

ASSIGNMENT REPORT 1: PROCESS AND THREAD IMPLEMENTATION

CENG2034, OPERATING SYSTEMS

Mehmet Pekcan
mehmetpekcan@posta.mu.edu.tr
github: mehmetpekcan

Monday 11th May, 2020

Abstract

At the last decades, the development of computer hardware and the resource consumption of applications were very fast. However, with the slowdown in the development of computer hardware in recent years, the resource consumption of the applications used is still increasing rapidly, and the gap between hardware and resource usage is opening wide. To close this gap is with the development of the hardware part, but also based on the fact that the developed applications are based on a smart architecture.

1 Introduction

The purpose of this report is to learn how the processes work and learn the basics of developing our applications faster and with less resource consumption with a solid architecture using thread.

2 Assignments

In this project, synchronous thread operations were performed using python. At the same time, the process id of the application and how much load it was given to the user were shown.

2.1 What is used in the project

2.1.1 Which kernel and system version used in the project

System Version: macOS 10.15.3 (19D76) Kernel Version: Darwin 19.3.0

2.1.2 Which language and version used in the project

Python and version 3.8.0 were used.

2.1.3 Which libraries used in the project

os, for using operating systems command which is basically codes that uses in terminal

sys, it is big library for system command but we will only use exit and clear command from them

time, this for fun, i just want to give a mood like real world project so that i will use sleep command for a second


threading, this is for threading operations

requests, this is for requesting site and getting response from them

2.2 Problems and How to solve them in the project

2.2.1 How to reach Process id of process?

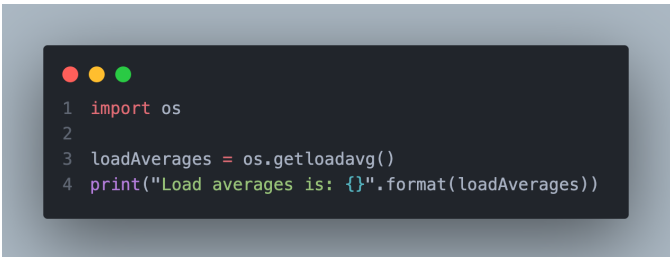
For this, i used **os** library. First imported os library then, **getpid** method. It gives us process id of the running script/program.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains four lines of Python code:

```
1 import os
2
3 processId = os.getpid()
4 print("Process Id is: " + str(processId))
```

2.2.2 How to reach Load averages of process?

For this, i used **os** library. First imported os library then, **getloadavg** method. It returns a tuple that contain 3 integers, first one of is that load average over the last 1 minute, second one is 5 minutes the last one is 15 minutes of the load average of the running program.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains four lines of Python code:

```
1 import os
2
3 loadAverages = os.getloadavg()
4 print("Load averages is: {}".format(loadAverages))
```

2.2.3 How to reach exact minutes a load average of process

For this, first you implement the above codes then you just need to take one of them. It is easy to as seen below.

```

1 import os
2
3 loadAverages = os.getloadavg()
4
5 loadAverageofOneMin = loadAverages[0]
6 loadAverageofFiveMins = loadAverages[1]
7 loadAverageofFifteenMins = loadAverages[2]
8
9 print("""Load average of one min is: {}\n
10       Load average of five min is: {}\n
11       Load average of fifteen min is: {}\n""")
12 .format(loadAverageofOneMin, loadAverageofFiveMins, loadAverageofFifteenMins))
13

```

2.2.4 Checking links are valid or not

For this, i used **requests** and **threading** libraries. Requests library is for sending request to http servers. Then getting response from them. And threading library is for creating a thread for our next process and using them. For implement them, first i created a thread using **Thread** function is takes a constructor which is function and arguments. I give them and started it, after started that process thread using **start** function in threading lib. Then i use **join** function which is again in threading lib. As you seen above the code is like:

```

1 urlList = ["https://api.github.com", "http://bilgisayar.mu.edu.tr", "https://www.python.org/",
2           "http://akrepnalan.com/ceng2034", "https://github.com/caesarsalad/wow"]
3
4 for url in urlList:
5     x = threading.Thread(target=requestByThread, args=(url,))
6     x.start()
7     x.join()

