Παράλληλος Προγραμματισμός 2018

Προγραμματιστική Εργασία #2

Ονοματεπώνυμο: Ηλιάδης Νικόλαος

ΑΜ: Π2014009

Δεν μπόρεσα να υλοποιήσω τον κώδικα με βάση τα ζητούμενα της εργασίας καθώς συνάντησα κάποια προβλήματα στην μεταφορά στον circular buffer. Έπειτα από κάποιες αλλαγές στο παράδειγμα που μας δώσατε καθώς και στο παράδειγμα της quicksort κατέστη δυνατή η εκτέλεση του κώδικα αλλά υπάρχει ένα πρόβλημα στα στοιχεία που ταξινομεί ο αλγόριθμος. Ειδικότερα το σορτάρισμα του κάθε μηνύματος από το γίνεται από το επόμενο σε σειρά thread. Το σορτάρισμα ξεκινάει με ολόκληρο το array και κάθε εκτέλεση ο πίνακας σπάει σε μικρότερα κομμάτια τα οποία παραλαμβάνονται από το επόμενο σε σειρά thread. Τα πρώτα 3 στοιχεία του random πίνακα δεν σουτάρονται σωστά και έχουμε error. Τα στοιχεία αυτά τα παίρνει ο producer (παράδειγμα που μας δίνετε) και παίρνει το επόμενο κομμάτι του πίνακα από το thread consumer και το προσθέτει στην ουρά. Έπειτα τσεκάρουμε αν η ουρά είναι γεμάτη η άδεια και προσθέτει αν ισχύει το πρώτο κριτήριο. Για να μην έχουμε κάποιο deadend έχουμε το mutex όπου μας ενημερώνει με μήνυμα λάθους.

Αρχικά έχουμε:

- μια struct που έχει μέσα τα στοιχεία του πίνακα.
- typedef struct όπου γίνονται define οι μεταβλητές “message”
- έναν κυκλικό buffer.
- την pthread_cond_t msg_in = PTHREAD_COND_INITIALIZER όπου ορίζεται η εντολή για το πότε μπορεί να διαβάσει από τον buffer.
- pthread_cond_t msg_out = PTHREAD_COND_INITIALIZER όπου ορίζεται η εντολή για το πότε μπορεί να γράψει στον buffer.
- pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER όπου ορίζεται η εντολή για το ποιο κομμάτι του κώδικα που τρέχει έχει πρόσβαση στον κώδικα.
- Ο λόγος που χρησιμοποιούμε inssort είναι για το ότι το όριο (cutoff) είναι μεγαλύτερο από το μέγεθος του πινακα.
- (παράδειγμα σειριακής υλοποίησης (append κομματιών πινακα σε ένα).
- στην pthread_create έχουμε δημιουργία thread (το πρώτο όρισμα είναι το που θα αποθηκευτεί το id του καινούριο thread), το producer_thread είναι η main function για το thread (ξεκινάει από εδώ).