ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
www.cslab.ece.ntua.gr

ΣΥΣΤΗΜΑΤΑ ΠΑΡΑΛΛΗΛΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ
9ο εξάμηνο ΗΜΜΥ, ακαδημαϊκό έτος 2007-08

Εργαστηριακή Άσκηση 2: Συγχρονισμός Νημάτων

1 Βιβλιοθήκη pthreads

1.1 Ουρές Παραγωγών και Καταναλωτών

Μια από τις συχνά χρησιμοποιούμενες δομές στον πολυνηματικό προγραμματισμό είναι οι ουρές παραγωγών και καταναλωτών (producers-consumers queues). Αυτή η δομή επιτρέπει τη συνεργασία πολλαπλών νημάτων, από τα οποία ένα σύνολο παράγει τιμές με κάποιο τρόπο, ενώ τα υπόλοιπα τις καταναλώνουν (χρησιμοποιούν). **Ζητείται** η δημιουργία βιβλιοθήκης με τις κατάλληλες συναρτήσεις που υλοποιούν την παραπάνω λειτουργικότητα χρησιμοποιώντας *p*threads. Συγκεκριμένα, θα πρέπει να υλοποιηθούν δύο βασικές συναρτήσεις: μία για την εισαγωγή στοιχείων και μια για την εξαγωγή. Σε περίπτωση που δεν υπάρχουν στοιχεία στην ουρά η συνάρτηση εξαγωγής θα αναστείλει την εκτέλεση του καλούμενου νήματος έως ότου να υπάρξουν διαθέσιμα δεδομένα. Ακολουθεί παράδειγμα ορισμού των συναρτήσεων και της χρήσης τους.

```
pcq_t *pcq_create(void);                    // Αρχικοποίηση ουράς
void pcq_enqueue(pcq_t *pcq, void *data);   // Εισαγωγή στοιχείου
void *pcq_dequeue(pcq_t *pcq);              // Εξαγωγή στοιχείου

void *producer(void *arg){
    pcq_t pcq = (pcq_t *)arg;
    srand(time(NULL));
    int *v;
    for (;;){
        v = malloc(sizeof(int));
        *v = rand();
        pcq_enqueue(pcq, v);
    }
    return NULL;
}

void *consumer(void *arg){
    pcq_t pcq = (pcq_t *)arg;
    int total = 0, *v;
    for (;;){
        void *d = pcq_dequeue(pcq);
        v = (int *)d;
        total += *v;
        free(v);
    }
    return NULL;
}
```

Επιπρόσθετα, **ζητείται** η δημιουργία ενός προγράμματος που θα χρησιμοποιεί την παραπάνω βιβλιοθήκη. Το πρόγραμμα αυτό θα δημιουργεί νήματα παραγωγών και καταναλωτών, τα οποία θα χρησιμοποιούν μια ουρά για επικοινωνία. Το πρόγραμμα θα δέχεται τις εξής παραμέτρους: αριθμός νημάτων παραγωγών, αριθμός νημάτων καταναλωτών, ρυθμός κατανάλωσης ανά καταναλωτή, ρυθμός παραγωγής ανά παραγωγό

και αριθμός συνολικών παραγόμενων στοιχείων. Τα παραγόμενα στοιχεία θα είναι τυχαίοι αριθμοί, ενώ οι ρυθμοί παραγωγής και κατανάλωσης θα προσομοιώνονται με τη χρήση συναρτήσεων αναστολής λειτουργίας για συγκεκριμένο χρονικό διάστημα, όπως η `usleep()`. Κάθε νήμα με την περάτωσή του θα επιστρέφει τα στατιστικά της εκτέλεσης του, τα οποία θα τυπώνονται στην οθόνη από το κεντρικό νήμα.
Τι συμβαίνει στην περίπτωση όπου ο συνολικός ρυθμός παραγωγής είναι μεγαλύτερος από τον συνολικό ρυθμό κατανάλωσης; Πώς μπορεί να αντιμετωπισθεί το πρόβλημα αυτό; **Προαιρετικά ζητείται** η υλοποίηση των αντίστοιχων συναρτήσεων και παραδείγματος χρήσης τους.

1.2 Υλοποίηση Σηματοφορέων

Ζητείται η υλοποίηση μηχανισμού σηματοφορέων (*S*empahores), χρησιμοποιώντας τη βιβλιοθήκη *p*threads. Παράδειγμα των ορισμών των προς υλοποίηση συναρτήσεων παρουσιάζεται παρακάτω:

```
sema_t *sema_create(int value);
void sema_wait(sema_t *sema);
void sema_signal(sema_t *sema);
```

2 Συγχρονισμός με Σηματοφορείς

Έστω 10 νήματα που αντιπροσωπεύουν χορευτές και 1 νήμα που αντιπροσωπεύει τον κριτή σε έναν διαγωνισμό χορού. Οι χορευτές "χορεύουν" σε κάθε γύρο ταυτόχρονα έως ότου ο καθένας από αυτούς ολοκληρώσει το νούμερό του. Όταν και ο τελευταίος χορευτής ολοκληρώσει, ο κριτής επιλέγει έναν χορευτή, ο οποίος και αποκλείεται από τον επόμενο γύρο. Η διαδικασία συνεχίζεται μέχρι να μείνει μόνο ένας χορευτής, ο οποίος θα είναι και ο νικητής. **Ζητείται** ο αλγόριθμος συγχρονισμού των νημάτων αυτών χρησιμοποιώντας σηματοφορείς σε ψευδοκώδικα και κατόπιν η υλοποίησή της παραπάνω διαδικασίας, χρησιμοποιώντας τον κώδικα του ερωτήματος 1.2. Η διαδικασία του "χορού" θα προσομοιωθεί με μια τεχνητή καθυστέρηση, ενώ η επιλογή του χορευτή που δεν θα συνεχίσει θα γίνεται με τυχαίο τρόπο. Το τελικό πρόγραμμα θα πρέπει να τυπώνει στην οθόνη τα κατάλληλα μηνύματα, ώστε να είναι φανερή η διαδικασία που ακολουθείται. **Προαιρετικά ζητείται** η επέκταση του αλγόριθμου συγχρονισμού, ώστε να περιλαμβάνει 3 νήματα κριτών. Στην περίπτωση που και οι 3 κριτές έχουν διαφορετική άποψη, η άποψη του πρώτου υπερισχύει.