# ASSIGNMENT REPORT 2: CHILD PROCESS AND MULTIPROCESSING

CENG2034, OPERATING SYSTEMS

Mehmet Cihan Sakman
mehmetcihansakman@posta.mu.edu.tr
github: **https://github.com/ctyp55**

Sunday 7th June, 2020

**Abstract**

The purpose of this project is that learn how to create a child process and using multiprocessing in Python language. During this project, I used Python's **os**, **uuid**, **requests**, **hashlib** and **multiprocessing** libraries to achieve my goal. At the begging of the project, I created a child process with the **os** library. After that, I experienced that we have a problem with waiting child process and I solved this problem during this project. In the multiprocessing part, I try to do what I want with using more than one CPU in my computer.

## 1 Introduction

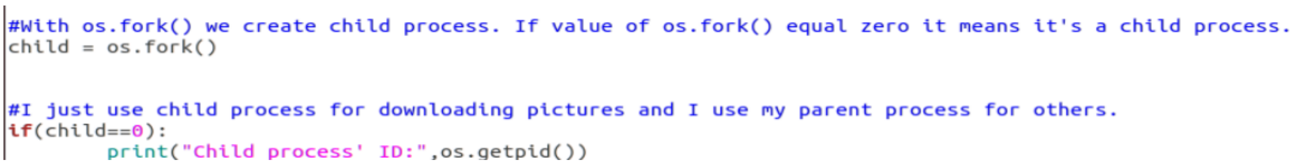
The main purpose of this project is that understand how child processes are creating, how can we use our child process in harmony with parent process and what are the benefits of using multiprocessing.

## 2 Assignments

Following subsections explains how different tasks solved step by step.

### 2.1 Child Process and it's PID

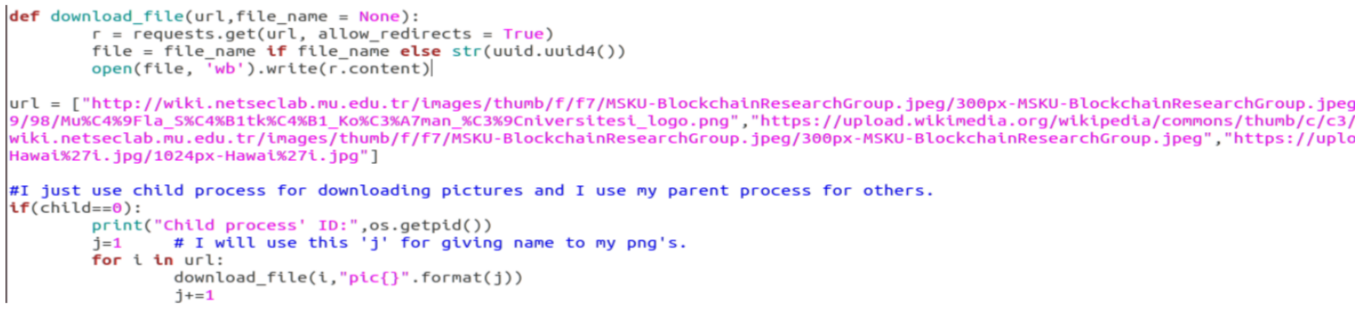
Using **os.fork()** command we can create two processes parent process and child process. **os.fork()** command returning a value to us. If this value equal to zero it means that it's a child process, if it is bigger than zero it means that it's a parent process. And with **os.getpid()** command we get the process ID of both parent and child process to see whether they are using the same memory area or not.



```
#With os.fork() we create child process. If value of os.fork() equal zero it means it's a child process.
child = os.fork()


#I just use child process for downloading pictures and I use my parent process for others.
if(child==0):
        print("Child process' ID:",os.getpid())
```

