ASSIGNMENT REPORT 1: PROCESS AND THREAD IMPLEMENTATION

CENG2034, OPERATING SYSTEMS

Cansel Hatice Kucukyilmaz
canselhaticekucukyilmaz@posta.mu.edu.tr
`https://github.com/crossing-over/ceng_2034_2020_final`

Monday 8th June, 2020

Abstract

In this project we have seen how to create child process and how to use the child process using getpid () command. I learned how to avoid orphan process using the syscall feature. Next, tried to find out if the downloaded images were duplicated using the hashlib library. Using multiprocessing, we have seen how we can use whole kernels.

1 Introduction

The purpose of this project is that understand how to create child process and multiprocessing execute.

2 Assignments

2.1
Using the fork function created new child process and with sys call function print its PID and the parent PID.

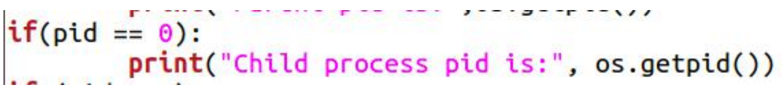
2.2
Firstly, I created a function which is name download`file. In this function we using the request module to a`
function, downloaded the `urls`in the `urls`array.

2.3
Sometimes the parent process may die before the child process ends. If the parent process dies before the child process, then the child process is inherited by the orphan process. In this case, the child process can get out of control. To avoid this situation, it is provided to wait for the child process of the parent process by using the os.wait () function.

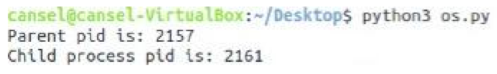
2. 4
Using python hashlib library we check the images which is one step earlier downloaded. if there is any duplicate files using the the downloadFile function we try to remove the duplicate files, in this step we use multiprocessing, A process pool object was created to control the pool of worker processes to which transactions can be sent, and a parallel map application was made.Thus, the cores were used more efficiently.

3 Results

1.1 Printing child process id: Using the fork() is an operation whereby a process creates a copy of itself. Ant with getpid() returns the process ID (PID) of the calling process.

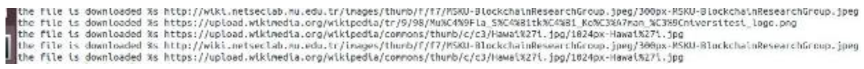


```
if(pid == 0):
        print("Child process pid is:", os.getpid())
```

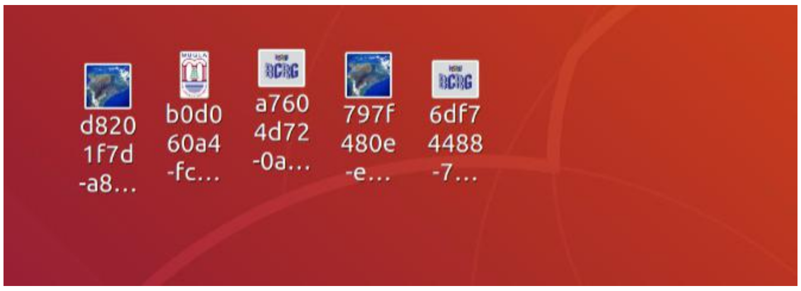


```
cansel@cansel-VirtualBox:~/Desktop$ python3 os.py
Parent pid is: 2157
Child process pid is: 2161
```

2.2 Firstly, I created a function which is name downloadFile. In this function we using the request module to allows us to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data. After that, we want to assign a name the file different from each other. So, we use the UUID(Universal Unique Identifier) module which is a python library which helps to generate the random id. Than, we check if the pid is equal to zero this means its a child process.Using child process and downloadFile function, downloaded the urls in the urls array.

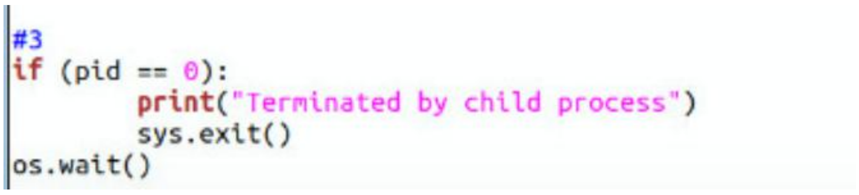


```
the file is downloaded %s http://wiki.netseclab.mu.edu.tr/images/thumb/f/f7/MSKU-BlockchainResearchGroup.jpeg/300px-MSKU-BlockchainResearchGroup.jpeg
the file is downloaded %s https://upload.wikimedia.org/wikipedia/tr/9/98/Mu%C4%9Fla_S%C4%B1tk%C4%B1_Ko%C3%A7man_%C3%9Cniversitesi_logo.png
the file is downloaded %s https://upload.wikimedia.org/wikipedia/commons/thumb/c/c3/Hawai%27i.jpg/1024px-Hawai%27i.jpg
the file is downloaded %s http://wiki.netseclab.mu.edu.tr/images/thumb/f/f7/MSKU-BlockchainResearchGroup.jpeg/300px-MSKU-BlockchainResearchGroup.jpeg
the file is downloaded %s https://upload.wikimedia.org/wikipedia/commons/thumb/c/c3/Hawai%27i.jpg/1024px-Hawai%27i.jpg
```



