



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Experiment No. 12
Demonstrate the concept of Multi-threading
Date of Performance: 10/04/2024
Date of Submission: 17/04/2024



Experiment No. 12

Title: Demonstrate the concept of Multi-threading

Aim: To study and implement the concept of Multi-threading

Objective: To introduce the concept of Multi-threading in python

Theory:

Thread

In computing, a **process** is an instance of a computer program that is being executed. Any process has 3 basic components:

- An executable program.
- The associated data needed by the program (variables, work space, buffers, etc.)
- The execution context of the program (State of process)

A **thread** is an entity within a process that can be scheduled for execution. Also, it is the smallest unit of processing that can be performed in an OS (Operating System).

In simple words, a **thread** is a sequence of such instructions within a program that can be executed independently of other code. For simplicity, you can assume that a thread is simply a subset of a process!

A thread contains all this information in a **Thread Control Block (TCB)**:

- **Thread Identifier:** Unique id (TID) is assigned to every new thread
- **Stack pointer:** Points to thread's stack in the process. Stack contains the local variables under thread's scope.
- **Program counter:** a register which stores the address of the instruction currently being executed by thread.
- **Thread state:** can be running, ready, waiting, start or done.
- **Thread's register set:** registers assigned to thread for computations.
- **Parent process Pointer:** A pointer to the Process control block (PCB) of the process that the thread lives on.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Program:

```
# Python program to illustrate the concept
# of threading
# importing the threading module
import threading

def print_cube(num):
    """
    function to print cube of given num
    """
    print("Cube: {}".format(num * num * num))

def print_square(num):
    """
    function to print square of given num
    """
    print("Square: {}".format(num * num))

if __name__ == "__main__":
    # creating thread
    t1 = threading.Thread(target=print_square, args=(10,))
    t2 = threading.Thread(target=print_cube, args=(10,))
```



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

starting thread 1

t1.start()

starting thread 2

t2.start()

wait until thread 1 is completely executed

t1.join()

wait until thread 2 is completely executed

t2.join()

both threads completely executed

print("Done!")



Output:

A screenshot of a terminal window with a dark background. At the top, there is a navigation bar with several tabs: 'PORTS', 'SEARCH ERROR', 'PROBLEMS', 'DEBUG CONSOLE', 'OUTPUT', and 'TERMINAL'. The 'TERMINAL' tab is selected and highlighted with a yellow border. Below the tabs, the terminal shows a command prompt 'PS C:\Users\gawad\OneDrive\Desktop\php>' followed by the command 'python -u "c:\Users\gawad'. The output of the program is displayed on the next lines: 'Square: 100', 'Cube: 1000', and 'Done!'. The prompt returns to 'PS C:\Users\gawad\OneDrive\Desktop\php>'.

Conclusion:

Experiment demonstrated the concept of multithreading in Python, showcasing the ability to execute multiple tasks concurrently within a single process. By utilizing the threading module, the program executed separate functions concurrently, enhancing efficiency in computation. This experiment highlighted the benefits of multithreading in improving program performance by leveraging parallelism. Overall, the successful implementation of multithreading underscores its significance in optimizing resource utilization and enhancing program responsiveness.