ASSIGNMENT REPORT 2: CHILD PROCESS AND MULTIPROCESSING

CENG2034, OPERATING SYSTEMS

Gizem PESEN
gizempesen@posta.mu.edu.tr
github: **https://github.com/GPS199**

Wednesday 24th June, 2020

Abstract

Today, this is the world of technology . Technology is developing, the needs have increased. The development of processor and operating system architectures respond to the needs of the age. So we , the programmers need to understand how operating system works. we need to understand creating child process in the operating system. Understanding the multi processing is important to understand the computer.

1 Introduction

Purpose of this report is learning how to use **multiprocessing , threads** in our codes and how it will be increased our script performance.

2 General Information

2.0.1 My device:

I generally used virtual environment but I did some process in my device that use windows.

System Version: Windows 10
CPU: Intel(R) Core(TM) i5-7200U
Ram: 8 GB
64 bit processor

2.0.2 My Virtual Environment:

Vmware Workstation 15.5
Virtual Environment: Ubuntu 20.04
Memory: 2 GB
Processors: 1

2.0.3 Repl.it (Online Virtual Environment):

Kernel version: Linux 5.4.0-1009

2.0.4 Language and Ide:

Python 3.8.0
PyCharm Community Edition 2020.1.2

2.0.5 Used Libraries:

os, os.path to create child and parent process, to check path exist or not
requests: to download images
uuid:to create random image name if images don't have
time:to check elapsed times
threading: to download images with thread
hashlib: to use hashlib -md5/sha256-
multiprocessing:
psutil: to learn memory and cpu usage
subprocess : to clean orphans
sys: to animation

3 Assignments

Following subsections explains how different tasks solved step by step.

3.0.1 Create child process and using it

It is easy to created child process with **fork()**, importing **os** library, we should first forking a process from parent.

Listing 1: Creating child process then writing its process id

```
def parent_child_process():# Create Parent and Child Process
    pid = os.fork()

    if(pid > 0):# n greater than 0 means parent process
      print("parent process id: {}" .format(os.getpid()))

    elif (pid == 0):# Pid is equal to 0 means child process
      print("child process id: {}".format(os.getpid()))
```

Listing 2: Output

```
parent process id: 254
child process id: 260
```

In addition to this; In this world it is possible to encounter an error. So we can avoid errors with try and except .
Like below if we get error because of **os** to create **pid** , the program will **exit** with a message. ;

Listing 3: Output

```
try:
    pid = os.fork()
  except OSError:
    exit("A problem occur when try to create pid.")
```

3.0.2 Download files

Here, we can easily download files but we should use multiprocessing to gain time advantage. I will show you the **time difference** of **finding duplicte files** and **finding duplicate files with threads**.

Listing 4: Downloading files

```
def download_file2():
    print("{} files downloading...\n".format(len(urls)))
    for i in range(len(urls)):
        download_file(urls[i], "file{}".format(i))
        print("file{0} downloaded from\n{1}\n".format(i, urls[i]))
```

This function calls **download file()**. I created a list in **download file()** function for downloading thread.Because of it this function gave an **AttributeError**. To solve it I added try and except and it runs.

Listing 5: Try - Except

```
    try:
        download_file(urls[i], "file{}".format(i))
    except AttributeError:
        print("it is running")
```

3.0.3 Create Directory

It is created directory named Images. I used **os** library for creating it, **os.mkdir** is the method to create directory.

Listing 6: Create Directory

```
source = 'Images/'

def createDirectory(path):
    access_rights = 0o777

    try:
        os.mkdir(path, access_rights)
    except OSError:
        print('Creation of the directory %s failed' % path)
    else:
        print('Successfully created the directory %s' % path)
```

3.0.4 Change Directory

After we created directory, we can learn the current working directory of a process with **getcwd()** , Then we can change the directory with **os.chdir** method.

Listing 7: Change Directory

```
def changeDirectory(getDirectory): # Change Directory
    first_directory = os.getcwd() #current
    os.chdir(getDirectory) #change it
    print('Directory Changed as', first_directory + '/' + getDirectory)
```

And result is like below ;

Listing 8: Output

```
Successfully created the directory Images/
Directory Changed as /home/gizem/But/Images/
Images folder contains 939 files.
```

3.0.5 Check Is There a File Exist

To check there is a file or not , it is used **path.exists()** method with importing **os.path** . This function is **boolean** and it gives random file name if there is not exist a file name.

