OD - LAB 9: Inter Process Communication

Exercise 1

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.

Scheduler.sh

```
#!/bin/bash

function task_100ms() {
    echo “task 100ms running”
}

function task_1s() {
    echo “task 1s running”
}

function task_5s() {
    echo “task 5s running”
}

number=0
while true;
do
```

```
    sleep 0.1s
    task_100ms
    number=$((number + 1))

    if [ $number -eq 10 ]
    then
        task_1s
    elif [ $number -eq 50 ]
    then
        task_5s
    fi
done
```

Exercise 2

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.

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

Main.py

```
import json
import requests_unixsocket

# Sources:
# https://docs.docker.com/engine/api/v1.40/#section/Authentication
# https://github.com/msabramo/requests-unixsocket

if __name__ == '__main__':
```

```
    session = requests_unixsocket.Session()

    r = session.get('http+unix://%2Fvar%2Frun%2Fdocker.sock/containers/json')
    print(r.json())
```

Exercise 3

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.

containers.sh

```
#!/bin/bash

docker network create CONTAINER_CREATE

for i in {1..10}
do
    docker run --name=bash_container_$i --network=CONTAINER_CREATE ubuntu &
done
```

`docker run -it` can cause this error to be printed: `'the input device is not a TTY'` so just remove it as I did above.

& Means that it should be run in the background

Exercise 4

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

container_stop.sh

```
#!/bin/bash

docker network rm f CONTAINER_CREATE

for i in {1..10}
do
    docker stop bash_container_$i
    docker rm bash_container_$1
done
```

In case this doesn't work, use: `docker container prune` and hit 'yes'.

To remove the network, use: `docker network prune` and hit 'yes'.

Helpful commands:

`docker network ls`

`docker container ls`

`docker network rm f network_name` (the f is a shorthand for —f, which means 'force')