Σειρά Εργασιών 1

1.1 FIFO pipe

Υλοποιήστε έναν αγωγό FIFO μιας κατεύθυνσης για την επικοινωνία ανάμεσα σε δύο νήματα, ως ένα ανεξάρτητο τμήμα λογισμικού με τις λειτουργίες void pipe_init(int size) για την αρχικοποίηση του αγωγού, void pipe_write(char c) για το γράψιμο ενός byte, void pipe_writeDone() για τη σηματοδότηση ότι δεν θα γραφτούν άλλα δεδομένα, int pipe_read(char *c) για την ανάγνωση ενός byte, και void pipe_destroy() για την καταστροφή του αγωγού. Η pipe_init και η pipe_destroy καλείται προτού αρχίσει και αντίστοιχα αφού ολοκληρωθεί η εγγραφή/ανάγνωση δεδομένων σε/από τον αγωγό. Αν ο αγωγός είναι γεμάτος, η pipe_write πρέπει να «μπλοκάρει» μέχρι να δημιουργηθεί χώρος. Αν ο αγωγός είναι άδειος, η pipe_read πρέπει να «μπλοκάρει» μέχρι να υπάρξουν δεδομένα. Αν διαβαστούν επιτυχώς δεδομένα, η pipe_read επιστρέφει 1. Αν ο αγωγός είναι άδειος και έχει κληθεί η pipe_writeDone τότε η pipe_read πρέπει να επιστρέφει άμεσα 0.

Βασιστείτε στην τεχνική της κυκλικής αποθήκης (circular/ring buffer). Σκεφτείτε κατά πόσο μπορεί να προκύψουν ανεπιθύμητες συνθήκες ανταγωνισμού για ταυτόχρονη εκτέλεση των pipe_write/writeDone και της pipe_read. Ελέγξτε την υλοποίηση σας μέσω ενός προγράμματος όπου δύο νήματα επικοινωνούν μεταξύ τους μέσω ενός αγωγού μεγέθους 64 bytes, για τη μεταφορά (μεγάλων) αρχείων.

1.2 Αναγνώριση πρώτων αριθμών

Υλοποιήστε την συνάρτηση int isPrime(int val) που επιστρέφει 1 αν το val είναι πρώτος αριθμός, διαφορετικά επιστρέφει 0. Στη συνέχεια υλοποιήστε ένα πρόγραμμα που διαβάζει επαναληπτικά από την είσοδό του ακέραιες τιμές, για κάθε μια από τις οποίες καλεί την isPrime() και εκτυπώνει το αποτέλεσμα. Για να επιταχυνθεί ο υπολογισμός, το πρόγραμμα πρέπει να χρησιμοποιεί νήματα στο πνεύμα του παρακάτω ψευτοκώδικα:

```
main thread:

create workers
while (job exists) {
  wait for a worker to become available
  assign next job to an available worker
  notify the worker to process the job
}
wait for all workers to become available
notify workers to terminate
wait for all workers to terminate
```

```
worker thread:


while (1) {
  notify main that I am available
  wait for notification by main
  if notified to terminate, break
  else process assigned job
}
notify main that I will terminate
```

Προσπαθήστε να αναλύσετε την απόδοση του προγράμματος ως συνάρτηση του αριθμού των νημάτων που χρησιμοποιεί (όρισμα του προγράμματος).

1.3 Quicksort

Υλοποιήστε μια παράλληλη αναδρομική έκδοση του quicksort όπου η ταξινόμηση των στοιχείων του πίνακα γίνεται από πολλά νήματα. Σε κάθε επίπεδο αναδρομής, ένα ξεχωριστό νήμα (αρχικά το κυρίως νήμα) ελέγχει το τμήμα του πίνακα που του έχει ανατεθεί (για το κυρίως νήμα, ολόκληρος ο πίνακας). Αν αυτό έχει μήκος < 2 τότε το νήμα δεν κάνει τίποτα και τερματίζει, διαφορετικά πραγματοποιεί το βήμα του «διαχωρισμού» για το κομμάτι του πίνακα που του έχει ανατεθεί και αναθέτει αναδρομικά την ταξινόμηση των δύο υπο-τμημάτων σε δύο νέα νήματα που δημιουργεί για αυτό τον σκοπό, περιμένει να τερματίσουν και μετά επιστρέφει το ίδιο. Τα επιπλέον νήματα και οι μεταβλητές που χρησιμοποιούνται για τον συγχρονισμού τους, πρέπει να δημιουργούνται και να καταστρέφονται δυναμικά κατά την αναδρομή.

Δοκιμάστε την υλοποίηση σας μέσω ενός προγράμματος που διαβάζει από την είσοδο του μια ακολουθία ακεραίων που αποθηκεύει σε έναν καθολικό πίνακα, στην συνέχεια ταξινομεί τον πίνακα χρησιμοποιώντας την παράλληλη έκδοση του quicksort, και τέλος εκτυπώνει το αποτέλεσμα.

Η υλοποίηση πρέπει να γίνει σε C με χρήση της βιβλιοθήκης pthreads. Ο συγχρονισμός μεταξύ των νημάτων πρέπει να υλοποιηθεί με **απλές κοινές μεταβλητές** χωρίς την χρήση κάποιου μηχανισμού συγχρονισμού των pthreads ή του λειτουργικού (μπορείτε να χρησιμοποιήσετε την λειτουργία yield).

Παράδοση: Σάββατο 31 Οκτωβρίου 2020, 23:59