ΕΡΓΑΣΤΗΡΙΟ ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ ΙΙ
Εαρινό Εξάμηνο 2024-25 - Άσκηση #2 (17,5%)
Ημ. Παράδοσης: 1/6/2025

Α.

Γράψτε ένα πρόγραμμα σε C, το οποίο θα χρησιμοποιεί την βιβλιοθήκη των POSIX Threads, και θα υπολογίζει *«παράλληλα»* το 'μέγιστο' στοιχείο (*maximum*) ενός διανύσματος *X* μήκους ***n***. Ο ζητούμενος υπολογισμός θα πρέπει να γίνεται με τη βοήθεια ***t*** threads κάθε ένα από τα οποία θα υπολογίζει το επιμέρους μέγιστο ***n/t*** όρων του διανύσματος (όπου τα *t, n, x_i* θα πρέπει να τα δίνει ο χρήστης – θεωρήστε επίσης ότι το ***n*** είναι ακέραιο πολλαπλάσιο του ***t***). Πιο συγκεκριμένα, κάθε thread θα πρέπει **(α)** να υπολογίζει τοπικά (σε μια τοπική μεταβλητή – π.χ. *local_max*) το επιμέρους μέγιστο που του αναλογεί και **(β)** να ενημερώνει στο τέλος με το τοπικό του μέγιστο μια κοινή μεταβλητή (π.χ. *total_max*) η οποία θα αντιπροσωπεύει το τελικό μέγιστο. Το τελικό αυτό μέγιστο θα πρέπει στο τέλος να τυπώνεται στην οθόνη. Εξετάστε επίσης μεταξύ των άλλων αν υπάρχει κάποιο «κρίσιμο τμήμα» στον κώδικά σας που απαιτεί προστασία (αμοιβαίο αποκλεισμό), και αν ναι, συμπεριλάβετε στη λύση σας και τον κώδικα για την προστασία του κρίσιμου αυτού τμήματος.

Υπόδειξη: Ξεκινήστε θεωρώντας αρχικά την πιο απλή περίπτωση κατά την οποία ισχύει ***n=t*** (κάθε thread να εκτελεί δηλαδή τον κώδικα υπολογισμού μεγίστου για έναν όρο μόνο του διανύσματος), συμβουλευόμενοι μεταξύ των άλλων και το παράδειγμα *'factorial-ex.c'* που είναι αναρτημένο στο eclass. Προσπαθήστε στη συνέχεια να επεκτείνετε το πρόγραμμά σας για ***n>t*** σύμφωνα με τις οδηγίες που δίνονται παραπάνω. Δοκιμάστε στη συνέχεια να τρέξετε το πρόγραμμά σας για μεγάλα και πολύ μεγάλα *n* (χρησιμοποιώντας π.χ. για είσοδο αντίστοιχα διανύσματα αρχικοποιημένα με τυχαίους αριθμούς) και μετρήστε τους χρόνους εκτέλεσης για ένα και περισσότερα από ένα threads (π.χ. 1, 2, 4 και 8 threads).

Β.

Γράψτε ένα πρόγραμμα σε C, το οποίο θα χρησιμοποιεί την βιβλιοθήκη των *POSIX Threads*, και θα έχει ως αποτέλεσμα να τυπώνεται επαναληπτικά η ακολουθία:

<*one*> <*two*> <*three*> <*one*> <*two*> <*three*> <*one*> <*two*> <*three*> …

Για να πετύχετε το παραπάνω θα πρέπει αρχικά να εκκινήσετε στο πρόγραμμά σας τρία (3) διαφορετικά threads (το ένα να τυπώνει **<*one*>**, το άλλο να τυπώνει **<*two*>** και το τρίτο να τυπώνει **<*three*>**), και στη συνέχεια να τα συγχρονίσετε κατάλληλα μεταξύ τους.

Για τον απαιτούμενο συγχρονισμό μπορείτε να χρησιμοποιήσετε είτε **σημαφόρους** (POSIX semaphores) είτε **μεταβλητές συνθήκης** (condition variables), ότι επιθυμείτε (θεωρείτε προτιμότερο για κάποιο λόγο) από τα δύο. Αιτιολογήστε σύντομα το σκεπτικό της επιλογής σας.

Γ.

Γράψτε δύο προγράμματα σε C, ένα πρόγραμμα **server** και ένα πρόγραμμα **client** (το οποίο θα μπορούν να το εκτελούν εν δυνάμει πολλοί clients – και να εξυπηρετούνται *ταυτόχρονα* από το server), τα οποία θα μπορούν να επικοινωνούν μεταξύ τους (με χρήση UNIX-domain stream sockets) επαναληπτικά ως εξής:

- Ο client θα στέλνει στο server μία ακολουθία ακεραίων αριθμών μήκους N, την οποία θα διαβάζει σαν είσοδο από το χρήστη.
- Ο server αφού παραλάβει την ακολουθία θα υπολογίζει το μέσο όρο της και αν αυτός είναι πάνω από 20 θα στέλνει πίσω στον client τόσο (α) τον μέσο όρο που βρήκε όσο και (β) ένα κατάλληλο μήνυμα αποδοχής (π.χ. 'Sequence Ok'). Ειδάλλως θα στέλνει μόνο ένα μήνυμα αποτυχίας (π.χ. 'Check Failed').
- Ο client θα πρέπει απλά μετά από κάθε επικοινωνία να τυπώνει την απάντηση του έστειλε ο server στην οθόνη, και να ζητά την επόμενη ακολουθία από το χρήστη. Η επικοινωνία θα τελειώνει (από πλευράς του client) όταν ο χρήστης δηλώσει ότι δεν επιθυμεί να δώσει άλλη ακολουθία προς έλεγχο.

Παραδοτέα:

*Τις απαντήσεις σας θα πρέπει να τις δώσετε σύμφωνα με τα υπαγορευόμενα στο συνοδευτικό αρχείο '**guidelines-ls2-ask2.pdf**' (είναι αναρτημένο στο Eclass στον ίδιο κατάλογο με το αρχείο της παρούσας εκφώνησης).*