Listing 9: Checking Path

```
def check_path_exist(file_name):
        isExist = path.exists(file_name) # path exists or not
        if isExist == 0: # if there is no file name;
            print ("directory not exists:" + str(path.exists('Images')))
                file = file_name if file_name else str(uuid.uuid4()) # random file name
        else:
                print ("directory exists:" + str(path.exists('Images')))
```

Listing 10: Output1

```
directory exists:True
```

Listing 11: Output2

```
directory not exists:False
```

3.0.6 Download files with using Multi Threading

Sending http request in python can make using **request** library. it is created **a list** for file name . We could check path.exist in this method .

Listing 12: Downloading files

```
file_name_list = []

def download_file(url, file_name_list, file_name = None, ):
    r = requests.get(url, allow_redirects=True) #this method makes a request to a url
    file = file_name if file_name else str(uuid.uuid4())
    #check_file_exist(file_name)
    open(file, 'wb').write(r.content) # open the downloaded file
    file_name_list.append(file)
```

3.0.7 Using Multi Threading

First I create a threads list to append the files with downloading using threading. it is used 2 loops; one of them is in url to start download files , other is in threads list that to join them.

Listing 13: Creating Threads

```
threads = []

    def create_download_thread(urls,file_name_list):

    for url in urls: #in urls array take loop
        download_thread =
            threading.Thread(target=download_file,args=(url,file_name_list,))
        download_thread.start()
        threads.append(download_thread) #add downloaded threads to list

    for thread in threads:
        thread.join()

proc = Process(target=create_download_thread,args=(urls,file_name_list,))
proc.start()
proc.join()
```

When you are downloading files with thread that will be occur in screen:

Listing 14: Output

```
/5 files downloading...

5 files downloading...

5 files downloading...

5 files downloading...

5 files downloading...
```

3.0.8 Clear orphan process

When parent finishes execution and exits while the child process is still executing ; it is called an orphan process.
We can avoid the orphan process if we know the place that we can use os.wait() . To create a Clear Orphan is more logical if it is hard to find the line for avoiding. The code below it is used subprocess library. An you can see the function cleaned orphans in output.

Listing 15: Clear Orphans

```
def clear_orphans():
  process= subprocess.Popen( ('ls', '-l', '/tmp'), stdout=subprocess.PIPE)
  for line in process.stdout:
       pass

  subprocess.call( ('ps', '-l') )
  process.wait()
  print( "clearing...\n")
  subprocess.call( ('ps', '-l') )
```

Then it clears orphans like this:

Listing 16: Example Output of Clearing Orphans

```
F S   UID    PID   PPID  C PRI  NI ADDR SZ WCHAN TTY        TIME CMD
0 S  1000     98      1  4  80   0 - 278139 wait pts/0  00:00:00 prybar-python3
4 Z  1000    104     98  0  80   0 -     0 -     pts/0  00:00:00 ls <defunct>
4 R  1000    105     98  0  80   0 -  7426 -     pts/0  00:00:00 ps
after wait
F S   UID    PID   PPID  C PRI  NI ADDR SZ WCHAN TTY        TIME CMD
0 S  1000     98      1  4  80   0 - 278139 wait pts/0  00:00:00 prybar-python3
0 R  1000    111     98  0  80   0 -  7426 -     pts/0  00:00:00 ps
```

3.0.9 Find Duplicates

Listing 17: Find Duplicate

```
def findDuplicates():
    for index, filename in enumerate(os.listdir('.')): #count each iteration
        if os.path.isfile(filename): #used path library and checked
            index = filename
            with open(filename, 'rb') as f: #open file
                image_hash = hashlib.md5(f.read()).hexdigest() #read with haslib

            if image_hash not in hashes:
                hashes[image_hash] = index

            else:
                duplicates.append((index, hashes[image_hash]))#adding the duplicate list
                print('Image ' + hashes[image_hash] + ' is duplicate of Image ' + index)
```

3.0.10 Remove Duplicates

First, asked the question and took input from user, according to answer ; y or n remove files with **os** library. I write this algorithm but there is **TypeError**.

