**Thread concept: clone, threads of java**

**Subject - Unix Operating System**

**Name – Hemant Sharma**

**PRN – 22610001 Class – TYIT**

**Assignment No – 4(f)**

**Title-** Write a program using p thread library of Linux. Create three threads to take numbers and use join to print their average.

**Objectives:**

1. To learn about threading in Linux/Unix and Java and difference between them.
2. Use of system call/library to write effective programs.

**Theory:**

* **POSIX Threads (pthreads)**: POSIX threads provide a set of functions for creating and managing threads in Unix-like operating systems. Threads are lightweight processes that share the same address space and resources, which makes them ideal for tasks that can be parallelized.
* **Thread Creation and Synchronization**: In this program, pthread\_create is used to create the threads, and pthread\_join is used to synchronize the threads. By calling pthread\_join, the main thread waits for the completion of each thread, ensuring that all averages are calculated before the final result is printed.
* **Thread Efficiency**: Using threads in this scenario allows each set of numbers to be processed concurrently, which can result in better performance if the number of sets or the size of the data increases. However, for small tasks, the overhead of thread management might not show a significant improvement but demonstrates the use of concurrency.

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#define SIZE 5  // Number of elements each thread will handle

// Structure to hold data for each thread

typedef struct {

    int \*numbers;

    int count;

    double average;

} ThreadData;

// Function to compute average for each thread

void\* compute\_average(void\* arg) {

    ThreadData\* data = (ThreadData\*)arg;

    int sum = 0;

    for (int i = 0; i < data->count; i++) {

        sum += data->numbers[i];

    }

    data->average = (double)sum / data->count;

    pthread\_exit(NULL);

}

int main() {

    pthread\_t thread1, thread2, thread3;

    ThreadData data1, data2, data3;

    // Input arrays for each thread

    int numbers1[SIZE] = {1, 2, 3, 4, 5};

    int numbers2[SIZE] = {6, 7, 8, 9, 10};

    int numbers3[SIZE] = {11, 12, 13, 14, 15};

    // Assign data to thread structures

    data1.numbers = numbers1; data1.count = SIZE;

    data2.numbers = numbers2; data2.count = SIZE;

    data3.numbers = numbers3; data3.count = SIZE;

    // Create threads

    pthread\_create(&thread1, NULL, compute\_average, &data1);

    pthread\_create(&thread2, NULL, compute\_average, &data2);

    pthread\_create(&thread3, NULL, compute\_average, &data3);

    // Wait for threads to finish

    pthread\_join(thread1, NULL);

    pthread\_join(thread2, NULL);

    pthread\_join(thread3, NULL);

    // Print the computed averages

    printf("Average of Thread 1: %.2f\n", data1.average);

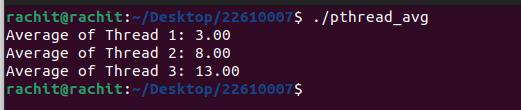
    printf("Average of Thread 2: %.2f\n", data2.average);

    printf("Average of Thread 3: %.2f\n", data3.average);

    return 0;

}

**Output:**

**Conclusion:**

This program demonstrates the use of POSIX threads to calculate averages concurrently for different sets of numbers. By creating threads with pthread\_create and synchronizing with pthread\_join, it efficiently computes the results in parallel, showcasing basic thread management and concurrency.