ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Προγραμματιστική Εργασία Εαρινό 2022

Ομάδα : 3200212-3200219

(ΓΕΝΙΚΑ) Για την υλοποίηση του project κάναμε χρήση μιας δομής struct , η οποία περιλαμβάνει την δήλωση ενός δυναμικού πίνακα για *τα id των πελατών* καθώς και το *μέγεθος του πίνακα* μας. Η αρχικοποίηση τους πραγματοποιείται εντός της main , καθώς τότε γίνεται γνωστός ο αριθμός των πελατών. Ακόμα, για το πέρασμα των μεταβλητών στην βασική μας συνάρτηση booking χρησιμοποιήσαμε καθολικές μεταβλητές , οι οποίες δηλώνονται (μερικές αρχικοποιούνται κιόλας) μαζί με τα *mutex* και τα *condition* πριν την υλοποίηση της main.Δηλώσαμε 7 mutex , ένα για τον τραπεζικό λογαριασμό του θεάτρου , ένα για τον αριθμό των διαθέσιμων τηλεφωνητών, ένα για τον αριθμό των διαθέσιμων ταμίων , για το πλήθος των πετυχημένων κρατήσεων , για το πλήθος των αποτυχημένων κρατήσεων λόγω περιορισμένων θέσεων, για το πλήθος των αποτυχημένων κρατήσεων λόγω αδυναμίας πληρωμής με κάρτα. Προσθέσαμε και ένα επιπλέον mutex για τον πίνακα των θέσεων του θεάτρου. Τέλος, τα 2 condition που δηλώσαμε τα χρειαστήκαμε στην περίπτωση που υπήρχε κάποιος πελάτης αλλά δεν υπήρχαν διαθέσιμοι πόροι για να τους εξυπηρετήσουν.

(MAIN) Όσον αφορά την *main* συνάρτηση , αρχικά ελέγχουμε αν ο χρήστης μας δίνει το σωστό πλήθος παραμέτρων. Χρειαζόμαστε 2 παράμετρους , ένα για τον αριθμό των πελατών και ένα για τον σπόρο. Γίνεται ο έλεγχος **argc!=3** καθώς το εκτελέσιμο αρχείο a.out προσμετράται στις παραμέτρους που γράφονται στο *command line*. Στην συνέχεια , αρχικοποιούμε κατάλληλα τις μεταβλητές *Ncust* , *seed* , τον *δυναμικό μας πίνακα* και φτιάχνουμε έναν πίνακα για τα threads μεγέθους Ncust. Αρχικοποιούμε όλα μας τα mutex και τα condition με την τιμή NULL , πραγματοποιώντας παράλληλα τους κατάλληλους ελέγχους για την σωστή αρχικοποίηση τους. Σκέφτήκαμε να δηλώσουμε στον πίνακα που αποικονίζει το πλάνο του θεάτρου , τις κένες θέσεις θέτοντας μηδέν και μόλις η θέση δοθεί σε κάποιον πελάτη θα τοποθετηθεί το id του.Στην συνέχεια , δημιουργήσαμε τα threads τοποθετώντας την συνάρτηση sleep μετά την δημιουργία του κάθε ένα νήματος , έτσι ώστε το *πρώτο* να φτιάχνεται την *χρονική στιγμή t0=0* και όλα τα *υπόλοιπα μετά από τυχαίο χρόνο*. Με τον τερματισμό των threads εμφανίζουμε το πλάνο του θεάτρου μετά την ολοκλήρωση των κρατήσεων , καθώς και τον τραπεζικό

λογαριασμό (*account*) , τα ζητούμενα ποσοστά επιτυχίας/αποτυχίας χρησιμοποιώντας τον τύπο **(double)ci/(double)Ncust*100** όπου *ci με i=1,2,3* αντιστοιχούν στις μεταβλητές που αφορούν το πλήθος των επιτυχημένων κρατήσεων , των αποτυχημένων λόγω θέσεων και των αποτυχημένων λόγω πληρωμής . Εν τέλει , καταστρέφουμε τα mutex και τα condition που χρησιμοποιήσαμε και ελευθερώνουμε την μνήμη που δεσμέυσαμε για τον πινακα των id πελάτη.

