Date - 03/05/2021

Experiment No. 4
Processes and Threads

Aim:

a.To write a program to create a child process and print the parent and child id
b.To write a multithreaded program in python such that thread1 finds the square of a number and thread finds the factorial of a number.

Theory:

Process: A process is a program in execution which follows a sequential pattern of execution. To create a child process in python. The os module is used to invoke various system calls. Processes are created by the operating system to run programs. Processes can have multiple threads. Two processes can execute code simultaneously in the same python program. Processes have more overhead than threads as opening and closing processes take more time. Sharing information between processes is slower than sharing between threads as processes do not share memory space. In python they share information by pickling data structures like arrays which requires IO time.

Thread: A thread is a path of execution within a process. A process can contain multiple threads. Threading in python is used to run multiple threads (tasks, function calls) at the same time. Note that this does not mean that they are executed on different CPUs. Python threads will NOT make your program faster if it already uses 100 % CPU time. Threads are like mini-processes that live inside a process. They share memory space and efficiently read and write to the same variables. Two threads cannot execute code simultaneously in the same python program (although there are workarounds*)

The Threading Module
The newer threading module included with Python 2.4 provides much more powerful, high-level support for threads than the thread module discussed in the previous section.

The threading module exposes all the methods of the thread module and provides some additional methods −

threading.activeCount() − Returns the number of thread objects that are active.

threading.currentThread() − Returns the number of thread objects in the caller's thread control.

threading.enumerate() − Returns a list of all thread objects that are currently active.

In addition to the methods, the threading module has the Thread class that implements threading. The methods provided by the Thread class are as follows −

run() − The run() method is the entry point for a thread.

start() − The start() method starts a thread by calling the run method.

join([time]) − The join() waits for threads to terminate.

isAlive() − The isAlive() method checks whether a thread is still executing.

getName() − The getName() method returns the name of a thread.

setName() − The setName() method sets the name of a thread.

Creating Thread Using Threading Module
To implement a new thread using the threading module, you have to do the following −

Define a new subclass of the Thread class.

Override the __init__(self [,args]) method to add additional arguments.

Then, override the run(self [,args]) method to implement what the thread should do when started.

Once you have created the new Thread subclass, you can create an instance of it and then start a new thread by invoking the start(), which in turn calls run() method.

Algorithm:

1. Creating a child process and printing the parent and child PID

Step 1. Start
Step 2. N = fork()
Step 3. If N > 0:
print(PID) # patent process
Else:
print(PID) # child process
Step 4. Stop

2. Multithreading program to calculate square and factorial of a number

Function square(int x):

*Return x*x*

Function factorial(int x):

If x == 1:

Return 1

Else:

*Return factorial(x-1) * x*

Step 1. Start
Step 2. Import Thread Class
Step 3. Set Nums = list of numbers to be computed
Step 4. T1 = Thread(square, nums)
Step 5. T2 = Thread(factorial, nums)
Step 6. T1.start(), T2.start()
Step 7. T1.stop(), T2.stop()
Step 8. Stop

Program:

a. Creating a child process and printing the parent and child PID

```
import os
def driver():
    n = os.fork()
    if n > 0:
        print("Parent process : ", os.getpid())
    else:
        print("Child process : ", os.getpid())
driver()
```

b. Multithreading program to calculate square and factorial of a number

```
import time from threading
import Thread
def findsquare(nums):
    for i in nums:
        time.sleep(0.3)
        print(f"Square of {i}: {i*i}")
def fact(i):
    if i == 1:
        return 1
```

```
        else:
                return(i*fact(i-1))
def findfactorial(nums):
        for i in nums:
                time.sleep(0.2)
                print(f"Factorial of {i}: {fact(i)}")
nums = [4,5,2,10,3] [1,9,7,5,8]
t = time.time()
t1 = Thread(target=findsquare, args=(nums,))
t2 = Thread(target=findfactorial, args=(nums,))
t1.start()
t2.start()
t1.join()
t2.join()
print("Execution Finished, total time: ", time.time()-t)
```

Output:

1. Creating a child process and printing the parent and child PID

Parent process : 17194
Child process : 17195

2. Multithreading program to calculate square and factorial of a number

Factorial of 1: 1
Square of 1: 1
Factorial of 9: 362880
Square of 9: 81
Factorial of 7: 5040
Factorial of 5: 120
Square of 7: 49
Factorial of 8: 40320
Square of 5: 25
Square of 8: 64
Execution Finished, total time: 1.5001117892344011

Result:

The program to create a child process and print the parent and child PID as well as the program to compute the square and factorial of a number using multithreading has been executed successfully.