## 2.2 Downloading the files with child process

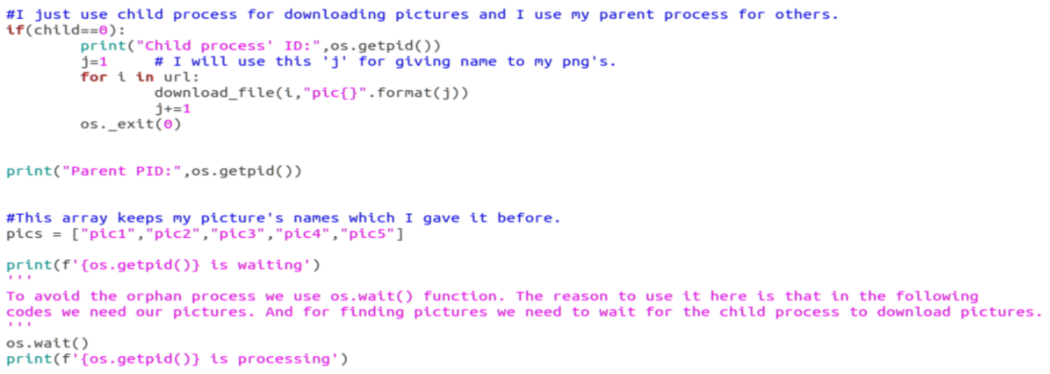
Within the child process, we download our files via given URLs. While this process we use a function which is given to us named *download-file*. This download-file function takes two parameters: the first one for the file's URL and the second one for the name of the file after download.



```
def download_file(url,file_name = None):
        r = requests.get(url, allow_redirects = True)
        file = file_name if file_name else str(uuid.uuid4())
        open(file, 'wb').write(r.content)

url = ["http://wiki.netseclab.mu.edu.tr/images/thumb/f/f7/MSKU-BlockchainResearchGroup.jpeg/300px-MSKU-BlockchainResearchGroup.jpeg
9/98/Mu%C4%9Fla_S%C4%B1tk%C4%B1_Ko%C3%A7man_%C3%9Cniversitesi_logo.png","https://upload.wikimedia.org/wikipedia/commons/thumb/c/c3/
wiki.netseclab.mu.edu.tr/images/thumb/f/f7/MSKU-BlockchainResearchGroup.jpeg/300px-MSKU-BlockchainResearchGroup.jpeg","https://uplo
Hawai%27i.jpg/1024px-Hawai%27i.jpg"]

#I just use child process for downloading pictures and I use my parent process for others.
if(child==0):
        print("Child process' ID:",os.getpid())
        j=1     # I will use this 'j' for giving name to my png's.
        for i in url:
                download_file(i,"pic{}".format(j))
                j+=1
```

## 2.3 Handle orphan process situation

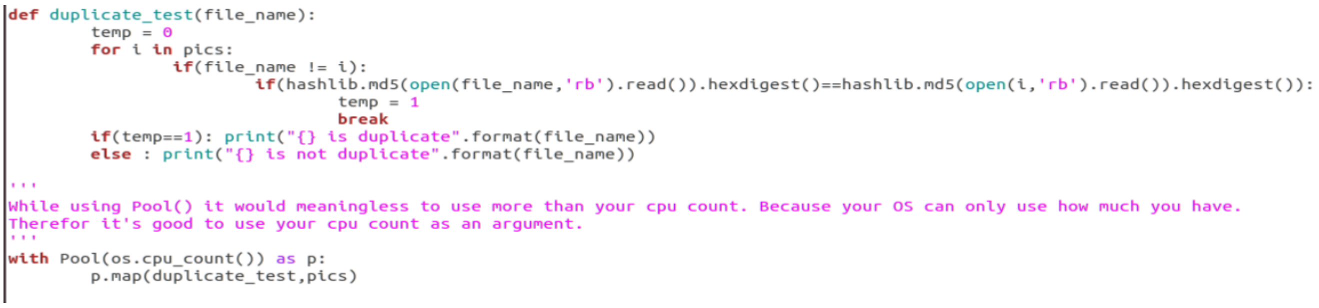
The child process and parent process are working **simultaneously**. Therefore there is a problem which we faced at this stage called **'orphan process'**. It means if our parent process end before child process after that stage child process become an orphan process and it continues to work out of our control. To avoid this situation we use **os.wait()** command to waiting child process to finish it works. Also, we use **os.-exit(0)** to ensure that the child process ends successfully.



```
#I just use child process for downloading pictures and I use my parent process for others.
if(child==0):
        print("Child process' ID:",os.getpid())
        j=1     # I will use this 'j' for giving name to my png's.
        for i in url:
                download_file(i,"pic{}".format(j))
                j+=1
        os._exit(0)


print("Parent PID:",os.getpid())


#This array keeps my picture's names which I gave it before.
pics = ["pic1","pic2","pic3","pic4","pic5"]

print(f'{os.getpid()} is waiting')
'''
To avoid the orphan process we use os.wait() function. The reason to use it here is that in the following
codes we need our pictures. And for finding pictures we need to wait for the child process to download pictures.
'''
os.wait()
print(f'{os.getpid()} is processing')
```

