



Εργασία 1 (υποχρεωτική) –Προγραμματισμός με Pthreads

ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2025 – 2026

(ΕΚΦΩΝΗΣΗ) ΠΕΜΠΤΗ 13 ΝΟΕΜΒΡΙΟΥ 2025

(ΠΑΡΑΔΟΣΗ ΣΤΟ ECLASS ΜΕΧΡΙ) ΔΕΥΤΕΡΑ 1 ΔΕΚΕΜΒΡΙΟΥ 2025

Πληροφορίες για τις Υποχρεωτικές Εργασίες του μαθήματος

- Κάθε ομάδα μπορεί να αποτελείται **από 1 ή 2 φοιτητές**. Όλα τα μέλη της ομάδας πρέπει να έχουν ισότιμη συμμετοχή και να γνωρίζουν τις λεπτομέρειες της υλοποίησης της ομάδας.
- Για την εξεταστική Σεπτεμβρίου δε θα δοθούν άλλες εργασίες. Τον Σεπτέμβριο εξετάζεται μόνο το γραπτό.
- Επίσημη υπολογιστική πλατφόρμα του μαθήματος είναι το δίκτυο των υπολογιστών του Τμήματος με λειτουργικό σύστημα Linux Ubuntu (linux01.di.uoa.gr έως linux30.di.uoa.gr).
- Για την ανάπτυξη των προγραμμάτων σας καλείστε να χρησιμοποιήσετε τη γλώσσα προγραμματισμού C.
- Μαζί με τον κώδικα σας καλείστε να υποβάλετε και τα σχετικά αρχεία Makefile.
- Στην αναφορά σας καλείστε να δώσετε πληροφορίες σχετικά με το όνομα του υπολογιστικού συστήματος που χρησιμοποιείτε, καθώς επίσης και το μοντέλο επεξεργαστή, τον αριθμό των πυρήνων, την έκδοση του λειτουργικού συστήματος, και την έκδοση του μεταγλωττιστή. Για τα πειραματικά δεδομένα που παρουσιάζετε, καλείστε να αναφέρετε ρητά στα εκάστοτε σημεία της αναφοράς τις εισόδους που χρησιμοποιήσατε. Καλείστε να εκτελέσετε κάθε πείραμα (το οποίο ορίζεται ως η εκτέλεση ενός προγράμματος για συγκεκριμένο αριθμό νημάτων και παραμέτρους εισόδου) πολλές φορές (για παράδειγμα, 4 φορές) και να παρουσιάσετε τον μέσο όρο των αποτελεσμάτων (σύσταση: δημιουργήστε scripts για την εκτέλεση των πειραμάτων, ακόμα και για την επεξεργασία των αποτελεσμάτων και την δημιουργία γραφημάτων).
- Προαιρετικά, μπορείτε να υποβάλετε και τα scripts που χρησιμοποιήσατε για να τρέξετε τα πειράματα και να δημιουργήσετε τα σχετικά γραφήματα.
- Καλείστε να προσεγγίσετε την κάθε άσκηση στην αναφορά σας ως εξής: περιγραφή προβλήματος, σύντομη περιγραφή της λύσης σας, παράθεση πειραματικών αποτελεσμάτων (χρήση πινάκων ή γραφημάτων), και σχολασμός αποτελεσμάτων.
- Σε περίπτωση αντιγραφής θα μηδενίζονται όλες οι ομάδες που μετέχουν σε αυτή.
- Η παράδοση της **Εργασίας** πρέπει να γίνει μέχρι τα **μεσάνυχτα της προθεσμίας ηλεκτρονικά** και μόνο στο eclass (να ανεβάσετε ένα μόνο αρχείο zip ή rar με την αναφορά σας σε PDF και τον κώδικα σας). **Μην περιμένετε μέχρι την τελευταία στιγμή**.

