Παράλληλος Προγραμματισμός
Εργασία 2 Εαρινού Εξαμήνου

Υπεύθυνος Καθηγητής: δρ. Μιχαήλ Στεφανιδάκης
Φοιτητής: Κωνσταντίνος Μπέζας Π2013026

Ιόνιο Πανεπιστήμιο
Τμήμα Πληροφορικής

Σκοπός της εργασίας

Διάταξη ενός πίνακα τυχαίων double δεδομένων κατά αύξουσα σειρά, με τη χρήση πολλαπλών thread και μίας ουράς εκτέλεσης.

Περιγραφή Κώδικα που Αναπτύχθηκε

Αντικείμενα που χρησιμοποιήθηκαν:

- Enum messageType {Διατηρεί τους διαφορετικούς τύπους μηνυμάτων που αποστέλλονται στην ουρά - WORK, FINISH, SHUTDOWN}
- Struct message{Περιέχει δεδομένα που αφορούν τον τύπο του μηνύματος και ένα εύρος από τον πίνακα με τα δεδομένα (αρχή και τέλος)}

Εντός της main οι πρώτες ενέργειες, αφορούν την αρχικοποίηση των πινάκων που διατηρούν τα δεδομένα προς διάταξη και τα threads. Έπειτα, ο πρώτος πίνακας δέχεται τυχαίες τιμές για κάθε στοιχείο του από 0 έως 1 και η main αποστέλλει την πρώτη δουλειά στην ουρά εκτέλεσης. Αυτή περιέχει το εύρος ολόκληρου του πινάκα. Αμέσως μετά γίνεται η ενεργοποίηση των threads όπου τους ανατίθεται η threadFunc. Σε αυτήν τη συνάρτηση το κάθε thread αρχικά διαβάζει ένα μήνυμα από την ουρά και ανάλογα με τον τύπο του προχωράει σε διαφορετικές ενέργειες. Ακολούθως, η main εκκινεί ενέργειες που αφορούν την ανίχνευση της ολοκλήρωσης της διαδικασίας της διάταξης. Πιο συγκεκριμένα, διατηρεί σε μια μεταβλητή (completed) τον αριθμό των στοιχείων που έχουν διαταχθεί και όσο αυτός ο αριθμός είναι μικρότερος από τον αριθμό των στοιχείων του πίνακα, διαβάζει μηνύματα από την ουρά. Για την αποφυγή απώλειας κάποιας εργασίας, WORK message type, οποιοδήποτε μήνυμα φέρει τον ανωτέρω τύπο επαναπροωθείται στην ουρά εκτέλεσης. Τα μηνύματα που περιέχουν την απαραίτητη πληροφορία είναι τύπου FINISH. Η στιγμή που όλα τα στοιχεία έχουν διαταχθεί όπως πρέπει, είναι και τερματική για την εκτέλεση των threads, δηλαδή η main αποστέλλει ένα μήνυμα τύπου SHUTDOWN στην ουρά δίνοντας στα threads εντολή τερματισμού. Αμέσως μετά γίνεται ένας έλεγχος στη διάταξη των στοιχείων του πίνακα, πράγμα που πιστοποιεί την σωστή εκτέλεση των διαδικασιών του προγράμματος, και εκτυπώνεται το ανάλογο μήνυμα. Τέλος, καταστρέφονται τα conditional variables και το mutex και αποδεσμεύεται η μνήμη του πίνακα.

