The Linux Crontab

***Easy* and available on most Unix-like operating systems**

Bonus: `systemd` services

April 2024

Jacob Cohen

University of Illinois, Chicago

jcohen30@uic.edu lug.cs.uic.edu CJacob314

Before We Begin

Firstly!

Please stop me at any time to ask for an example!

I *want* to alt-tab and show you how things actually work!

Crontab Basics[1]

- Super simple to set up ★
- Reliable (unlike some operating systems[1]) 🔨
- Allows you to consistently run tasks 🕒
  - Get status updates via email (or not) ✉

[1] Looking at you, `taskschd.msc`

Cronjob Examples

We call tasks that run on the crontab "cronjobs"
Layout: `minute hour dom month dow task [arguments]`
crontab.guru is a super helpful website for designing cronjobs!

Examples

- Run a script to check if we need to send a notification every minute.

  ```
  */1 * * * * /home/jacob/notification-checker.py
  ```

- Runs every Monday at 2:00 a.m., pulling my server backups

  ```
  0 2 * * mon /root/pull-remote-backups.sh
  ```

- Send yourself an email every time your machine reboots

  ```
  @reboot echo 'Message Text' | mail -s '
     Server Rebooted!' email_address@domain.
     tld
  ```

Editing the Crontab

How do I edit the Crontab?

Run this command, that's all

```
crontab -e
```

To change your editor, you can run

```
select-editor
```

Or you can just change your `$EDITOR` environment variable.

Configuring Emails

> If you already have emailing setup on your Linux server (or home computer!), you can simply write, somewhere in your crontab
>
> `MAILTO=email_address@domain.tld`

Systemd

What is `systemd`?

`systemd` is an array of software tools for many different Linux operating systems.
Its main goal is to serve as a “system and service manager” across Linux distributions.
The crontab actually runs on top of systemd on many systems!

Systemd Services

According to a helpful `man systemd.service` command, a systemd service is a “process controlled and supervised by systemd.”

Below is a `systemd` service I have for a Rust Discord bot I wrote.

```
[Unit]
Description=Rust Discord Bot
After=network.target

[Service]
Type=exec
EnvironmentFile=/root/Rust-Discord-Bot-Env
ExecStart=/sbin/rust-discord-bot

[Install]
WantedBy=multi-user.target
```

Working with Systemd Services

To configure a service to start on boot, we must first add it (as a `.service` file) to the systemd system directory (usually `/etc/systemd/system/`).

Then, we use the `systemctl` command to configure the service (sudo here only because our service is a *system* service).

```
sudo systemctl enable service-name.service
    # Makes the service start on boot

sudo systemctl start service-name.service
   # Starts the service now (so no reboot)

sudo systemctl status service-name.service
    # Checks the current status of the
   service

journalctl -u service-name.service # Shows
    service logs
```

Systemd User Services?

You can also have systemd services run on your *user accounts* (no `root` needed)!

This is just as simple as the system services, with three differences to keep in mind:

1. You place the service file in `~/.config/systemd/user/` instead of `/etc/systemd/system/`
2. You add a `--user` flag to all `systemctl` commands
3. Your service can't start on boot anymore, now, since it must start within the context of *your* user account.