Άσκηση 1.1

Σε αυτή την άσκηση καλείστε να γράψετε ένα πρόγραμμα Pthreads το οποίο υπολογίζει τον πολλαπλασιασμό πολυωνύμων. Πιο συγκεκριμένα, καλείστε να γράψετε ένα πρόγραμμα το οποίο: (i) δημιουργεί δύο τυχαία πλήρη πολυωνύμα η βαθμού οι συντελεστές των οποίων είναι ακέραιοι μη-μηδενικοί αριθμοί, (ii) τα πολλαπλασιάζει χρησιμοποιώντας τον σειριακό αλγόριθμο, (iii) τα πολλαπλασιάζει χρησιμοποιώντας τον παράλληλο αλγόριθμο, και (iv) επιβεβαίωνει ότι ο σειριακός και ο παράλληλος αλγόριθμος παράγουν τα ίδια αποτελέσματα. Το πρόγραμμά σας θα λαμβάνει σαν ορίσματα τον βαθμό η του κάθε πολυωνύμου και τον αριθμό των νημάτων, ενώ σαν έξοδο θα τυπώνει τον χρόνο δημιουργίας και αρχικοποίησης των πολυωνύμων (καθώς και όποιων άλλων δομών δεδομένων χρειάζεστε), τον χρόνο εκτέλεσης του σειριακού αλγόριθμου, τον χρόνο εκτέλεσης του παράλληλου αλγόριθμου, και την επιβεβαίωση της σωστής εκτέλεσης του παράλληλου αλγόριθμου σε σχέση με τον σειριακό. Συγκρίνετε την απόδοση του παράλληλου προγράμματος σας για διάφορους βαθμούς πολυωνύμων n (για παράδειγμα, 10^5 ή 10^6) και για διαφορετικό αριθμό νημάτων σε σχέση με τον σειριακό αλγόριθμο. Παρατηρείτε επιτάχυνση και γιατί; Χρειάστηκε να χρησιμοποιήσετε συγχρονισμό και γιατί; Στην άσκηση αποφύγετε την επίλυση μέσω Fast Fourier Transform.

Άσκηση 1.2

Σε αυτή την άσκηση καλείστε να γράψετε ένα πρόγραμμα Pthreads στο οποίο όλα τα νήματα ανανεώνουν μια κοινόχροστη μεταβλητή. Αυτή η κοινόχροστη μεταβλητή αρχικοποιείται στη τιμή ο από την main. Το κάθε νήμα υλοποιεί ένα for loop με σταθερό αριθμό επαναλήψεων, και σε κάθε επανάληψη αυξάνει κατά ένα την κοινόχροστη μεταβλητή. Η main τυπώνει την τελική τιμή η οποία θα πρέπει να είναι ντετερμινιστική. Επειδή προκύπτει ανταγωνισμός στην ενημέρωση της κοινόχροστης μεταβλητής, καλείστε να υλοποιήσετε τις εξής τρεις προσεγγίσεις διασφάλισης της ορθής εκτέλεσης του προγράμματος: (i) με τη χρήση κλειδωμάτων αμοιβαίου αποκλεισμού, (ii) με τη χρήση κλειδωμάτων ανάγνωσης-εγγραφής, και (iii) με τη χρήση ατομικών εντολών (περισσότερες πληροφορίες σχετικά με τη χρήση τους μπορείτε να βρείτε στον εξής σύνδεσμο: https://gcc.gnu.org/onlinedocs/gcc/_005f_005fatomic-Builtins.html). Συγκρίνετε την επίδοση των τριών προσεγγίσεων για διαφορετικό αριθμό επαναλήψεων και διαφορετικό αριθμό νημάτων και σχολιάστε αν προκύπτουν διαφορές ανάμεσα στις τρεις προσεγγίσεις και γιατί.

