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

4η Εργαστηριακή Ενότητα

Προγραμματισμός με νήματα (Posix Threads) και σημαφόρους

Άσκηση 11: Δημιουργήστε ένα αρχείο με όνομα `exer11.c` και χρησιμοποιείστε τον παρακάτω κώδικα. Τι πρόκειται να εμφανίσει ο παρακάτω κώδικας;

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

#define NUM_THREADS 2
#define MAX_COUNT 1000000

volatile int counter = 0;
pthread_mutex_t mutex;

void *increment(void *threadid) {
    for (int i = 0; i < MAX_COUNT; ++i) {
        pthread_mutex_lock(&mutex);
        counter++;
        pthread_mutex_unlock(&mutex);
    }
    return NULL;
}

void *decrement(void *threadid) {
    for (int i = 0; i < MAX_COUNT; ++i) {
        while (counter == 0)
            ;
        pthread_mutex_lock(&mutex);
        counter--;
        pthread_mutex_unlock(&mutex);
    }
    return NULL;
}

int main() {
    pthread_t threads[NUM_THREADS];
    int rc;
    long t;

    // Initialize mutex
    pthread_mutex_init(&mutex, NULL);

    // Create increment thread
    rc = pthread_create(&threads[0], NULL, increment, (void *)0);
    if (rc) {
        printf("ERROR; return code from pthread_create() is %d\n", rc);
        exit(-1);
    }

    // Create decrement thread
    rc = pthread_create(&threads[1], NULL, decrement, (void *)1);
    if (rc) {
        printf("ERROR; return code from pthread_create() is %d\n", rc);
        exit(-1);
```

```
    }

    // Join threads
    for (t = 0; t < NUM_THREADS; ++t) {
        rc = pthread_join(threads[t], NULL);
        if (rc) {
            printf("ERROR; return code from pthread_join() is %d\n", rc);
            exit(-1);
        }
    }

    // Print final value of counter
    printf("Final value of counter: %d\n", counter);

    // Destroy mutex
    pthread_mutex_destroy(&mutex);

    pthread_exit(NULL);
}
```

Μεταγλωττίστε τον κώδικα με την εντολή `gcc -Wall -o exer11.run exer11.c -lpthread`

Άσκηση 12: Προσπαθήστε να δημιουργήσετε μια εκδοχή της Άσκησης 12 χρησιμοποιώντας σημαφόρους αντί για mutexes (κλειδώματα). Χρησιμοποιείστε τις ακόλουθες κλήσεις/συναρτήσεις για να το επιτύχετε.

```
#include <semaphore.h>

int sem_init(sem_t *sem, int pshared, unsigned int value);
int sem_destroy(sem_t *sem);
int sem_post(sem_t *sem);
int sem_wait(sem_t *sem);
```

Άσκηση 13: Χρονομετρείστε τις υλοποιήσεις της Άσκησης 11 και της Άσκησης 12. Ποια υλοποίηση είναι πιο γρήγορη; Για τη χρονομέτρηση χρησιμοποιείστε την παρακάτω συνάρτηση.

```
#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;
}
```