(BOOKING) Σε αυτή την παράγραφο θα περιγράψουμε την συνάρτηση *booking*. Τοποθετούμε την μεταβλητή που περάσαμε στην συνάρτηση και αφορά το id του νήματος που κάλεσε την συνάρτηση σε μια τοπική μεταβλήτη,την *idd.* Κλειδώνουμε την *μεταβλητή tel* για να μην μπορούν να την επεξεργαστούν πολλά νήματα ταυτόχρονα και γίνεται ο έλεγχος αν υπάρχουν διαθέσιμοι τηλεφωνητές. Αν δεν υπάρχουν , το νήμα περιμένει. Διαφορετικά , μειώνουμε τον αριθμό των διαθέσιμων πόρων(ενν. τηλεφωνητών) , ξεκλειδώνουμε το mutex και προχωράμε την εύρεση των θέσεων. Πρώτα, βρίσκουμε την ζώνη στην οποία θα καθίσει ο πελάτης με χρήση της **rand_r(&seed)%(10-1+1)+1**. Αν επιστραφεί *αριθμός* <= 3 τότε έχει επιλεγχεί η *ζώνη Α* , αλλιώς *η ζώνη Β.* Επίσης δίνουμε την τιμή 0 σε μία μεταβλητή x αν επιλεχθεί η ζώνη Α και την τιμή 1 αν επιλεχθεί η ζώνη Β. Αυτό θα μας χρειαστεί στη συνέχεια καθώς θα διατρέχουμε τον πίνακα των θέσεων. Στο πρόγραμμά μας για την ύψωση σε δύναμη χρησιμοποιούμε τη δική μας συνάρτηση myPow η οποία υλοποιεί την πράξη της ύψωσης σε δύναμη. Γίνεται η τυχαία επιλογή του αριθμού των ζητούμενων θέσεων **(rand_r(&seed)%(Nseathigh-Nseatlow+1)+Nseatlow)** και ο πελάτης περιμένει **(sleep(rand_r(&seed)%(tseathigh-tseatlow+1)+tseatlow))** μέχρι να βρεθούν οι θέσεις.Κλειδώνουμε τον πίνακα των θέσεων. Κανονικά θα διατρέχαμε τον πίνακα είτε από 0 έως και 9 ή 10 έως και 29 αλλά για να «γλυτώσουμε» την μία επανάληψη χρησιμοποιήσαμε την εξής σχέση: x*10<(3^x)*10 (^ ύψωση σε δύναμη). Για παράδειγμα αν βρισκόμαστε στη Ζώνη Α ,όπου επιθυμούμε να τρέξει από 0 έως και 9, το x θα έχει την τιμή 0 οπότε το i ισούται με x*10 όπου στην προκειμένη περίπτωση είναι 0 και το 3^0* 10 = 10. Οπότε ικανοποιεί τη σχέση. Αντίστοιχα για τη Ζώνη Β το x = 1, x* 10 =10 και 3^x*10 = 30. Ο τηλεφωνητής , τώρα, ξεκινάει να ψάχνει ανά σειρά. Όσο δεν έχει βρει τον απαιτούμενο αριθμό θέσεων και οι διαδοχικές θέσεις έχουν την τιμή 0 , αυξάνει τον *μετρητή counter*. Αν αλλάξει σειρά , δηλαδή το j ή βρεθεί μη διαθέσιμη θέση , μηδενίζουμε τον μετρητή μας. Αν δεν συμβεί κάποιο από τα προηγούμενα και βρούμε τον απαιτούμενο αριθμό,αντικαθιστούμε το 0 με το id του πελάτη σε αυτές τις θέσεις , υπολογίζουμε το κόστος και σταματάμε την σάρωση κάνοντας break στα 2 while-loop μας. Ξεκλειδώνουμε τον πίνακα , κλειδώνουμε την μεταβλητή για τους διαθέσιμους τηλεφωνητές , την αυξάνουμε κατά 1 από την στιγμή που έχει ολοκληρωθεί η διαδικασία