Άσκηση 1.3

Σε αυτή την άσκηση καλείστε να γράψετε ένα πρόγραμμα Pthreads το οποίο αναλύει τα δεδομένα τεσσάρων διαφορετικών πινάκων ακεραίων χρησιμοποιώντας την κοινόχρηστη μεταβλητή `array_stats` τύπου `struct array_stats_s` όπως ορίζεται παρακάτω:

```
struct array_stats_s {  
    long long int info_array_0;  
    long long int info_array_1;  
    long long int info_array_2;  
    long long int info_array_3;  
} array_stats;
```

Πιο συγκεκριμένα, το πρόγραμμά σας θα λαμβάνει σαν όρισμα τον αριθμό των στοιχείων που θα έχει ο κάθε πίνακας (και οι τέσσερις πίνακες έχουν ίσο αριθμό στοιχείων) και στη συνέχεια θα δεσμεύει μνήμη για τους πίνακες και θα τους αρχικοποιεί με τυχαίους ακέραιους αριθμούς που έχουν τιμές 0-9. Έπειτα, θα δημιουργεί ακριβώς 4 νήματα και το κάθε νήμα θα αναλαμβάνει αποκλειστικά την ανάλυση του κάθε πίνακα. Η ανάλυση περιλαμβάνει την ακριβή καταμέτρηση των μη-μηδενικών στοιχείων του κάθε πίνακα. Κάθε φορά που ένα νήμα εντοπίζει ένα μη-μηδενικό στοιχείο του πίνακα που έχει αναλάβει, αυξάνει απευθείας το αντίστοιχο πεδίο της κοινόχρηστης δομής `array_stats` (δηλαδή, το πεδίο `info_array_0`, για τον πίνακα 0, το πεδίο `info_array_1`, για τον πίνακα 1, κ.ο.κ.). Επίσης, υλοποιήστε μια συνάρτηση που πραγματοποιεί ακριβώς την ίδια ανάλυση και για τους 4 πίνακες σειριακά. Το πρόγραμμά σας στην έξοδο θα τυπώνει τον χρόνο δημιουργίας και αρχικοποίησης των πινάκων (καθώς και όποιων άλλων δομών δεδομένων χρειάζεστε), τον χρόνο εκτέλεσης του σειριακού αλγόριθμου, τον χρόνο εκτέλεσης του παράλληλου αλγόριθμου, και την επιβεβαίωση της σωστής εκτέλεσης του παράλληλου αλγόριθμου σε σχέση με τον σειριακό.

Συγκρίνετε την απόδοση του παράλληλου προγράμματος σας για διάφορα μεγέθη πινάκων (για παράδειγμα, 10^5 ή 10^6 στοιχεία) σε σχέση με τον σειριακό αλγόριθμο. Παρατηρείτε επιτάχυνση και γιατί; Χρειάστηκε να χρησιμοποιήσετε συγχρονισμό και γιατί; Μπορείτε να βελτιώσετε την επίδοση της παράλληλης εκτέλεσης τροποποιώντας τη δομή `array_stats`;

Άσκηση 1.4

Σε αυτή την άσκηση καλείστε να γράψετε ένα πρόγραμμα Pthreads το οποίο προσομοιώνει τη λειτουργία μιας τράπεζας. Πιο συγκεκριμένα, το πρόγραμμα χρησιμοποιεί έναν κοινόχρηστο πίνακα η στοιχείων, όπου το κάθε στοιχείο αποθηκεύεται το υπόλοιπο κάθε λογαριασμού. Αρχικά, το πρόγραμμα δεσμεύει δυναμικά τον κοινόχρηστο πίνακα και τον αρχικοποιεί με τυχαίους ακέραιους αριθμούς. Στη συνέχεια, το πρόγραμμά σας θα δημιουργεί πολλαπλά νήματα, και το κάθε νήμα θα αναλαμβάνει να πραγματοποιήσει έναν συγκεκριμένο αριθμό συναλλαγών. Οι συναλλαγές μπορεί να είναι είτε (i) συναλλαγές μεταφοράς χρημάτων, ή (ii) συναλλαγές ερώτησης υπολοίπου. Για κάθε συναλλαγή μεταφοράς χρημάτων, το νήμα θα υπολογίζει με τυχαίο τρόπο από ποιον λογαριασμό θα μεταφέρει χρήματα, σε ποιον λογαριασμό θα τα μεταφέρει, καθώς και το ποσό που θα μεταφέρει, και έπειτα θα πραγματοποιεί τη συναλλαγή. Για κάθε συναλλαγή υπολοίπου, το νήμα θα υπολογίζει με τυχαίο τρόπο από ποιον λογαριασμό θα διαβάσει το υπόλοιπο του και απλά θα διαβάζει το αντίστοιχο στοιχείο του πίνακα και θα ενημερώνει μια ιδιωτική μεταβλητή που θα έχει το άθροισμα όλων των υπολοίπων που θα έχει διαβάσει.

