**Programs on thread**

***Thread creation1***

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

void \*add(void \*arg){

printf("Thread called function stmt\n");

return NULL;}

void main(){

pthread\_t tid;

pthread\_create(&tid, NULL, add, NULL);

pthread\_join(tid, NULL); }

***Thread creation2***

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

void \*hello(){

printf("Welcome Everyone\n");}

void main(){

pthread\_t tid;

pthread\_create(&tid, NULL, hello, NULL);

pthread\_join(tid, NULL);}

***Thread creation3***

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

void \*sum(void \*val){

int \*val\_num = (int \*)(val);

printf("Value = %d\n", \*val\_num);}

void main(){

int val = 5;

pthread\_t tid;

pthread\_create(&tid, NULL, sum, (void \*)&val);

pthread\_join(tid, NULL);}

***program 4***

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

// Function to be executed by each thread

void \*thread\_function(void \*arg) {

int \*thread\_id = (int \*)arg;

printf("Thread %d: Hello, I am a new thread!\n", \*thread\_id);

return NULL; }

int main() {

pthread\_t threads[5]; // Array to hold thread identifiers

int thread\_ids[5]; // Array to pass unique IDs to threads

// Create 5 threads

for (int i = 0; i < 5; i++) {

thread\_ids[i] = i + 1;

if (pthread\_create(&threads[i], NULL, thread\_function, &thread\_ids[i]) != 0) {

perror("Failed to create thread");

return 1; } }

// Wait for all threads to complete

for (int i = 0; i < 5; i++) {

if (pthread\_join(threads[i], NULL) != 0) {

perror("Failed to join thread");

return 1; } }

printf("All threads have completed execution.\n");

return 0; }

***Program 5***

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

// Function to be executed by each thread

void \*thread\_function(void \*arg) {

int \*thread\_id = (int \*)arg;

printf("Thread %d: Executing...\n", \*thread\_id);

int \*result = malloc(sizeof(int));

\*result = \*thread\_id \* 10; // Example computation

pthread\_exit(result); } // Return result to the joining thread

int main() {

pthread\_t threads[3]; // Array to hold thread identifiers

int thread\_ids[3]; // Array to pass unique IDs to threads

// Create threads

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

thread\_ids[i] = i + 1;

if (pthread\_create(&threads[i], NULL, thread\_function, &thread\_ids[i]) != 0) {

perror("Failed to create thread");

return 1; } }

// Join threads and collect their results

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

int \*result;

if (pthread\_join(threads[i], (void \*\*)&result) != 0) {

perror("Failed to join thread");

return 1;

}

printf("Thread %d joined. Result: %d\n", thread\_ids[i], \*result);

free(result); // Free memory allocated by the thread

}

printf("All threads have completed execution.\n");

return 0;}