

**Mawlana Bhashani Science and Technology University**

**Lab-Report**

Report No:02

Course code: ICT-3110

Course title: Operating System Lab

Date of Performance:01/09/2020

Date of Submission:10/09/2020

**Submitted by Submitted To**

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Name:Md Milon Ahmed

ID:IT-18034

3rd  year 1st semester

Session: 2017-2018

Dept. of ICT

MBSTU.

Experiment No:03

Experiment Name: Implementation of Threads.

Theory:

A thread is a path of execution within a process. A process can contain multiple threads.A thread is also known as lightweight process. The idea is to achieve parallelism by dividing a process into multiple threads. For example, in a browser, multiple tabs can be different threads. MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc. The primary difference is that threads within the same process run in a shared memory space, while processes run in separate memory spaces.Threads are not independent of one another like processes are, and as a result threads share with other threads their code section, data section, and OS resources (like open files and signals). But, like process, a thread has its own program counter (PC), register set, and stack space.

Working Procedure:

import threading

import time

class myThread (threading.Thread):

def \_\_init\_\_(self, threadID, name, counter):

threading.Thread.\_\_init\_\_(self)

self.threadID = threadID

self.name = name

self.counter = counter

def run(self):

print "Starting " + self.name

# Get lock to synchronize threads

threadLock.acquire()

print\_time(self.name, self.counter, 3)

# Free lock to release next thread

threadLock.release()

def print\_time(threadName, delay, counter):

while counter:

time.sleep(delay)

print "%s: %s" % (threadName, time.ctime(time.time()))

counter -= 1

threadLock = threading.Lock()

threads = []

# Create new threads

thread1 = myThread(1, "Thread-1", 1)

thread2 = myThread(2, "Thread-2", 2)

# Start new Threads

thread1.start()

thread2.start()

# Add threads to thread list

threads.append(thread1)

threads.append(thread2)

# Wait for all threads to complete

for t in threads:

t.join()

print "Exiting Main Thread"

Output:

Starting Thread-1

Starting Thread-2

Thread-1: Thu Mar 21 09:11:28 2013

Thread-1: Thu Mar 21 09:11:29 2013

Thread-1: Thu Mar 21 09:11:30 2013

Thread-2: Thu Mar 21 09:11:32 2013

Thread-2: Thu Mar 21 09:11:34 2013

Thread-2: Thu Mar 21 09:11:36 2013

Exiting Main Thread

Discussion:

In this lab, we will learn how to synchronize the actions of multiple threads. Synchronization is a means to ensure correct flow of execution between two or more threads working with shared data. We cover two types of synchronization: locking and waiting. Locking is used to prevent race condition between two or more threads to access shared data. It is used to ensure that only one thread that can access shared data at a time (to prevent race conditions).