Για να πραγματοποιηθούν οι συναλλαγές με ασφαλή τρόπο, θα πρέπει να χρησιμοποιήσετε συγχρονισμό. Καλείστε να υλοποιήσετε διάφορα σχήματα συγχρονισμού και να πειραματιστείτε με το `granularity` των κλειδωμάτων (`coarse-grained` vs `fine-grained locking`), καθώς και με τη χρήση κλειδωμάτων αμοιβαίου αποκλεισμού και κλειδωμάτων ανάγνωσης-εγγραφής.

Το πρόγραμμά σας θα λαμβάνει σαν ορίσματα: (i) τον αριθμό των στοιχείων που θα έχει ο κοινόχρηστος πίνακας, (ii) τον αριθμό των συναλλαγών που θα πραγματοποιήσει το κάθε νήμα, (iii) το ποσοστό των συναλλαγών ερώτησης υπολοίπου που θα πραγματοποιηθούν επί του αριθμού των συνολικών συναλλαγών, (iv) την προσέγγιση κλειδωμάτων που θα χρησιμοποιηθεί, και (v) τον αριθμό των νημάτων. Το πρόγραμμά σας θα τυπώνει στην έξοδο το χρόνο της παράλληλης εκτέλεσης και θα επιβεβαιώνει τη σωστή λειτουργία του προγράμματος.

Καλείστε να αναλύσετε τα αποτελέσματα που προκύπτουν για σταθερό αριθμό νημάτων (για παράδειγμα, 4 νήματα) καθώς αλλάζετε διάφορες παραμέτρους του προγράμματος, όπως το μέγεθος του κοινόχρηστου πίνακα των λογαριασμών, ο αριθμός των συναλλαγών ανά νήμα, και η αναλογία των συναλλαγών μεταφοράς χρημάτων/ερώτησης υπολοίπου.

Τροποποιήστε το πρόγραμμά σας ώστε κάθε φορά που ένα νήμα πραγματοποιεί μια συναλλαγή ερώτησης υπολοίπου να κάνει κάποια επιπλέον λειτουργία με την τιμή αυτή (για παράδειγμα, κλήση κάποιας μαθηματικής συνάρτησης) ή απλά να κοιμάται για μερικά μικροδευτερόλεπτα. Με άλλα λόγια, αυξήστε τη διάρκεια του κρίσιμου τμήματος της συναλλαγής ερώτησης υπολοίπου. Τι παρατηρείτε τώρα όσον αφορά τη σύγκριση κλειδωμάτων αμοιβαίου αποκλεισμού και κλειδωμάτων ανάγνωσης-εγγραφής;

Προαιρετικά: Υλοποιήστε τη δική σας βιβλιοθήκη κλειδωμάτων ανάγνωσης-εγγραφής η οποία να συνδυάζει με αυτόματο τρόπο τα οφέλη επίδοσης των κλειδωμάτων αμοιβαίου αποκλεισμού και των κλειδωμάτων ανάγνωσης-εγγραφής με Pthreads για διαφορετικές παραμέτρους εκτέλεσης με βάση τα χαρακτηριστικά που έχετε παρατηρήσει (για παράδειγμα, με βάση τη διάρκεια του κρίσιμου τμήματος ανάγνωσης ή με βάση τη συχνότητα εκτέλεσης κρίσιμων τμημάτων ανάγνωσης σε σχέση με τα κρίσιμα τμήματα εγγραφής).

