is a **cron.deny** file only, then everybody included in it are not allowed to use **cron**
- If there is a **cron.allow** file only, then only those included in it are allowed to use **cron**
- If both files exist then **cron.deny** is ignored and only those included in **cron.allow** will be allowed to use **cron**.





# The *at* system

- The **at** system comprises the **atd** daemon and the commands **at**, **atq**, **atrm** and **batch**.
- This system allows the execution of one time only scheduled commands (not periodically like **cron**)



# Scheduling commands with `at`

- The `at` command is used to schedule one time only commands
- `# echo "shutdown -r now" | at 0400 # restart at 04:00 in the morning`
- `# echo 'mail -s Logs root < /var/log/messages' | at 00:00 feb 28`  
`# email logfiles at midnight of 28th of February`
- `# at 12:00 dec 21 2012 # shutdown the system on 21/12/2012`  
`shutdown -h now "This is the end"`  
`^D # (Press Ctrl-D to terminate the text)`
- `# at 8pm + 3 days <<EOF # run backup in 3 days at 8pm`  
`backup.sh`  
`EOF`



# Scheduling commands with `at`

- `# at -f commands.txt tomorrow # run the commands from the commands.txt text file tomorrow same time.`
- `# at -f commands.txt now + 2 days # run the commands from the commands.txt text file in two days same time.`
- Options:
  - `-f file # run commands from text file`
  - `-l # list tasks. The root user will view all the tasks for all users`
  - `-d job1 job2 job3 ... # delete tasks`



# View tasks with `atq`

- The **atq** command is identical to **at -l** and can be used to view scheduled tasks
- **# atq # = at -l**. list **at** tasks. The root user will view all the tasks for all users. Regular users will only see their own tasks.



# Delete tasks with `atrm`

- The **atrm** command is identical to **at -d** and is used for deleting tasks
- **# atrm 3 5 6 # = at -d**. Delete tasks 3, 5 and 6. Regular users can only delete their own tasks while the root user can delete the tasks of every user.



# Controlling *at* usage rights with `/etc/at.allow` and `/etc/at.deny`

- The `/etc/at.allow` and `/etc/at.deny` files control which users can use the `at` service.
- If none of the files exist then no user is allowed to use the `at`, `atq` and `atrm` commands.
- If there is a `at.deny` file only, then everybody included in it are not allowed to use `at`.
- If there is a `at.allow` file only, then only those included in it are allowed to use `at`.
- If both files exist then `at.deny` is ignored and only those included in `at.allow` will be allowed to use `at`.



# Systemd timers

- On systems that use systemd init, there is an alternative way to run scheduled tasks, using systemd timers.
- For this you need to have a timer file with the same name as the service file:

`/etc/systemd/system/backup.service`

`/etc/systemd/system/backup.timer`

- If necessary, it is possible to control a differently-named unit using the **Unit=** option in the timer's **[Timer]** section.

# Example Service File

```
[Unit]
```

```
Description=Bacakup Task
```

```
[Service]
```

```
Type=oneshot
```

```
User=root
```

```
ExecStart=/usr/local/sbin/backup.sh
```

```
RemainAfterExit=no
```

```
Nice=19
```

```
[Install]
```

```
WantedBy=multi-user.target
```





# Example Timer File

```
[Unit]
```

```
Description=Backup Timer
```

```
[Timer]
```

```
# Runs every day at 3:00 am
```

```
OnCalendar=*-*-* 03:00:00
```

```
[Install]
```

```
WantedBy=timers.target
```



# Example Timer File

```
[Unit]
```

```
Description=My Task
```

```
[Timer]
```

```
# Runs weekly and on boot
```

```
OnBootSec=15min
```

```
OnUnitActiveSec=1w
```

```
[Install]
```

```
WantedBy=timers.target
```



# OnCalendar Format

- **OnCalendar** uses the following format:  
DayOfWeek Year-Month-Day Hour:Minute:Second
- **OnCalendar=Mon,Tue \*-\*-01..04 12:00:00 #** Run the first four days of the month but only if Monday or Tuesday
- **OnCalendar=Sat \*-\*-1..7 18:00:00 #** run every Saturday
- **OnCalendar=\*-\*-\* 4:00:00 #** Run daily at 4:00 am
- OnCalendar can be specified more than once in a timer file.



# The `systemd-run` command

- **systemd-run** is used to start transient services on systemd systems. One use of this is as an **at** replacement but it is much more powerful
- **Options:**
- **--on-active= #** you can specify seconds from now
- **--on-calendar= #** you can specify a time using the OnCalendar format
- **--user:** for regular users

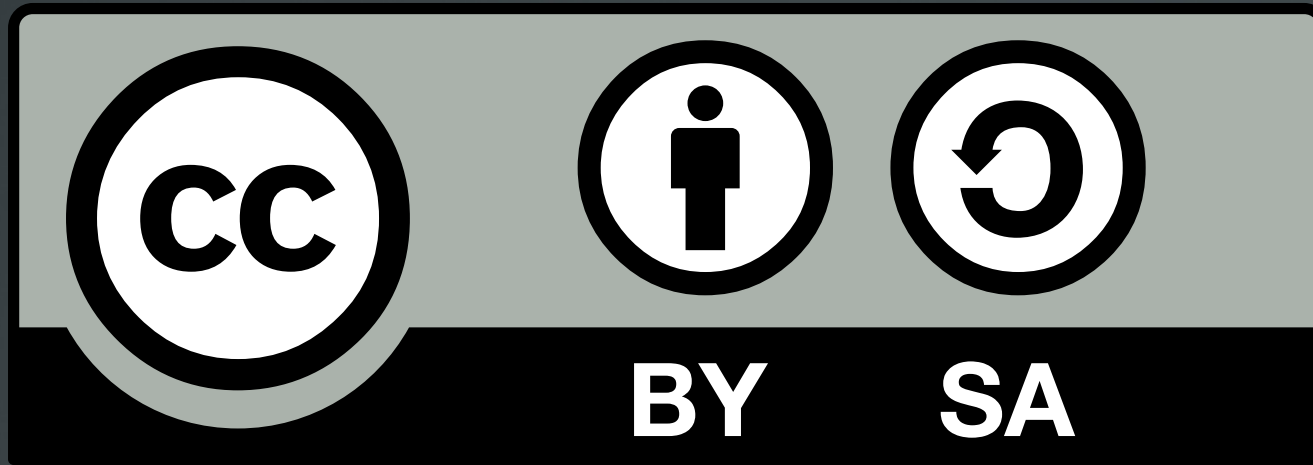


# The `systemd-run` command

- `systemd-run -on-active=300 /myscript # run task after 300 seconds from now`
- `systemd-run -on-calendar=2025-01-01 /myscript # run task on 2025 new year's day!`
- `systemd-run --on-calendar "2021-12-05 23:00" /myscript # run task at 11 pm on 2021-12-05`



# License



The work titled "LPIC-1 102-500 – Lesson 7" by Theodotos Andreou is distributed with the Creative Commons Attribution ShareAlike 4.0 International License.

