Πανεπιστήμιο Ιωαννίνων
Τμήμα Πληροφορικής & Τηλεπικοινωνιών
Μάθημα: Κατανεμημένα και Παράλληλα Συστήματα
Διδάσκων: Νικόλαος Καλλιμάνης

3η Εργαστηριακή Ενότητα

Προγραμματισμός με νήματα (Posix Threads)

Άσκηση 7: Δημιουργήστε ένα αρχείο με όνομα `exer07.c` και χρησιμοποιείστε τον παρακάτω κώδικα. Τι πρόκειται να εμφανίσει ο παρακάτω κώδικας;

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

#define LOOP_SIZE 10000000

volatile unsigned long shared_counter = 0;
volatile unsigned int num_threads = 0;

void *thread_work(void *arg) {
    long mytid = (long)arg;
    long iterations = LOOP_SIZE / num_threads;

    if (mytid == num_threads-1)
        iterations += LOOP_SIZE % num_threads;

    printf("Thread: %ld starts working\n", mytid);

    for (long i = 0; i < iterations; i++)
       shared_counter += 1;

    return NULL;
}

int main(int argc, char* argv[]) {
    if (argc != 2) {
        printf("Usage: %s <num_threads>\n", argv[0]);
        return EXIT_FAILURE;
    }
    num_threads = atoi(argv[1]);
    if (num_threads <= 0) {
        printf("Number of threads must be a positive integer.\n");
        return EXIT_FAILURE;
    }

    pthread_t *threads = malloc(num_threads * sizeof(pthread_t));
    // Create threads
    for (long t = 0; t < num_threads; t++)
        pthread_create(&threads[t], NULL, thread_work, (void *)t);

    for (long t = 0; t < num_threads; t++)
        pthread_join(threads[t], NULL);
    printf("\nMain thread: shared_counter: %ld\n", shared_counter);
    return 0;
}
```

Μεταγλωττίστε τον κώδικα με την εντολή `gcc -Wall -o exer07.run exer07.c -lpthread`

Άσκηση 8: Προσπαθήστε να διορθώσετε το σφάλμα της Άσκησης 7 χρησιμοποιώντας μηχανισμούς κλειδώματος mutexes που παρέχονται από την βιβλιοθήκη των POSIX threads. Δημιουργείστε τουλάχιστον δύο αρχεία με σωστές διαφορετικές εκδοχές (π.χ. exer08a.c και exer08b.c). Οι ακόλουθες συναρτήσεις/μεταβλητές υλοποιούν mutexes.

```
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
int pthread_mutex_destroy(pthread_mutex_t *mutex);
int pthread_mutex_init(pthread_mutex_t *restrict mutex, const pthread_mutexattr_t *restrict attr);
```

Άσκηση 9: Χρονομετρήστε τις δύο σωστές εκδοχές της Άσκησης 8, χρησιμοποιώντας τον κώδικα που παρατίθεται παρακάτω.

```
#include <time.h>

int64_t getTimeMillis(void) {
    struct timespec tm;

    if (clock_gettime(CLOCK_MONOTONIC, &tm) == -1) {
        perror("clock_gettime");
        return 0;
    } else return tm.tv_sec*1000LL + tm.tv_nsec/1000000LL;
}
```

Άσκηση 10: Δημιουργήστε ένα αρχείο με όνομα `exer10.c` και χρησιμοποιείστε τον παρακάτω κώδικα.
α) Τι πρόκειται να εμφανίσει ο παρακάτω κώδικας;
β) Δημιουργείστε μια πολύ-νηματική εκδοχή του παρακάτω κώδικα.

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

#define N 10

volatile double A[N][N];
volatile double B[N][N];
volatile double C[N][N];

void mat_mul2D(void) {
    int i, j, k;
    for (i = 0; i < N; ++i) {
        for (j = 0; j < N; j++) {
            C[i][j] = 0.0;
            for (k = 0; k < N; k++)
                C[i][j] += A[i][k] * B[k][j];
        }
    }
}

int main(int argc, char* argv[]) {
    for (long i = 0; i < N; i++) {
        for (long j = 0; j < N; j++) {
            A[i][j] = i + j;
            if (i == j) B[i][j] = 1.0;
            else B[i][j] = 0.0;
        }
    }

    mat_mul2D();
    if (N < 20) {
        for (long i = 0; i < N; i++) {
```

```
            for (long j = 0; j < N; j++) {
                printf("C[%ld][%ld] = %.2lf \t", i, j, C[i][j]);
            }
            printf("\n");
        }
    }
    return 0;
}
```