Άσκηση 1.5

Σε αυτή την άσκηση καλείστε να: (i) πειραματιστείτε με την υλοποίηση φράγματος (barrier) που παρέχει η βιβλιοθήκη Pthreads (pthread_barrier), (ii) υλοποιήσετε φράγμα με τη χρήση κλειδώματος και μεταβλητής συνθήκης, (iii) να υλοποιήσετε φράγμα που χρησιμοποιεί αναμονή σε εγρήγορση αλλά είναι επαναχρησιμοποιούμενο.

Πιο συγκεκριμένα, γράψτε ένα πρόγραμμα με τη χρήση Pthreads στο οποίο όλα τα νήματα εκτελούν έναν βρόχο επανάληψης για έναν μεγάλο αριθμό επαναλήψεων N (για παράδειγμα, 10^5 ή 10^6) και σε κάθε επανάληψη του βρόχου τα νήματα συγχρονίζονται χρησιμοποιώντας την ίδια μεταβλητή φράγματος, όπως φαίνεται και στον ψευδοκώδικα που ακολουθεί.

```
for (int i = 0; i < N ; i++) {  
    barrier_wait(&barrier);  
}
```

Στη συνέχεια, γράψτε ένα δεύτερο πρόγραμμα Pthreads που εκτελεί ακριβώς την ίδια εργασία αλλά χρησιμοποιεί τη δική σας υλοποίηση φράγματος η οποία βασίζεται στη χρήση κλειδώματος και μεταβλητής συνθήκης. Χρησιμοποιήστε τη σχετική υλοποίηση που περιλαμβάνεται στο βιβλίο και είδαμε στο μάθημα. Για ευκολία, δείτε το σχετικό πρόγραμμα από τα παραδείγματα του βιβλίου (pth_cond_bar.c). Προσέξτε πως αυτή η υλοποίηση του φράγματος δεν στηρίζεται στην αναμονή σε εγρήγορση αλλά επιτρέπει την επαναχρησιμοποίηση του ίδιου φράγματος.

Τέλος, γράψτε ένα τρίτο πρόγραμμα Pthreads που εκτελεί ακριβώς την ίδια εργασία, αλλά αυτή τη φορά χρησιμοποιεί φράγμα που βασίζεται σε αναμονή σε εγρήγορση αλλά είναι επαναχρησιμοποιούμενο. Πιο συγκεκριμένα, υλοποιήστε και χρησιμοποιήστε το sense-reversal centralized barrier (περισσότερες πληροφορίες μπορείτε να δείτε στον εξής σύνδεσμο [https://en.wikipedia.org/wiki/Barrier_\(computer_science\)](https://en.wikipedia.org/wiki/Barrier_(computer_science)), στο Κεφάλαιο 5 του βιβλίου «Parallel Computer Architecture: A Hardware/Software Approach», Morgan Kaufmann Publishers Inc., 1998 των D. Culler, J. Singh, A. Gupta, καθώς επίσης και στο βιβλίο «Highly Parallel Computing», Benjamin-Cummings Pub Co., 1994 των G. Almasi and A. Gottlieb, και στην εργασία «Two algorithms for barrier synchronization» International Journal of Parallel Programming, 17(1):1-17, February 1988 των D. Hensgen, R. Finkel, and U. Manber.

Εκτελέστε τα τρία προγράμματα για διαφορετικό αριθμό νημάτων και επαναλήψεων. Τι παρατηρείτε όσον αφορά την επίδοση μεταξύ των τριών υλοποιήσεων; Τι συμβαίνει όταν ο αριθμός των νημάτων είναι μεγαλύτερος από τον αριθμό των πυρήνων; Γιατί το sense-reversal centralized barrier, αν και βασίζεται σε αναμονή σε εγρήγορση, είναι επαναχρησιμοποιούμενο?