Εθνικο & Καποδιστριακο Πανεπιστημιο Αθηνων
Τμημα Πληροφορικης & Τηλεπικοινωνιων
Μαθημα: **Παραλληλα Συστηματα**

Εργασία 1 (υποχρεωτική) –Προγραμματισμός με Pthreads

Ακαδημαϊκο Ετος 2024 – 2025
(εκφωνηση) Παρασκευη 15 Νοεμβριου 2024
(παραδοση στο eclass μεχρι) **Δευτερα 2 Δεκεμβριου 2024**

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).
- Μαζί με τον κώδικά 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 σχολιασμός 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 κώδικά σας). **Μην περιμένετε μέχρι την τελευταία στιγμή. Η εκφώνηση της επόμενης εργασίας θα ανατεθεί αμέσως μετά.**

Άσκηση 1.1

Υποθέστε ότι πετάμε βελάκια σε έναν τετράγωνο στόχο του οποίου οι πλευρές έχουν μήκος 2 μέτρα, με το κέντρο του στόχου να αποτελεί το σημείο αρχής (0, 0) του συστήματος συντεταγμένων. Υποθέστε επίσης ότι σε αυτόν τον τετράγωνο στόχο είναι εγγεγραμμένος ένας κύκλος. Η ακτίνα του κύκλου είναι 1 μέτρο και το εμβαδόν του είναι π τετραγωνικά μέτρα. Αν τα σημεία στα οποία καταλήγουν τα βέλη είναι ομοιόμορφα κατανεμημένα (και τα βέλη πετυχαίνουν πάντα τον τετράγωνο στόχο), τότε το πλήθος των βελών που πετυχαίνουν το εσωτερικό του κύκλου θα πρέπει, κατά προσέγγιση, να ικανοποιεί την εξίσωση:

$$\frac{\pi\lambda\acute{\eta}\theta o\varsigma\_\beta\varepsilon\lambda\acute{\omega}\nu\_\varepsilon\nu\tau\acute{o}\varsigma\_\kappa\acute{\upsilon}\kappa\lambda o\upsilon}{\pi\lambda\acute{\eta}\theta o\varsigma\_\rho\acute{\iota}\psi\varepsilon\omega\nu} = \frac{\pi}{4}$$

αφού ο λόγος του εμβαδού του κύκλου προς το εμβαδόν του τετραγώνου είναι π/4.

Μπορούμε να χρησιμοποιήσουμε αυτή την εξίσωση για να υπολογίσουμε μια εκτίμηση για την τιμή του π μέσω μιας γεννήτριας τυχαίων αριθμών:

```
βέλη_κύκλου = 0;
for (ρίψη = 0; ρίψη < πλήθος_ρίψεων; ρίψη++) {
      x = τυχαίος double μεταξύ -1 και 1;
      y = τυχαίος double μεταξύ -1 και 1;
      τετράγωνο_απόστασης = x*x + y*y;
      if (τετράγωνο_απόστασης <= 1)
            βέλη_κύκλου++;
}
εκτίμηση_π = 4*βέλη_κύκλου/((double) πλήθος_ρίψεων);
```

Αυτή η μέθοδος ονομάζεται «Μόντε Κάρλο» επειδή χρησιμοποιεί τυχαίες μεταβλητές (οι ρίψεις των βελών).

Γράψτε ένα πρόγραμμα που να χρησιμοποιεί τη μέθοδο Μόντε Κάρλο για την εκτίμηση της τιμής του π με σειριακό αλγόριθμο και με χρήση Pthreads. Το πρόγραμμά σας θα πρέπει να λαμβάνει σαν ορίσματα το πλήθος των ρίψεων και τον αριθμό των νημάτων. Το πρόγραμμά σας θα υπολογίζει την εκτίμηση για την τιμή του π χρησιμοποιώντας πρώτα τον σειριακό αλγόριθμο και έπειτα τον παράλληλο αλγόριθμό σας. Το πρόγραμμά σας θα πρέπει να τυπώνει τον χρόνο εκτέλεσης του σειριακού αλγόριθμου και του παράλληλου αλγόριθμου ξεχωριστά. Προτιμήστε τον τύπο long long int για το πλήθος των βελών εντός του κύκλου και για το πλήθος των ρίψεων, αφού και οι δύο αυτοί αριθμοί θα πρέπει να είναι πολύ μεγάλοι (για παράδειγμα, το πλήθος ρίψεων να είναι 10^8 ή 10^9) για να πάρετε μια καλή εκτίμηση για το π.