## 2.4 Multiprocessing

At this stage of the project, we have five different png files to check whether they are duplicate or not. To check that, I created a function named **duplicate-test** and we use multiprocessing to handle these png files I send them to **duplicate-test** as a different process. First of all we use **Pool** from **multiprocessing** library. As you can see below it takes an argument which determines how many CPUs will you use. At this point I used**os.cpu-count()** to use my all CPU. After that using **map()** function from **Pool**, we just give our function and parameter as an array(*map() function will take members of this array one by one*).

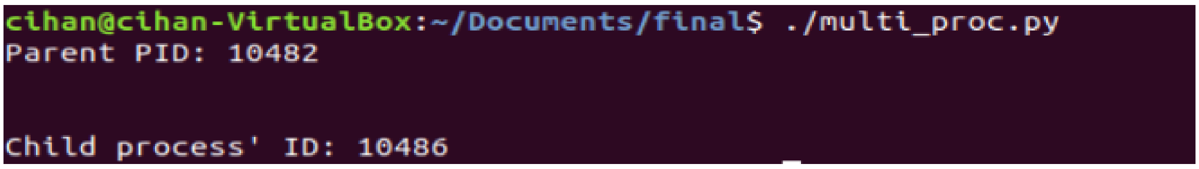


```python
def duplicate_test(file_name):
        temp = 0
        for i in pics:
                if(file_name != i):
                        if(hashlib.md5(open(file_name,'rb').read()).hexdigest()==hashlib.md5(open(i,'rb').read()).hexdigest()):
                                temp = 1
                                break
        if(temp==1): print("{} is duplicate".format(file_name))
        else : print("{} is not duplicate".format(file_name))
'''
While using Pool() it would meaningless to use more than your cpu count. Because your OS can only use how much you have.
Therefor it's good to use your cpu count as an argument.
'''
with Pool(os.cpu_count()) as p:
        p.map(duplicate_test,pics)
```

## 3 Results

We will handle results step by step.

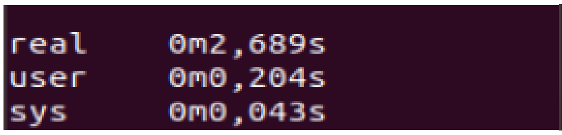
### 3.1 Child and Parent Process' ID



```
cihan@cihan-VirtualBox:~/Documents/final$ ./multi_proc.py
Parent PID: 10482


Child process' ID: 10486
```

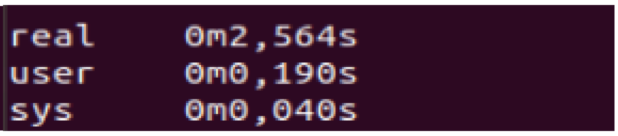
As a result of **Assignment 2.1** we can see that Parent PID and Child PID are different. Therefore we can assume that they are using different memory areas.

