Λειτουργικά Συστήματα

2η Προγραμματιστική Εργασία, Αναφορά

Εαρινό Εξάμηνο 2018-2019

Στριμμένος Γιάννης (p3140197) | Μπιτσάκος Τρύφωνας (p3160108) |
Καραστατήρης Παναγιώτης (p3150064)

Ο κώδικας ξεκινά από την **main** όπου και αφού τυπώσει τον αριθμό των θέσεων του θεάτρου, δίνει τον έλεγχο στην **arguments_check(argc, argv)** που ελέγχει αν έχουν δοθεί οι δύο απαραίτητες παράμετροι κατά την εκτέλεση. Εκεί αρχικοποιούμε τις μεταβλητές *n_cust* και *seed* που είναι ο πίνακας όλων των πελατών κι ο σπόρος για την γεννήτρια ψευδό-τυχαίων αριθμών. Αν υπάρχει σφάλμα με τις παραμέτρους, τυπώνεται μήνυμα σφάλματος και τερματίζεται το πρόγραμμα. Επιστρέφοντας στην main, αρχικοποιούμε την μνήμη που αφορά τα νήματά μας, δηλαδή τους πελάτες. Η *rc* είναι βοηθητική και κρατά κάποιες τιμές επιστροφής. Η *clients* είναι ο πίνακας που περιέχει τα νήματα. Αν δεν υπάρχει αρκετός χώρος στη μνήμη για όλα τα νήματα, μετά από μήνυμα σφάλματος, τερματίζεται το πρόγραμμα. Ο πίνακας *clientCount* έχει θέσεις όσες κι ο αριθμός των νημάτων και χρησιμοποιείται για την αντιστοίχιση κάθε νήματος με έναν ακέραιο, περιγράφει δηλαδή, τον αριθμό συναλλαγής. Στη συνέχεια, αρχικοποιούμε όλα τα mutexes και τις συνθήκες με τη βοήθεια κατάλληλων συναρτήσεων. Η **mutex_handle** δέχεται έναν δείκτη σε mutex και μια ακέραιη σημαία. Οι σημαίες είναι ορισμένες στο header file. Ανάλογα με την σημαία, κάνει init, destroy, lock και unlock τον mutex και επιστρέφει τον κωδικό επιστροφής της βιβλιοθήκης pthread. Αν υπάρξει πρόβλημα σε κάποια διαδικασία, ενημερώνει με κατάλληλο μήνυμα και τερματίζει. Αντίστοιχα λειτουργεί κι η **condition_handle** για τις συνθήκες. Πλέον, έχουμε περάσει στον βρόχο των νημάτων. Εδώ δημιουργούνται όλα τα νήματα και ενημερώνονται ότι πρέπει να ξεκινήσουν από την transaction. Αν υπάρξει πρόβλημα με τη δημιουργία κάποιου νήματος, τυπώνεται μήνυμα και τερματίζει το πρόγραμμα. Για να γίνουν οι κλήσεις με γραμμική σειρά, μπορούμε να χρησιμοποιήσουμε την sleep(1); που είναι σε σχόλια.

