ASSIGNMENT 2

1. Write a C program to define 3 different threads with the following purposes where N is the input Thread A-To run a loop and return the sum of first N prime numbers. Thread B & C-should run in parallel. One prints "Thread 1 running" every 2 seconds, and the other prints "Thread 2 running" every 3 seconds for 100 seconds.

#include <stdio.h>

#include <pthread.h>

#include <unistd.h>

int N;

int isPrime(int num) {

if (num < 2) return 0;

for (int i = 2; i \* i <= num; i++)

if (num % i == 0)

return 0;

return 1;

}

void\* threadA(void\* arg) {

int sum = 0, count = 0, num = 2;

while (count < N) {

if (isPrime(num)) {

sum += num;

count++;

}

num++;

}

printf("Sum of first %d prime numbers is: %d\n", N, sum);

return NULL;

}

void\* threadB(void\* arg) {

for (int i = 0; i <= 100; i += 2) {

printf("Thread 1 running\n");

sleep(2);

}

return NULL;

}

void\* threadC(void\* arg) {

for (int i = 0; i <= 100; i += 3) {

printf("Thread 2 running\n");

sleep(3);

}

return NULL;

}

int main() {

pthread\_t t1, t2, t3;

printf("Enter value of N: ");

scanf("%d", &N);

pthread\_create(&t1, NULL, threadA, NULL);

pthread\_create(&t2, NULL, threadB, NULL);

pthread\_create(&t3, NULL, threadC, NULL);

pthread\_join(t1, NULL);

pthread\_join(t2, NULL);

pthread\_join(t3, NULL);

return 0;

}

1. In the above program, add signal handling for SIGINT (etc) and prevent termination. Convert the above threads to individual functions and note down the time taken. and the flow of execution.

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <signal.h>

#include <time.h>

int N;

void sigintHandler(int sig) {

printf("\n[!] Caught SIGINT (Ctrl+C). Program will not terminate.\n");

}

void\* calculatePrimes(void\* arg) {

int sum = 0, count = 0, num = 2;

while (count < N) {

int isPrime = 1;

for (int i = 2; i \* i <= num; i++) {

if (num % i == 0) {

isPrime = 0;

break;

}

}

if (isPrime) {

sum += num;

count++;

}

num++;

}

printf("[Thread A] Sum of first %d prime numbers is: %d\n", N, sum);

return NULL;

}

void\* thread1Printer(void\* arg) {

for (int i = 0; i <= 100; i += 2) {

printf("[Thread B] Thread 1 running at %d seconds\n", i);

sleep(2);

}

return NULL;

}

void\* thread2Printer(void\* arg) {

for (int i = 0; i <= 100; i += 3) {

printf("[Thread C] Thread 2 running at %d seconds\n", i);

sleep(3);

}

return NULL;

}

int main() {

pthread\_t tA, tB, tC;

signal(SIGINT, sigintHandler);

printf("Enter value of N (number of primes to sum): ");

scanf("%d", &N);

time\_t startTime, endTime;

time(&startTime);

printf("\n[Main] Starting all threads...\n");

pthread\_create(&tA, NULL, calculatePrimes, NULL);

pthread\_create(&tB, NULL, thread1Printer, NULL);

pthread\_create(&tC, NULL, thread2Printer, NULL);

pthread\_join(tA, NULL);

pthread\_join(tB, NULL);

pthread\_join(tC, NULL);

time(&endTime);

double totalTime = difftime(endTime, startTime);

printf("\n[Main] All threads completed.\n");

printf("[Main] Total execution time: %.2f seconds\n", totalTime);

return 0;

}

AREAS OF EXPLORATION

1. CHILD PROCESS

 fork() is a system call in C used to create a **new process**.

 When fork() is called, it creates a **child process** that runs alongside the parent.

 Both processes start executing from the point where fork() was called.

 Each process has a **unique process ID (PID)**.

 Commonly used in multitasking applications like servers or shells.

2.HANDLING COMMON SIGNALS

1. Signals are special messages sent to a program by the operating system.
2. Used to notify or control the program (e.g., stop, continue, terminate).
3. Common signals:
4. SIGINT – Sent when you press Ctrl+C.
5. SIGTERM – Standard signal to terminate a process.
6. SIGSEGV – Sent on a segmentation fault (e.g., invalid memory access).

3.KERNEL CRASHES

1. A kernel crash happens when the operating system encounters a critical error.
2. Causes:
3. Accessing invalid memory.
4. Running bad device drivers.
5. Faulty system calls from user programs.
6. In C, kernel crashes can be triggered by errors like:
7. Dereferencing a NULL pointer.
8. Stack overflows.
9. These crashes usually result in a system panic or core dump

4.TIME COMPLEXITY

1. Time complexity describes **how long a program takes** as the input size increases.
2. It's a way to measure the **efficiency** of algorithms.
3. Common notations:
   1. **O(1)** – Constant time (very fast).
   2. **O(n)** – Linear time.
   3. **O(n²)** – Quadratic time (slow for large inputs).
4. It helps you choose the **best algorithm** for performance.

5.LOCKING MECHANISM

* These are used to prevent data conflicts when multiple threads access shared resources.

🔹 Mutex (Mutual Exclusion):

* A locking mechanism that allows only one thread to access a resource at a time.
* If another thread tries to use it, it waits (blocks) until the mutex is unlocked.
* Safe but may be slower due to blocking.

🔹 Spinlock:

* Similar to mutex but instead of blocking, it keeps checking (spinning) until the resource is free.
* It uses CPU cycles while waiting, so it's faster for short waits, but not energy efficient.