2.3

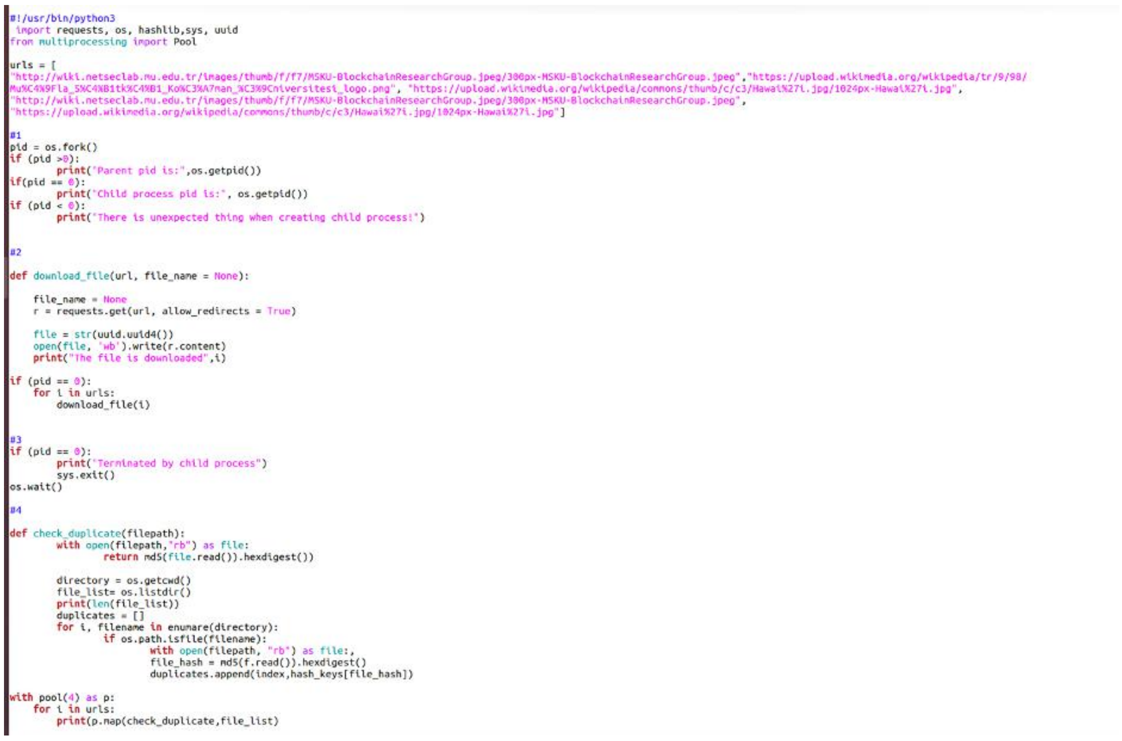
Avoiding Orphan Process In this part we check the process pid, if pid is equal to zero(child process).We wait to finish the child process.



```
#3
if (pid == 0):
        print("Terminated by child process")
        sys.exit()
os.wait()
```

4 Conclusion

Conclusion: In this project, a process pid was found using some commands using the os module in linux operating system and Python. Also, by creating a subprocess-this technique pertains to multitasking operating systems, and is sometimes called a subprocess or traditionally a subtask-parallel work was done. By using the os.wait() command, the parent process is prevented from dying before the child and the orphan process situation is avoided. While downloading images using the download`file function, id is assigned for each image using the uuid module.` Next, I tried to find out if the downloaded images were duplicated using the hash library. Also, Using multiprocessing, we have seen how we can use whole kernels more effectively.



```
#!/usr/bin/python3
 import requests, os, hashlib,sys, uuid
from multiprocessing import Pool

urls = [
"http://wiki.netseclab.mu.edu.tr/images/thumb/f/f7/MSKU-BlockchainResearchGroup.jpeg/300px-MSKU-BlockchainResearchGroup.jpeg","https://upload.wikimedia.org/wikipedia/tr/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/Hawai%27i.jpg/1024px-Hawai%27i.jpg",
"http://wiki.netseclab.mu.edu.tr/images/thumb/f/f7/MSKU-BlockchainResearchGroup.jpeg/300px-MSKU-BlockchainResearchGroup.jpeg",
"https://upload.wikimedia.org/wikipedia/commons/thumb/c/c3/Hawai%27i.jpg/1024px-Hawai%27i.jpg"]

#1
pid = os.fork()
if (pid >0):
        print("Parent pid is:",os.getpid())
if(pid == 0):
        print("Child process pid is:", os.getpid())
if (pid < 0):
        print("There is unexpected thing when creating child process!")


#2

def download_file(url, file_name = None):

    file_name = None
    r = requests.get(url, allow_redirects = True)

    file = str(uuid.uuid4())
    open(file, 'wb').write(r.content)
    print("The file is downloaded",i)

if (pid == 0):
    for i in urls:
        download_file(i)


#3
if (pid == 0):
        print("Terminated by child process")
        sys.exit()
os.wait()

#4

def check_duplicate(filepath):
        with open(filepath,"rb") as file:
                return md5(file.read()).hexdigest())

        directory = os.getcwd()
        file_list= os.listdir()
        print(len(file_list))
        duplicates = []
        for i, filename in enumare(directory):
                if os.path.isfile(filename):
                        with open(filepath, "rb") as file:,
                        file_hash = md5(f.read()).hexdigest()
                        duplicates.append(index,hash_keys[file_hash])

with pool(4) as p:
    for i in urls:
        print(p.map(check_duplicate,file_list)
```