Lab 09: Inter Process Communication

The exercises marked with * are obligatory to complete before the upcoming weeks lecture.

The purpose of this lab is to provide the student with an elaborated understanding of interprocess communication, achieved through simple tasks. The lab is structured in a way, where the student is assisted through the tasks.

Remember to write useful commands in your cheat sheets.

Table of content

Exercise 01*

In this exercise the student is going to write a bash script that acts as a scheduler. If you are unfamiliar with bash scripting, remember the tutorial about it from the beginning of the course.

Your tasks:

- Make a scheduler.sh file
- Edit the rights to the file with `chmod +x scheduler.sh` to make the file executable
- Add the `#!/bin/bash` to the top of the file
- Make a while loop that sleeps every 0.1 seconds
- Write a function task_100ms that prints “task 100ms running” every time it is called from the while loop
- Add a counter that counts up every time the while loop runs, call task_1s when 10x task_100ms has run. Do the same for task_5s every 50 times.

The exercise will return the following message:

```
100ms task running
100ms task running
100ms task running
100ms task running
100ms task running
100ms task running
100ms task running
```

```
100ms task running
100ms task running
100ms task running
1s task running
100ms task running
```

Exercise 02*

In this exercise the student is going to create a Python script that communicates with the docker daemon through a socket and request a list of running containers.
This also means that the default Python Docker module cannot be used.

Daemons

A daemon is a program that runs in the background, without a GUI and controls a number of different processes related to the program.

Docker is built up with a client, which is the program we interact with through the CLI and an engine which is run and maintained by the daemon. They communicate through the socket accessible at `/var/run/docker.sock`

As an example for requesting a list of running containers, use the following curl command `$ curl --unix-socket /var/run/docker.sock http://localhost/containers/json`

Remember to have at least a single container running, as the command returns a list of running containers.

Hints:

- Import both `json` and `requests_unixsocket`
- It is possible to find out more than just about containers, read about the docker engine on the link: https://docs.docker.com/engine/api/v1.40/#operation/ContainerList

Your tasks:

- Make a Python script that executes the get request to the docker daemon
- Print out the result from the daemon

The exercise will return the following message:

```
[
  {
    "Id": "99b06215ed574347a640330d601b7e0e79a0a54bebfac528c2d21c33b1a8c067",
    "Names": [
      "/containerName"
    ],
    "Image": "ubuntu",
    "ImageID":
"sha256:cf0f3ca922e08045795f67138b394c7287fbc0f4842ee39244a1a1aaca8c5e1c",
    "Command": "sleep 100000",
    "Created": 1573570547,
    "Ports": [],
    "Labels": {},
    "State": "running",
    "Status": "Up 2 seconds",
    "HostConfig": {
      "NetworkMode": "default"
    },
    "NetworkSettings": {
      "Networks": {
        "bridge": {
          "IPAMConfig": null,
          "Links": null,
          "Aliases": null,
          "NetworkID":
"aa8a80b5eea94a4c4ed96e799f0750abfd5f41c55eddcd85849f70724a7c7815",
          "EndpointID":
"88aaf1ee61f9420b7fe98da0a4457148b40632d011eea3f990ddc03ecc648e9e",
          "Gateway": "172.17.0.1",
          "IPAddress": "172.17.0.2",
          "IPPrefixLen": 16,
          "IPv6Gateway": "",
          "GlobalIPv6Address": "",
          "GlobalIPv6PrefixLen": 0,
          "MacAddress": "02:42:ac:11:00:02",
          "DriverOpts": null
        }
      }
    },
    "Mounts": []
  }
]
```

Exercise 03*

In this exercise the student is going to write a bash script that starts 10 containers just by running the script. The containers are going to all run on the same network, which must be created, also in the script.

Your tasks:

- In the bash script create a new network by using `docker network create` command. It is possible to use lines that you would normally use in the terminal: “docker run -it ubuntu”.
- Write a bash script that starts 10 containers. Using a for loop is advisable.
- The names of the 10 containers should be `bash_container_1`, `bash_container_2` and so on.
- Run the script to start the 10 containers.

Exercise 04*

In this exercise the student is going to create a script that stops and removes all the containers created in exercise 03.

Your tasks:

- Create a new script that
    - Stops and removes the created containers
    - Remove the created network