Χρειάστηκε να χρησιμοποιήσετε συγχρονισμό και γιατί; Συγκρίνετε την απόδοση του παράλληλου προγράμματος σας για διαφορετικό αριθμό νημάτων και διαφορετικό αριθμό πλήθος ρίψεων σε σχέση με τον σειριακό αλγόριθμο. Παρατηρείτε επιτάχυνση και γιατί;

Άσκηση 1.2

Γράψτε ένα πρόγραμμα με τη χρήση Pthreads στο οποίο όλα τα νήματα ανανεώνουν μια κοινόχρηστη μεταβλητή. Αυτή η κοινόχρηστη μεταβλητή αρχικοποιείται στη τιμή 0 από την main. Το κάθε νήμα υλοποιεί ένα for loop με σταθερό αριθμό επαναλήψεων, και σε κάθε επανάληψη αυξάνει κατά ένα την κοινόχρηστη μεταβλητή. Η main τυπώνει την τελική τιμή η οποία θα πρέπει να είναι ντετερμινιστική. Επειδή προκύπτει ανταγωνισμός στην ενημέρωση της κοινόχρηστης μεταβλητής, καλείστε να υλοποιήσετε τις εξής δύο προσεγγίσεις διασφάλισης της ορθής εκτέλεσης του προγράμματος: (1) με τη χρήση pthread κλειδωμάτων, και (2) 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 δύο προσεγγίσεων για διαφορετικό αριθμό επαναλήψεων και διαφορετικό αριθμό νημάτων και σχολιάστε αν προκύπτουν διαφορές ανάμεσα στις δύο προσεγγίσεις και γιατί.

Άσκηση 1.3

Τροποποιήστε το πρόγραμμα της προηγούμενης άσκησης ώστε να υπάρχει ένας κοινόχρηστος πίνακας από ακεραίους (ο αριθμός των στοιχείων του πίνακα ισούται με τον αριθμό των νημάτων) και το κάθε νήμα να είναι υπεύθυνο να αυξάνει ένα μόνο στοιχείο του πίνακα. Πιο συγκεκριμένα, τα στοιχεία του κοινόχρηστου πίνακα αρχικοποιούνται στη τιμή 0 από την main. Το κάθε νήμα υλοποιεί ένα for loop με σταθερό αριθμό επαναλήψεων, και σε κάθε επανάληψη αυξάνει κατά ένα το στοιχείο του κοινόχρηστου πίνακα που του αντιστοιχεί. Η main τυπώνει τις τελικές τιμές των στοιχείων του πίνακα οι οποίες θα πρέπει να είναι ντετερμινιστικές, καθώς και το άθροισμά τους.

Χρειάζεται να χρησιμοποιήσετε συγχρονισμό και γιατί; Φροντίστε να κρατήσετε σταθερό τον συνολικό αριθμό των ενημερώσεων στον κοινόχρηστο πίνακα καθώς αυξάνεται ο αριθμός των νημάτων, και συγκρίνετε την επίδοση του προγράμματός σας καθώς αυξάνεται ο αριθμός των νημάτων. Τι παρατηρείτε όσον αφορά τη βελτίωση της επίδοσης; Μπορείτε να τροποποιήσετε το πρόγραμμα σας ώστε να βελτιωθεί η επίδοση του προγράμματός σας;

Άσκηση 1.4

Σε αυτή την άσκηση καλείστε να υλοποιήσετε τα δικά σας κλειδώματα ανάγνωσης-εγγραφής. Για την υλοποίησή σας θα χρησιμοποιήσετε μια δομή δεδομένων που χρησιμοποιεί δύο μεταβλητές συνθήκης – μία για νήματα «ανάγνωσης» και μία για νήματα «εγγραφής» – και ένα mutex. Η δομή περιέχει επίσης μέλη που δείχνουν:

1. πόσα νήματα ανάγνωσης έχουν προσλάβει το κλείδωμα, δηλαδή, διαβάζουν δεδομένα,
2. πόσα νήματα ανάγνωσης περιμένουν για το κλείδωμα,
3. αν κάποιο νήμα εγγραφής έχει προσλάβει το κλείδωμα, και
4. πόσα νήματα εγγραφής περιμένουν για το κλείδωμα.