Listing 18: Remove Duplicates

```
def remove_dup():
  print("Do you want to delete duplicate images? [y/n]")
  while(True):
    remove = input("Please enter the command number:\n")
    if(remove == "y"):
        files_list = os.listdir()#list of the all files located

      #delete files after printing:
        for index in duplicates:
          index = 0
          os.remove(files_list[index]) ##remove duplicate file

    else:
      print("you didn't delete duplicte images")
```

Listing 19: An Error

```
TypeError: list indices must be integers or slices, not str
```

3.0.11 CPU Count

It is checked the system and learn the number of CPU cores. and use **cpu count()** method to create n processes if there are n cores with using **multiprocessing** library but it is also possible to search **cpu count()** with **os** library.

Listing 20: CPU Count

```
print("The number of cpu count is:",multiprocessing.cpu_count())
```

Listing 21: Output

```
The number of cpu count is: 4
```

Cpu Count in Multiprocessing

Using Pool() it would meaningless to use more than your cpu count. So it used os.cpu count() as an argument,because your OS can only use how much you have.

Listing 22: n process if there are n cores

```
with multiprocessing.Pool(os.cpu_count()) as Pool:
```

3.0.12 Creating Multiprocess

Before continue with our task we should know to use multiprocesses. Multiprocesses run at a same time. So let's create a multiprocess using **multiprocessing** library.

Above code create a process pool to call function. Then mapping them with a function and parameter. And the output is:

Listing 23: How to create multiprocess example

```
import multiprocessing
import time # this is for simulating

def do_staff(second):
    print(f"do_staff function starts with parameter: {second}")
    time.sleep(second)
    print("do_staff function ends")

if __name__ == "__main__":
    processPool = multiprocessing.Pool()
    processPool.map(do_staff, [1,2,3,4])
```

3.0.13 Creating Multiprocess for our task

First let's download files using **multiprocessing** library.

Listing 24: Download File using Multiprocess

```
processPool = multiprocessing.Pool()

# Assume that "...." end of the urls
request_urls = [
"http://wiki.netseclab.mu.edu.tr/images/thum.....",
"https://upload.wikimedia.org/wikipedia/tr/9....",
"https://upload.wikimedia.org/wikipedia/c....",
"http://wiki.netseclab.mu.edu.tr/images/thum....",
"https://upload.wikimedia.org/wikipedia/commons...."]

processPool.map(download_file, request_urls)
```

3.0.14 Starmap

It is used duplicate first list and duplicate Finder function to find duplicate files with multiprocessing .

Listing 25: Starmap

```
    with multiprocessing.Pool(os.cpu_count()) as Pool:

        list_duplicate =
            Pool.starmap(duplicateFinder,([duplicate_first[0],duplicate_first],[duplicate_first[1],duplicate
```

It is used **starmap** above code. Because built-in **map** function is taking just one parameter to send function as a parameter, but in our case we should send two parameter to function. So we used **starmap** to send two parameter using **repeat** arguments.

3.0.15 Hashlib

The fuction that hashed files.

Listing 26: Hash Generator Function

```
    from hashlib import md5

    def takegethash():
    for index, filename in enumerate(os.listdir('.')):
        if os.path.isfile(filename):
            index = filename
            with open(filename, 'rb') as f:
                image_hash = hashlib.md5(f.read()).hexdigest()
            duplicate_first.append((index, image_hash))
```

3.0.16 Multiprocess and Signal

You can clearly see the usage of signal below. First we should import **signal** library, In example it is used for except an error , you can use this library to interrupt something or with **time.sleep()** etc.

Listing 27: Multiprocess and Signal Example

```
import signal
...
def init_worker():
    signal.signal(signal.SIGINT, signal.SIG_IGN)
...
def main()
    pool = multiprocessing.Pool(size, init_worker)
    ...
    except KeyboardInterrupt:
        pool.terminate()
        pool.join()
```

So I used signal library , and kill the process with **os.kill()** if it takes any time that you choose instead of seconds parameter.