### 3.2 Multiprocessing



```
real    0m2,689s
user    0m0,204s
sys     0m0,043s
```

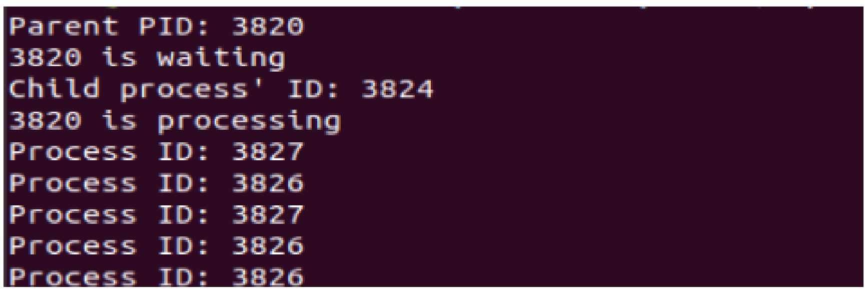
*Figure 1:* Normal Time



```
real    0m2,564s
user    0m0,190s
sys     0m0,040s
```

*Figure 2:* Multiprocessing Time

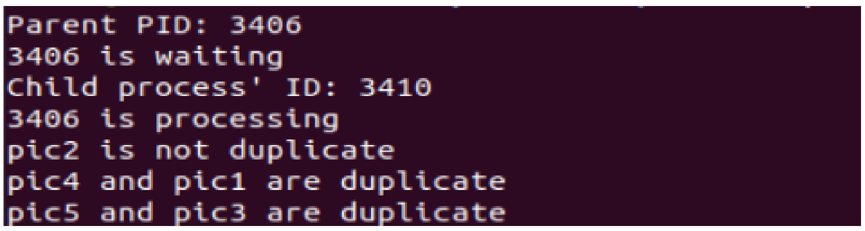
When we run our code with Multiprocessing it takes 2.56 seconds and it takes 2.69 seconds without multiprocessing. As you can see there is no big difference between Multiprocessing and Normal processing for this project.



```
Parent PID: 3820
3820 is waiting
Child process' ID: 3824
3820 is processing
Process ID: 3827
Process ID: 3826
Process ID: 3827
Process ID: 3826
Process ID: 3826
```

As we can see above, when we use multiprocessing each CPU act like different processes. Due to this system has 2 CPU there is two different processes ID when the multiprocessing called.

### 3.3 Duplication



```
Parent PID: 3406
3406 is waiting
Child process' ID: 3410
3406 is processing
pic2 is not duplicate
pic4 and pic1 are duplicate
pic5 and pic3 are duplicate
```

As a final result of this project, we can see that there is only one non-duplicate photo among five URLs.

## 4 Conclusion

As a result, the purpose of having this project is to show that whether more than time-saving for parents and child it showed a use that after child and parent processes use the same memory area since the child process is a separate process while downloading, it performs this operation in its own memory space. Normally we can use child process and parent process as parallel processing. But in this project we can not use the parent process at the same time to identify whether our URLs' content duplicate or not. Because in **Assignment 2.3** there was a problem as an orphan process. Therefore we need to wait for our child process to complete its process. And we solved this problem with **os.wait()** function. Due to identify duplicate URLs' we need the have these pictures as downloaded. And the other problem is that if parent process finish it work's before child process our program ends and child process continues to run out of our control. We avoided this problem as we discuss in **Assignment 2.3**. We are just waiting for the child process to finish its work.

The other point in this project aims to show us that using multiprocessing for saving time. Because of our finding duplicate URLs' process is not so complicated, unfortunately, we couldn't see this time-saving clearly in this project. But in normal, multiprocessing much faster than without multiprocessing. In multiprocessing due to the work as different processes, each CPU uses its own memory area therefore there may be some problems with using arrays operation.

For example, if you want to append or delete something from your array and if you want to do these operations with multiprocessing you may face some problems. Because each process has their own arrays. They can not delete or append at the same time. We should be careful about such kind of problems.