

# ASSIGNMENT REPORT 1: PROCESS AND THREAD IMPLEMENTATION

CENG2034, OPERATING SYSTEMS

Ümit Kadiroğlu  
umitkadiroglu@posta.mu.edu.tr

Tuesday 12<sup>th</sup> May, 2020

## Abstract

The aim of this project is to understand the functioning of threads and processes, to get used to python OS and requests libraries.

Github: [https://github.com/umitkadiroglu/ceng2034\\_020\\_m midterm](https://github.com/umitkadiroglu/ceng2034_020_m midterm)

## 1 Introduction

One of the purposes of this project is to learn how the operations are run. To understand the structure of threads and multithreading. This project was made using Linux Mint 19.3 Tricia version; using python OS, threading and requests libraries.

## 2 Assignments

The OS library was used to print the process id of the script, find loadavg, and determine the cpu core number. To check if the links are valid (working) or not, a function was written using the requests library and this function was called with the threads method.

### 2.1 Assignment 1 (Printing PID)

```
#!/usr/bin/python3
import os, threading, requests

#question1
pid = os.getpid()

print("PID of this process: ", pid)
```

Figure 1: Printing PID with os.getpid()

Output is:

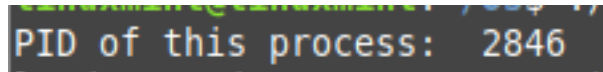
A terminal window with a dark background showing the text "PID of this process: 2846" in a light-colored monospace font.

Figure 2: The output of assignment 1

## 2.2 Assignment 2 (Printing loadavg)

```
#question2
current_os = os.name

if (current_os == "posix"):
    print("loadavg: ", os.getloadavg())
```

Figure 3: Printing loadavg with os.getloadavg()

Output is:

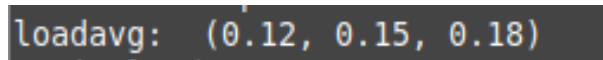
A terminal window with a dark background showing the text "loadavg: (0.12, 0.15, 0.18)" in a light-colored monospace font.

Figure 4: The output of assignment 2

## 2.3 Assignment 3 (Printing CPU core count)

Output is:

## 2.4 Assignment 4 (Validation check of url's)

Output is:

## 3 Conclusion

It was noticed that when the function was called one by one instead of using thread, it was spent 2 times more time.

```
#question3
loadavg = os.getloadavg()
cpu_count = os.cpu_count()

print("5 min loadavg: ", loadavg[1])
print("CPU core count: ", cpu_count)

if (cpu_count - loadavg[1] < 1):
    exit()
```

Figure 5: Printing CPU core count()

```
loadavg: (0.12, 0.15, 0.15)
5 min loadavg: 0.15
CPU core count: 2
```

Figure 6: The output of assignment 3

```
#question4
def requester(url):
    response = requests.get(url)
    if(response.status_code == 200):
        print(url, " is VALID")
    else:
        print(url, " is INVALID")

thread1 = threading.Thread(target=requester, args=('https://api.github.com',))
thread2 = threading.Thread(target=requester, args=('http://bilgisayar.mu.edu.tr/',))
thread3 = threading.Thread(target=requester, args=('https://www.python.org',))
thread4 = threading.Thread(target=requester, args=('http://akrepnalan.com/ceng2034',))
thread5 = threading.Thread(target=requester, args=('https://github.com/caesarsalad/wow',))

thread1.start()
thread2.start()
thread3.start()
thread4.start()
thread5.start()
```

Figure 7: Printing CPU core count()

```
http://akrepnalan.com/ceng2034 is INVALID
https://api.github.com is VALID
https://github.com/caesarsalad/wow is INVALID
https://www.python.org is VALID
http://bilgisayar.mu.edu.tr/ is VALID
```

Figure 8: The output of assignment 4