Listing 28: Multiprocess and Signal Example

```
def init_worker():
    signal.signal(signal.SIGINT, signal.SIG_IGN)

def wait_timeout(proc, seconds):
    start = time.time()
    end = start + seconds
    interval = min(seconds / 1000.0, .25)

    while True:
        pool = multiprocessing.Pool(os.cpu_count(), init_worker)
        if pool is not None:
          print("let live")
          return Pool
        if time.time() >= end:
          print("kill")
          os.kill(proc, signal.SIGTERM)
```

3.0.17 General Errors that I face with

IndentationError: unindent does not match any outer indentation level
solution: it is about tab key, just fixed spaces

TabError: inconsistent use of tabs and spaces in indentation
solution: Make a search and replace to replace all tabs with 4 spaces.

NameError: name 'threading' is not defined
solution:just forgot import thread

UnboundLocalError: local variable 'file names' referenced before assignment

IndentationError: expected an indented bloc

NameError: name 'subprocess' is not defined
solution: just import subprocess.

AttributeError: 'str' object has no attribute 'append'

SyntaxError: invalid syntax
AttributeError file name list.append(file)
solution: I used try-except.

IndentationError: unindent does not match any outer indentation level **solution:**there might be spaces mixed in with your tabs.

IndexError: list index out of range
UnboundLocalError: local variable 'filehash' referenced before assignment
TypeError: list indices must be integers or slices, not str

4 Memory

4.0.1 Available Memory

Listing 29: Available memory

```
def available_memory() -> int:
    # open this file for memory information
    with open('/proc/meminfo', 'r') as mem:
        ret = {}
        free = 0
        for i in mem:
            sline = i.split()
            if str(sline[0]) == 'MemTotal:':
                ret['total'] = int(sline[1])
            elif str(sline[0]) in ('MemFree:', 'Buffers:', 'Cached:'):
                free += int(sline[1])
        return free
```

4.0.2 Wait Until Memory is Available

With this function we call the available memory function back and wait until memory is available.

Listing 30: Waiting Available memory

```
def wait_until_memory_is_available():
    while True: # wait until memory becomes available
        print("memory is available")
        available_memory_mb = available_memory() / 1000
        if available_memory_mb < required_memory_mb:
            print(
                "memory is not enough! Available: {} MB | Required: {}
                    MB".format(available_memory_mb, required_memory_mb))
            time.sleep(1) # wait
        else:
            return
```

We used this data above:

Listing 31: Data that I should have

```
required_memory_mb = 30 # minimum required amount of memory in mb
hash_buffer_size = 30000 # buffer size in bytes
movie_file_path = None # the movie file path is not exist
```

4.0.3 The output without psutil library

I just access any Memory Information with a class FreeMemLinux(object) without importing **psutil** library. You can see some memory informations below.

Listing 32: Output

```
Total Memory: 26694612

Used Memory: 20619100

Free Memory: 24546296

Total Memory (MB): 26068.95703125 MB

Used Memory (MB): 20135.83984375 MB

Free Memory (MB): 23970.9921875 MB

Percentage of Total Memory: 1.0 %

Percentage of Used Memory: 0.7724068062873511 %

Percentage of Free Memory: 0.9195224864103663 %
```

4.0.4 Checking Memory and CPU with Psutil library

I realized that **psutil** is a great library for checking cpu and memory like below , without psutil we need to write a lot of functions with object orianted.

```
max_cpu_percent = 50
max_memory_percent = 50

print(psutil.cpu_times())

def cpu_usage():
  print("CPU Percent="+psutil.cpu_percent(interval=1),"\n")
  cpu_percent = psutil.cpu_percent(interval=1)
  if(cpu_percent > max_cpu_percent):
      print("Warning")
      print("CPU Usage Greter than 50%")

def memory_usage():
  print(psutil.virtual_memory())
  mem = psutil.virtual_memory()

  print("Memory Percent="+str(mem.percent))
  if(mem.percent > max_memory_percent):
      print("Memory Usage Greter than 50%")
```

.
.