Η **transaction** δέχεται έναν ακέραιο, τον κωδικό του πελάτη, και τον αντιστοιχεί σε μια νέα συναλλαγή, *tid*. Κατόπιν, ορίζει ένα στιγμιότυπο του struct **TRANSACTION_INFO** που περιέχει όλα τα στοιχεία της συναλλαγής. Κρατά τον αριθμό συναλλαγής (transaction_no), έναν πίνακα με τις θέσεις που αφορά η συναλλαγή (seats), τον αριθμό θέσεων που θέλει ο πελάτης (requested_seats), την ζώνη την οποία επιθυμεί ο πελάτης (requested_zone) και το κόστος της συναλλαγής (cost). Το struct είναι δηλωμένο στο header file. Η transaction δηλώνει και κάποιες απαραίτητες τοπικές μεταβλητές του struct timespec που χρησιμοποιούνται στον υπολογισμό των χρόνων αναμονής. Αφού δημιουργηθεί το νήμα και μπει στην transaction, περιμένει έναν τηλεφωνητή και κρατά τον χρόνο αναμονής. Αν υπάρχουν ακόμα διαθέσιμες θέσεις στο θέατρο, καλείται η **handle_seats**. Η μέθοδος αυτή, υπολογίζει τυχαία με τη βοήθεια των **request_seats** και ***request_zone*** των αριθμό θέσεων και τη ζώνη που ζητά ο πελάτης και τ=ενημερώνει το TRANSACTION_INFO. Στη συνέχει καλεί την **find_seats** για να βρει τις διαθέσιμες θέσεις. Η find_seats προσομοιώνει τον χρόνο αναζήτησης και στην συνέχεια δίνει τιμές στις *start, end* που δείχνουν από ποια μέχρι ποια θέση του θεάτρου ενδιαφέρει τον πελάτη, βάση της ζώνης. Ο βρόχος στη συνέχεια ξεκινά από την αρχή της ζώνης και κοιτά προς τα δεξιά της κάθε θέσης αν υπάρχουν οι ζητούμενες ελεύθερες θέσεις. Η συνθήκη του βρόχου ορίζει να κοιτάξει μέχρι το σημείο που έχουν μείνει τουλάχιστον όσες θέσεις ψάχνουμε προς τα δεξιά. Με αυτό τον τρόπο, δεν βγαίνουμε εκτός της ζητούμενης ζώνης. Ενώ η μεταβλητή *i* αφορά την τρέχουσα θέση-αφετηρία αναζήτησης στον πίνακα, η μεταβλητή *i_in* είναι αυτή που κοιτά τις θέσεις στα δεξιά της i. Αν η θέση-αφετηρία δεν είναι ελεύθερη, ψάχνει την αμέσως επόμενη ελεύθερη θέση και προχωράει συνεχίζει την αναζήτηση από εκεί. Αν είναι σε διαθέσιμη θέση, την μαρκάρει ως OCCUPIED και την τοποθετεί και στον πίνακα seats του TRANSACTION_INFO. Αν δεν συμπληρωθεί ο αριθμός των ζητούμενων θέσεων, ο πίνακας seats αδειάζει και μαρκάρει τις θέσεις ως EMPTY. Μόλις ολοκληρωθεί η αναζήτηση, επιστρέφει κατάλληλο κωδικό επιτυχίας ή

αποτυχίας στην handle_seats. Αν βρέθηκαν οι θέσεις, τυπώνει την απαραίτητη έξοδο προς τον πελάτη και επιστρέφει στην transaction με κωδικό επιτυχίας. Αλλιώς, τυπώνει ότι δεν βρήκε θέσεις και επιστρέφει με κωδικό αποτυχίας. Σε αυτό το σημείο, η transaction, καλεί την handle_operator που ξεκλειδώνει το mutex του τηλεφωνητή, αυξάνει τον αριθμό των διαθέσιμων τηλεφωνητών και ενημερώνει με το condition. Εφόσον είχαν βρεθεί θέσεις, περνάμε στην **cashier_transaction**. Εδώ και πάλι κρατάμε τον χρόνο αναμονής του πελάτη. Στη συνέχεια υπολογίζουμε το κόστος της συναλλαγής ανάλογα με την ζώνη κράτησης. Καλείται η **pay_seats**, που με πιθανότητα 0.9 δέχεται την πληρωμή. Αν η πληρωμή πέτυχε, μαρκάρουμε στο πλάνο των θέσεων τις θέσεις του πελάτη με τον αριθμό συναλλαγής του και τυπώνουμε τα ζητούμενα μηνύματα εξόδου. Αν δεν πέτυχε η πληρωμή, οι θέσεις μαρκάρονται εκ νέου ως EMPTY και ενημερώνεται ο πελάτης. Όμοια με τον τηλεφωνητή παραπάνω, κάνουμε unlock και τον ταμία. Επιστρέφοντας πάλι στην transaction, ανανεώνουμε τα στατιστικά και το νήμα κάνει έξοδο. Το νήμα ολοκληρώνει τη ζωή του στην main στο βρόχο των join. Αφού ολοκληρώσουν όλα τα νήματα, υπολογίζει τα στατιστικά, τυπώνει την έξοδο και το τελικό πλάνο των θέσεων. Τέλος κλείνει τα mutexes, τα conditions και ελευθερώνει την δεσμευμένη μνήμη.