Το mutex προστατεύει τη δομή δεδομένων του κλειδώματος ανάγνωσης-εγγραφής: όποτε ένα νήμα καλεί μια από τις συναρτήσεις (κλειδώματος ανάγνωσης, κλειδώματος εγγραφής, ξεκλειδώματος) κλειδώνει πρώτα το mutex, και όποτε ένα νήμα ολοκληρώνει μια από αυτές τις κλήσεις ξεκλειδώνει το mutex. Αφού προσλάβει το mutex, το νήμα ελέγχει τα κατάλληλα μέλη δεδομένων για να προσδιορίσει πώς να προχωρήσει. Για παράδειγμα, αν χρειάζεται πρόσβαση για ανάγνωση, ελέγχει αν υπάρχει νήμα εγγραφής που να έχει προσλάβει το κλείδωμα. Αν δεν υπάρχει, αυξάνει κατά ένα το πλήθος των ενεργών νημάτων ανάγνωσης και προχωράει. Αν είναι ενεργό κάποιο νήμα εγγραφής, το τρέχον νήμα αυξάνει κατά ένα το πλήθος των νημάτων ανάγνωσης σε αναμονή και περνάει σε κατάσταση αναμονής για τη μεταβλητή συνθήκης ανάγνωσης. Όταν το νήμα «αφυπνιστεί», ελαττώνει κατά ένα το πλήθος των νημάτων ανάγνωσης σε αναμονή, αυξάνει κατά ένα το πλήθος των ενεργών νημάτων ανάγνωσης, και προχωράει. Η λειτουργία του κλειδώματος εγγραφής υλοποιείται με παρόμοιο τρόπο.

Οι ενέργειες που πραγματοποιούνται στη συνάρτηση ξεκλειδώματος εξαρτώνται από το αν το νήμα ήταν νήμα ανάγνωσης ή νήμα εγγραφής. Αν ήταν νήμα ανάγνωσης, δεν υπάρχουν άλλα ενεργά νήματα ανάγνωσης και υπάρχει νήμα εγγραφής σε αναμονή, τότε το τρέχον νήμα μπορεί, πριν επιστρέψει, να ειδοποιήσει ένα νήμα εγγραφής για να προχωρήσει. Αν, από την άλλη, ήταν νήμα εγγραφής, τότε μπορεί να βρίσκονται σε αναμονή τόσο νήματα ανάγνωσης όσο και νήματα εγγραφής, και το τρέχον νήμα πρέπει να αποφασίσει ποιον τύπο νημάτων θα προτιμήσει.

Καλείστε να υλοποιήσετε δύο προσεγγίσεις: (α) στην πρώτη προσέγγιση θα δίνεται προτεραιότητα στα νήματα ανάγνωσης, ενω (β) στην δεύτερη προσέγγιση θα δίνεται προτεραιότητα στη νήματα εγγραφής. Σας δίνεται το πρόγραμμα συνδεδεμένης λίστας του βιβλίου που χρησιμοποιεί κλειδώματα ανάγνωσης-εγγραφής της Pthreads (pth_ll_rwl.c). Καλείστε να τροποποιήστε αυτό το πρόγραμμα ώστε να χρησιμοποιεί τα δικά σας κλειδώματα, και να συγκρίνετε την απόδοση του προγράμματος όταν δίνεται προτεραιότητα στα νήματα ανάγνωσης με την απόδοσή του όταν δίνεται προτεραιότητα στα νήματα εγγραφής για διαφορετικές τιμές νημάτων και διάφορα ποσοστά λειτουργιών (για παράδειγμα, αρχικά 1000 κλειδιά, 500.000 λειτουργίες, 99.9% / 95% / 90% λειτουργίες Member()). Ίσως σας φανεί χρήσιμο να τροποποιήσετε το πρόγραμμα ώστε να λαμβάνει σαν ορίσματα όλες τις σχετικές επιλογές εισόδου.

Άσκηση 1.5 (προαιρετική)
Σε αυτή την άσκηση καλείστε να: (α) πειραματιστείτε με την υλοποίηση φράγματος (barrier) που παρέχει η βιβλιοθήκη Pthreads (pthread_barrier), (β) υλοποιήσετε φράγμα με τη χρήση κλειδώματος και μεταβλητής συνθήκης, (γ) 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, σας δίνεται 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), 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?