Listing 33: Output

```
/svmem(total=27335290880, available=20669296640, percent=24.4, used=8078446592,
    free=12509163520, active=7963410432, inactive=3783192576, buffers=658960384,
    cached=6088720384, shared=22503424, slab=2435268608)

Memory Percent=24.4

CPU Percent=71.7
```

4.0.5 Take a movie file (1GB) in your disk and try to hash it. How can you handle when hashing files that don't fit in the memory

Listing 34: Downloading huge data

```
def movie_file():
  if movie_file_path:
        movie_hash = takegethash(movie_file_path)
        print("The hash of the movie using {} byte chunks: {}".format(hash_buffer_size,
            movie_hash))


    print("\nExiting...")
```

In addition to this ; for huge file you can use this chunk function .

Listing 35: Chunk is for huge files

```
    #If files are huge ... use this
def read_chunk(fobj, chunk_size = 2048):
    """ Files can be huge so read them in chunks of bytes. """
    while True:
        chunk = fobj.read(chunk_size)
        if not chunk:
            return
        yield chunk
```

5 Time

You can clearly see that downloading with thread gives us time advantage.

Listing 36: Output Comparing Downloading Process

```
Elapsed time with downloading : 2.233150005340576
Elapsed time with downloading with thread: 1.9434547424316406
```

Comparison of thread and multiprocessing:

Listing 37: Output of difference

```
    Elapsed time with serial threading is: 2.09sec
    Elapsed time with multiprocessing is: 1.02sec
```

As you see the result multiprocessing is running in a half time of serial threading. Think that you have thousands of data to making something, multiprocessing will be make huge different for you.

6 Screen

Listing 38: Function and driver code for duplicate image finding

```
print("\nWelcome Screen\n")
print("-----------------------------------------------------------")


#On the Screen ..
print("General Information")
print("-----------------------------------------------------------")
print("Operating System Type: ", os.name, "\n") #Windows = nt / Linux = posix
print("Your Destination: ", os.getcwd(), "\n") #current path
print("CPU (Core) Count: ", str(os.cpu_count()), "\n")
print("Load avg:", os.getloadavg(), "\n")
print("-----------------------------------------------------------")
print("Python version:")
os.system("python3 --version")
print("\n")
print("Kernel version:")
os.system("uname -srm")
print("-----------------------------------------------------------")
print("\n \n Please enter the command number to execute the script: \n"
      " 1 : Create a New Parent and Child Process and Print Process ID (PID). \n"
      " 2 : Check is there a directory or not. \n"
      " 3 : Create a New Directory, Change Directory and Find count of File in it. "
      " 4 : Change Directory \n"
      " 5 : Download the Files With Child Process.(and Avoid Orphans) \n"
      " 6 : Download Files With Threads. \n"
      " 7 : Clean Out the Orphans. \n"
      " 8 : Find Duplicates. \n"
      " 9 : Control Duplicate Files With Multi Processing. \n"
      " 10 : Check Memory Usage and CPU Usage.\n"
      " 11 : Memory Information.\n"
      " 12 : Print Elapsed Time of Every Programs . \n"
      " 13 : Is Memory Available , Wait it. \n"

      " 0 : Exit the script.\n")
print("-----------------------------------------------------------\n")
```

7 Design

7.0.1 Process Bar

Listing 39: Loading Animation

```
# Print iterations progress
def printProgressBar (iteration, total, prefix = '', suffix = '', decimals = 1, length
    = 100, fill = '', printEnd = "\r"):

    percent = ("{0:." + str(decimals) + "f}").format(100 * (iteration / float(total)))
    filledLength = int(length * iteration // total)
    bar = fill * filledLength + '-' * (length - filledLength)
    print(f'\r{prefix} |{bar}| {percent}% {suffix}', end = printEnd)
    # Print New Line on Complete
    if iteration == total:
        print()

# A List of Items
items = list(range(0, 57))
l = len(items)

# Initial call to print 0% progress
printProgressBar(0, l, prefix = 'Progress:', suffix = 'Complete', length = 50)
for i, item in enumerate(items):
    # Do stuff...
    time.sleep(0.1)
    # Update Progress Bar
    printProgressBar(i + 1, l, prefix = 'Progress:', suffix = 'Complete', length = 50)
```

7.0.2 Loading

If you didn't see this loading animation you will feel boring while waiting. I used **sys** for it . And it is just for loop and time.sleep()

Listing 40: Loading Animation

```
#Just an animation for loading
def animation():
  animation = "|/-\\"

  for i in range(10):
      time.sleep(0.1)
      sys.stdout.write("\r" + animation[i % len(animation)])
      sys.stdout.flush()
```

I prefer to create a menu with Drive codes using if-else statements with taking input.