εύρεσης θέσεων και τώρα το νήμα-πελάτης περνάει στον ταμία και ξεκλειδώνουμε την μεταβλητή μας. Αντίθετα, αν δεν βρεθούν οι ζητούμενες θέσεις κλειδώνουμε την *μεταβλητή c2 ,* την αυξάνουμε κατά 1 , την ξεκλειδώνουμε και τερματίζουμε το νήμα κάνοντας **pthread_exit(NULL)**. Ο πελάτης περνάει στο στάδιο της πληρωμής , στον ταμία. Ακολουθούμε την ίδια διαδικασία με τους τηλεφωνητές , κλειδώνουμε την μεταβλητή *cash* για να μην μπορούν να την επεξεργαστούν πολλά νήματα ταυτόχρονα και γίνεται ο έλεγχος αν υπάρχουν διαθέσιμοι ταμίες. Αν δεν υπάρχουν , το νήμα περιμένει. Αν υπάρχουν ,το νήμα περιμένει μέχρι ο ταμίας να ολοκληρώσει την διαδικασία .Υπολογίζουμε την πιθανότητα μιας επιτυχούς πληρωμής. Αν επιστραφεί **(rand_r(&seed)%(10-1+1)+1)** *αριθμός<9 τότε* η πληρωμή αποτυγχάνει , απελεύθερωνουμε τον ταμία αυξάνοντας την μεταβλητή *cash,* αυξάνουμε την μεταβλητή *c3* καθώς το πλήθος αποτυχημένων κρατήσεων λόγω αδυναμία πληρωμής αυξήθηκε κατά 1 και τερματίζουμε το νήμα. Επιπλέον, αποδεσμεύουμε τις πιασμένες θέσεις στον πίνακα θέσεων. Απεναντίας, αν πραγματοποιήθει η πληρωμή , το κόστος των θέσεων μεταφέρεται στον λογαριασμό της επιχείρησης , αυξάνουμε την μεταβλητή *c1* καθώς είχαμε μια πετυχημένη κράτηση και το νήμα τερματίζει. Να σημειωθεί πως στις παραπάνω μεταβολές των τιμών καθολικών μεταβλητών κλειδώσαμε και ξεκλειδώσαμε κατάλληλα χρησιμοποιώντας το mutex του καθενός.

Για την υλοποίηση της gettime: Για τον μέσο χρόνο εξυπηρέτησης των πελατών κρατάμε τον τρέχοντα χρόνο στη μεταβλητή start3 καθώς μπαίνουμε στην συνάρτηση booking και έπειτα βρίσκουμε το σημείο όπου θα ολοκληρωθεί η εξυπηρέτηση. Στην προκειμένη περίπτωση βάσει του κώδικά μας είναι σε δύο σημεία. Το πρώτο είναι ο τερματισμός λόγω ανεπάρκειας θέσεων και το δεύτερο είναι αν πραγματοποιηθεί ή όχι η πληρωμή όπου λαμβάνουμε την τρέχουσα χρονική στιγμή και την τοποθετούμε στη μεταβλητή finish3 και έπειτα προσθέτουμε τη διαφορά των δύο μεταβλητών στην καθολική μεταβλητή time3. Για τον μέσο χρόνο αναμονής των πελατών αντίστοιχα μόλις ξεκινάει η booking κρατάμε τον χρόνο στην μεταβλητή start. Επειδή αυτό το ερώτημα χωρίζεται σε δύο μέρη χρησιμοποιούμε για αυτό δύο καθολικές μεταβλητές την time1 και την time2. Στην time1 προσθέτουμε το χρόνο που περιμένουν οι πελάτες μέχρι να επικοινωνήσουν με τον τηλεφωνητή. Έπειτα κρατάμε τον τρέχοντα χρόνο στην μεταβλητή start2. Αυτό θα γίνει αν η μεταβλητή cost είναι διάφορη του μηδενός, δηλαδή εάν υπάρχουν διαθέσιμες θέσεις. Έπειτα αφού πληρώσει και μειωθεί η τιμή cash θα κρατήσουμε το χρόνο και θα βρούμε τη διαφορά τον χρόνων και θα την προσθέσουμε στη μεταβλητή time2. Τέλος στη main θα εμφανίσουμε τους (μέσους) χρόνους (time1+time2) διαιρώντας τους με τον αριθμό των πελατών και αντίστοιχα για τον time3 διαιρώντας τον με τον αριθμό των πελατών.