Τα threads κατά την εκκίνηση τους εισέρχονται σε έναν ατέρμονα βρόγχο στον οποίο διαβάζουν συνεχεία νέα μηνύματα από την ουρά. Σύμφωνα με τον τύπο του μηνύματος εκτελούνται οι ακόλουθες ενέργειες. Για μηνύματα τύπου WORK, ελέγχεται, αρχικά, εάν το εύρος που ελήφθη είναι μικρότερο ή ίσο από ένα όριο (THRESHOLD = 10) και αν αυτό ισχύει, καλείται η insort() η οποία αναλαμβάνει να διατάξει τα στοιχεία κατά αύξουσα σειρά. Κατά την ολοκλήρωση, αποστέλλει το thread στην ουρά εκτέλεσης ένα νέο μήνυμα τύπου FINISH που περιλαμβάνει το εύρος του πίνακα που διετάχθη. Εάν το εύρος, που υπάρχει στο νέο ληφθέν μήνυμα, είναι μεγαλύτερο από το THRESHOLD τότε χωρίζεται ο πίνακας στην μέση με βάση το pivot στοιχείο του που επιστρέφει η εκτέλεση της partition() και αποστέλλονται στην ουρά δύο νέα μηνύματα τύπου WORK που περιέχουν τα δύο μισά του προηγούμενου εύρους.

Για μηνύματα τύπου SHUTDOWN επανατοποθετείται το ίδιο μήνυμα στην ουρά και σταματά το thread την εκτέλεση που έκανε έως εκείνη τη στιγμή.

Τέλος, εάν λάβει μήνυμα τύπου FINISH το τοποθετεί εκ νέου στην ουρά.

Για τις ανάγκες της υλοποίησης δημιουργήθηκε η συνάρτηση void swap(double *one, double *two) η οποία εναλλάσσει απλώς τις θέσεις μνήμης στις οποίες δείχνουν οι pointers που λαμβάνει ως είσοδο.

Για την ομαλή διεξαγωγή των διεργασιών του προγράμματος έχουν χρησιμοποιηθεί mutex και conditional variables. Η χρήση και των δύο γίνεται από τις συναρτήσεις send() και receive(). Οι τελευταίες διαχειρίζονται την είσοδο και την έξοδο μηνυμάτων από την ουρά. Η χρήση του mutex διασφαλίζει ότι μόνο ένα thread θα προσθέτει ή θα αφαιρεί μηνύματα από την ουρά, διαφορετικά θα προκύψουν τα εξής προβλήματα:

1. Το πρώτο εξ αυτών είναι ότι με μεγάλη πιθανότητα δύο ή παραπάνω threads θα προσπαθούσαν να προσπελάσουν την ίδια θέση μνήμης όπου διατηρείται ο αριθμός των μηνυμάτων που υπάρχουν στην ουρά.
2. Εάν δύο ή παραπάνω threads εκτελούσαν τη συνάρτηση receive χωρίς τη χρήση mutex θα ήταν πολύ πιθανό να λάβουν και να επεξεργαστούν το ίδιο μήνυμα.

Η χρήση των conditional variables γίνεται για διαφορετικούς λόγους στις δύο συναρτήσεις send() και receive(). Στην πρώτη περίπτωση, πρέπει να γίνεται έλεγχος εάν ο αριθμός των μηνυμάτων στην ουρά είναι μεγαλύτερος ή ίσος από το μέγεθος της για να αποφευχθεί η διαγραφή ενός υπάρχοντος μηνύματος μιας και η ουρά είναι ένας πινάκας wrap around. Σε αυτήν την περίπτωση το conditional variable msg_out περιμένει να λάβει "ειδοποίηση" προτού προσχωρήσει στην καταχώρηση νέου μηνύματος. Η ειδοποίηση αποστέλλεται αφού έχει αφαιρεθεί από την ουρά ένα μήνυμα μέσω της receive(). Στην δεύτερη περίπτωση, δηλαδή σε αυτή της receive(), υπάρχει η ανάγκη να βεβαιωθεί το thread ότι έστω και ένα μήνυμα έχει προστεθεί στην ουρά προτού προσπαθήσει να το αφαιρέσει. Αυτό επιτυγχάνεται, αρχικά, με τον έλεγχο του αριθμού των μηνυμάτων που υπάρχουν στην ουρά και έπειτα με την παύση της εκτέλεσης του κωδικά έως ότου το msg_in λάβει σήμα να προχωρήσει κάτι που πραγματοποιείται από την send() μόλις προσθέσει ένα μήνυμα στην ουρά.