Booting and system management daemons - Lab

Table of contents

1. Get started

1.1. Assignment

1. Set up the development environment
2. Try out the examples provided in the previous lectures
   1. Python: Basic stuff - Pt. 1
   2. Python: Basic stuff - Pt. 2
   3. Python: Advanced stuff
3. Complete the previous assignments
   1. Python: Basic stuff - Lab
   2. Python: Advanced stuff - Lab
4. Review system and service management in Linux

1.2. Hints

1.2.1. Run a Python program

```
$ python <your_program>.py
```

Depending on your system, you may have to use `python` or `python3`

1.2.2. Enable the interactive mode

```
$ python
```

Depending on your system, you may have to use `python` or `python3`

2. CPU logger

2.1. Problem

Implement a program in Python called `cpu_logger` that logs the CPU usage of the system over time. Use the function `cpu_percent` of the `psutil` module to get the current CPU usage of the system. `cpu_logger` accept a single argument, `interval`, which indicates the time interval between logs. For example, in the following case

```
$ python cpu_logger.py --interval 5
```

`cpu_logger` logs the CPU usage of the system every 5 seconds. Run `cpu_logger` as a user Linux service with `systemd`.

2.1.1. Assignment 1

Create a service unit file called `cpu-logger.service`. Check that `systemd` has installed it with the following command

```
$ systemctl --user list-unit-files --type=service | grep cpu-logger
cpu-logger.service                          static    -
```

2.1.2. Assignment 2

Connect `cpu_logger.py` with `cpu-logger.service` and start the service through `systemd`. Check that `systemd` has started it with the following command

```
$ systemctl --user status cpu-logger.service
● cpu-logger.service - CPU logger service
     Loaded: loaded (/home/ubuntu/.config/systemd/user/cpu-logger.service;
static)
     Active: active (running) since Mon 2025-03-10 22:02:34 UTC; 10s ago
   Main PID: 35593 (python)
      Tasks: 1 (limit: 2318)
```

```
     Memory: 6.6M (peak: 6.9M)
        CPU: 106ms
     CGroup: /user.slice/user-1000.slice/user@1000.service/app.slice/cpu-
logger.service
             └─35593 /home/ubuntu/cpu-logger/.venv/bin/python app.py

Mar 10 22:02:35 admin python[35593]: 1741644155.6969433 - 1.0

[...]
```

2.1.3. Assignment 3

Configure `cpu-logger.service` to be automatically restarted on failure. Run the following command to kill it and then check `systemd` has restarted it

```
$ systemctl --user kill --signal=SIGKILL cpu-logger.service
$ journalctl --user -u cpu-logger --no-pager | tail -n 10
Mar 10 22:06:23 admin python[35601]: 1741644383.0334637 - 0.0
Mar 10 22:06:24 admin python[35601]: 1741644384.033956 - 1.0
Mar 10 22:06:24 admin systemd[35373]: cpu-logger.service: Sent signal
SIGKILL to main process 35601 (python) on client request.
Mar 10 22:06:24 admin systemd[35373]: cpu-logger.service: Main process
exited, code=killed, status=9/KILL
Mar 10 22:06:24 admin systemd[35373]: cpu-logger.service: Failed with
result 'signal'.
Mar 10 22:06:24 admin systemd[35373]: cpu-logger.service: Scheduled
restart job, restart counter is at 2.
Mar 10 22:06:24 admin systemd[35373]: Started cpu-logger.service - CPU
logger service.
Mar 10 22:06:25 admin python[35616]: 1741644385.0087981 - 0.0
Mar 10 22:06:26 admin python[35616]: 1741644386.0097404 - 0.0
Mar 10 22:06:27 admin python[35616]: 1741644387.0101533 - 1.0
```

2.1.4. Assignment 4

Configure `cpu-logger.service` to be automatically started on boot in `default.target` mode. Reboot the system and then check that `systemd` has restarted it

```
$ systemctl --user status cpu-logger.service
● cpu-logger.service - CPU logger service
     Loaded: loaded (/home/ubuntu/.config/systemd/user/cpu-logger.service;
enabled; preset: enabled)
     Active: active (running) since Mon 2025-03-10 22:15:58 UTC; 11s ago
   Main PID: 817 (python)
      Tasks: 1 (limit: 2318)
     Memory: 7.2M (peak: 7.4M)
        CPU: 71ms
```

```
     CGroup: /user.slice/user-1000.slice/user@1000.service/app.slice/cpu-
logger.service
             └─817 /home/ubuntu/cpu-logger/.venv/bin/python app.py --
interval 5

Mar 10 22:15:58 admin python[817]: 1741644958.2480302 - 100.0
Mar 10 22:15:58 admin systemd[808]: Started cpu-logger.service - CPU
logger service.
Mar 10 22:16:03 admin python[817]: 1741644963.2484303 - 3.6
Mar 10 22:16:08 admin python[817]: 1741644968.6842127 - 0.6
```

2.2. Hints

2.2.1. Package management

`pip` is the standard package-management system in Python. To install a package

```
$ pip install <package-name>
```

A common practice is to list project dependencies in a file called `requirements.txt`, which is located in the project directory. For example

```
$ cd <path-to-project>
$ echo "psutil" > requirements.txt
```

To install all the project dependencies

```
$ pip install -r requirements.txt
```

A good practice is to manage dependencies separately from different projects. This is where virtual environments come into play. A virtual environment is an isolated Python development environment. `venv` is the module of the Python standard library to create virtual environments. To create a virtual environment

```
$ python -m venv .venv
```

The convention is to name the virtual environment directory `.venv` or `venv`.

To activate a virtual environment

```
$ source .venv/bin/activate
```

Then, `pip` will automatically install packages to the virtual environment.

To deactivate a virtual environment

```
$ deactivate
```

Fundamentally, a virtual environment is just a directory. To delete all the installed packages

```
$ rm -r .venv
```

2.2.2. Command-line argument parsing

`argparse` is the recommended command-line parsing module in the Python standard library. See this tutorial for a gentle introduction to `argparse`.

2.2.3. Unit files and where to find them

As mentioned here, there are several directories where `systemd` reads unit files. The recommended directory for user units created by the user is `~/.config/systemd/user`. For example, the path of the unit file for the service `cpu-logger` should be `~/.config/systemd/user/cpu-logger.service`. If the `~/.config/systemd/user` does not exist, just create it. To create a directory and make parent directories as needed

```
$ mkdir -p ~/.config/systemd/user
```

2.2.4. How to control a systemd user instance

Just append the `--user` option to the commands listed here to control the `systemd` user instance instead of the system-wide one. For example

```
$ systemctl --user list-units
```

Licenses

Content	License
Code	MIT License
Text	Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International