```

After then, lets look at the **requestByThread** function, as i said before i used **requests** library for it. We can send request using **get** after that request i used **status-code** function. Lets see how looks after i implemented it.

```

1 def requestByThread(url):
2     print("Url {}: requesting".format(url))
3     r = requests.get(url)
4     r.status_code = str(r.status_code)
5
6     if(r.status_code[0] == "2"):
7         print("Url {}: is valid and status code: {}".format(url, r.status_code))
8
9     elif(r.status_code[0] == "5" or r.status_code[0] == "4"):
10        print("Url {}: is not valid and status code: {}".format(url, r.status_code))
11
12 urlList = ["https://api.github.com", "http://bilgisayar.mu.edu.tr", "https://www.python.org/",
13           "http://akrepnalan.com/ceng2034", "https://github.com/caesarsalad/wow"]
14
15 for url in urlList:
16     x = threading.Thread(target=requestByThread, args=(url,))
17     x.start()
18     x.join()

```

2.2.5 Controlling of CPU usage of our program

For this, i substracted cpu count using **cpu count** method from **getloadavg** if it is less than 1 that means our program is overloads to our system.

```
1 if (os.cpu_count() - os.getloadavg()[1] < 1):
2     print("This is too much. Sorry\nExiting...")
3     time.sleep(1)
4     sys.exit()
```

3 Results

In the previous section, we did see how to implement our tasks to code. Lets fresh out. First i did implement process id using **getpid** and the result looks like that

```
Process id of running app: 4436
```

Then i determined the load average of process using **getloadavg** function and the result looks like that

```
Load average over 1 minute is: 2.67

Load average over 5 minute is: 2.36

Load average over 15 minute is: 2.51
```

After that i implemented the most important part which is sending request using threads. After whole these implementations, lets compare sending request using threads and not using threads. Lets try to understand what is the benefits of using threads in our scripts.

```
Url https://api.github.com: requesting
Url https://api.github.com: is valid and status code: 200

Url http://bilgisayar.mu.edu.tr: requesting
Url http://bilgisayar.mu.edu.tr: is valid and status code: 200

Url https://www.python.org/: requesting
Url https://www.python.org/: is valid and status code: 200

Url http://akrepnalan.com/ceng2034: requesting
Url http://akrepnalan.com/ceng2034: is not valid and status code: 404

Url https://github.com/caesarsalad/wow: requesting
Url https://github.com/caesarsalad/wow: is not valid and status code: 404

Main: All requested done successfully and passing time: 3.90 seconds
```

Above code is implementing not using threads and as we see it took to run 3.90seconds.

```
Url https://api.github.com: requesting
Url https://api.github.com: is valid and status code: 200

Url http://bilgisayar.mu.edu.tr: requesting
Url http://bilgisayar.mu.edu.tr: is valid and status code: 200

Url https://www.python.org/: requesting
Url https://www.python.org/: is valid and status code: 200

Url http://akrepnalan.com/ceng2034: requesting
Url http://akrepnalan.com/ceng2034: is not valid and status code: 404

Url https://github.com/caesarsalad/wow: requesting
Url https://github.com/caesarsalad/wow: is not valid and status code: 404

Main: All requested done successfully and passing time: 1.29 seconds
```

Now, above code is implementing using threads and as we see it took to run 1.29seconds. There is obviously a run time difference. And the second one is the efficient way to implement these program.

4 Conclusion

As a result, we saw all scripts/programs has own process id that is unique for that script in every running time. And we saw that process id specifically Unix based operating systems. On the other hand, every process has a cost on our operating system. And basically we saw the cost only over 1, 5, 15 minutes. And the most important part was how we implement our program architecture because it is related with the payload of program towards our operating system. And we saw that if we implement our program based on good threading base, we saw that it runs taking less time. In the big puzzle, we should know that we can serve better program to user without pushing them to upgrade their hardware, it is just a refactoring thing.