Listing 41: Driver Codes

```
while(True):
  command = input("Please enter the command number:")
  if(command==None or command=="" or command==" "): #base case
    print("Unvalid command number was sended.")

  elif(command=="1"):
    animation()
    parent_child_process()
. . .          # Assume that this 3 point is other commands
  elif(command=="5"):
    animation()
    print("\n")
    pid = os.fork()
    if pid > 0:
        os.waitpid(pid, 0) #For avoiding orphan process
        print("")
    else:
        download_file2()
. . .
  elif(command=="0"):
    print("Exiting from the script.")
```

8 Conclusion

To sum up; operating system is a world , and you can lost in it even. I am so happy to have this project make up. I gain a lot of things like **child process** , **multiprocessing** and **threads** . In addition to this I realized the importance of **memory** and **cpu** and being a computer engineer is using your system efficiently.

9 References

- https://www.programiz.com/python-programming/time/sleep
- https://docs.python.org/2/library/multiprocessing.html
- https://www.w3schools.com/python/python try except.asp
- https://stackoverflow.com/questions/22029562/python...
- https://forum.yazbel.com/t/virgullu-noktali-sayilarin-kac-basamakli-olacagini-belirleme/724/3
- https://stackoverflow.com/questions/1131220/get-md5-hash-of-big-files-in-python
- https://stackoverflow.com/questions/3173320/text-progress-bar-in-the-console
- https://stackoverflow.com/questions/276052/how-to-get-current-cpu-and-ram-usage-in-python
- https://stackoverflow.com/questions/82831/how-do-i-check-whether-a-file-exists-without-exceptions
- https://python.readthedocs.io/en/latest/library/os.path.html
- https://stackoverflow.com/questions/51520/how-to-get-an-absolute-file-path-in-python
- https://stackoverflow.com/questions/4446366/why-am-i-getting-indentationerror-expected-an-indented-block
- https://stackoverflow.com/questions/31607873/removing-orphan-letters-in-a-list-with-python
- https://www.youtube.com/watch?v=ozmL7TZ6Nm8
- https://stackoverflow.com/questions/17718449/determine-free-ram-in-python
- https://www.tutorialspoint.com/python/osgetcwd.htm
- https://stackoverflow.com/questions/32053618/how-to-to-terminate-process-using-pythons-multiprocessing
- https://www.tutorialspoint.com/python/osgetcwd.htm
- https://stackoverflow.com/questions/40672264/python-dynamic-multiprocessing-and-signalling-issues
- https://stackoverflow.com/questions/17718449/determine-free-ram-in-python
- https://stackoverflow.com/questions/2760652/how-to-kill-or-avoid-zombie-processes-with-subprocess-module/2761781
- https://www.cloudcity.io/blog/2019/02/27/things-i-wish-they-told-me-about-multiprocessing-in-python/
- https://www.cloudcity.io/blog/2019/02/27/things-i-wish-they-told-me-about-multiprocessing-in-python/
- https://realpython.com/python-memory-management/

- https://stackoverflow.com/questions/4446366/why-am-i-getting-indentationerror-expected-an-indented-block
- https://github.com/Kirk-H/Nagios-mem check/blob/master/mem check.py
- https://discuss.codecademy.com/t/typeerror-list-indices-must-be-integersorslices-not-str-problem/72521/5
- https://pymotw.com/2/multiprocessing/communication.html
- https://stackoverflow.com/questions/276052/how-to-get-current-cpu-and-ram-usage-in-python
- https://stackoverflow.com/questions/185936/how-to-delete-the-contents-of-a-folder
- https://stackoverflow.com/questions/1408356/keyboard-interrupts-with-pythons-multiprocessing-pool