Λειτουργικά Συστήματα

Προγραμματιστική Εργασία Εαρινού Εξαμήνου 2021-22

Σύστημα αγοράς θέσεων για θεατρικές παραστάσεις

Σε γλώσσα C με χρήση pthreads και mutexes

Κωνσταντίνος Μάρκο 3190112

Βόβλας Δημήτριος 3150016

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

1 Δομή Κώδικα

Ο κώδικας για την υλοποίηση του συστήματος αγοράς θέσεων αποτελείται από τρία αρχεία, το αρχείο των δηλώσεων (p3190112-p3150016-res.h), το αρχείο του κώδικα σε γλώσσα C (p3190112-p3150016-res.c) και το αρχείο μεταγλώττισης/εκτέλεσης (p3190112-p3150016-res.sh). Ακολουθούν πιο αναλυτικές πληροφορίες για τα τρία αυτά αρχεία.

1.1 Αρχείο Δηλώσεων

Στο αρχείο των δηλώσεων ορίζονται όλες οι σταθερές οι οποίες μας υποδεικνύονται στο μέρος “Είσοδος και δεδομένα” στην εκφώνησης της προγραμματιστικής εργασίας. Οι τίτλοι των σταθερών έχουν αλλαχτεί για διευκόλυνση της κατανόηση του κώδικα αλλά οι λειτουργίες τους παραμένουν ίδιες με αυτές που ζητούνται.

1.2 Αρχείο Μεταγλώττισης/Εκτέλεσης

Το αρχείο μεταγλώττισης/εκτέλεσης αποτελείται από δύο εντολές κονσόλας, την εντολή για τη μεταγλώττιση και την εντολή για την εκτέλεση. Για την ορθή εκτέλεση του προγράμματος μεταβείτε μέσω της κονσόλας στο directory του προγράμματος, όπου αρχικά πρέπει να εκτελέσετε την εντολή chmod +x test-res.sh και στη συνέχεια την εντολή ./test-res.sh. Θα αντιληφθείτε την εκτέλεση αφού στην κονσόλα θα εμφανιστούν οι εκτυπώσεις του προγράμματος.

1.3 Αρχείο Κώδικα

Το αρχείο του κώδικα για την ευκολία της αναφοράς θα το χωρισουμε σε τρία νοητικά μέρη, το μέρος των include, των δηλώσεων των global μεταβλητών και των δηλώσεων για την υλοποίηση των mutexes, το μέρος της main function και το μέρος της threadRunner function.

1.3.1 Includes | Global Variables Declaration | Mutexes Declaration

Σε αυτό το μέρος του κώδικα αρχικά γίνονται include οι βιβλιοθήκες της c που θα χρησιμοποιηθούν για την υλοποίηση και το include του αρχείου των δηλώσεων. Στη συνέχεια γίνονται οι δηλώσεις των global μεταβλητών που χρησιμοποιούνται στο πρόγραμμα και οι δηλώσεις των αναλόγων mutexes που χρειάζονται, πιο αναλυτικά.

customerID: Τα ID των πελατών.
availableTicketLines: Ο διαθέσιμος αριθμό από τηλεφωνητές ανά πάσα στιγμή. Διαθέτει mutex και condition.
availableCashLines: Ο διαθέσιμος αριθμό από ταμίες ανά πάσα στιγμή. Διαθέτει mutex και condition.

earnings: Τα έσοδα. Διαθέτει mutex.
randSeed: Ο αριθμός για την υλοποίηση της τυχαιότητας που θα δοθεί από το χρήστη.
seatsA[numZoneA][numSeats]: Το πλάνο των θέσεων του zone A. Διαθέτει mutex.
seatsB[numZoneB][numSeats]: Το πλάνο των θέσεων του zone B. Διαθέτει mutex.
availableSeatsA[numZoneA]: Οι διαθέσιμες θέσεις ανά σειρά στο zone A.
availableSeatsB[numZoneB]: Οι διαθέσιμες θέσεις ανά σειρά στο zone B.
successCount: Ο συνολικός αριθμός των επιτυχημένων αγορών. Διαθέτει mutex.
failCountSeat: Ο συνολικός αριθμός των αποτυχημένων αγορών λόγω έλλειψης θέσεων. Διαθέτει mutex.
failCountCard: Ο συνολικός αριθμός των αποτυχημένων αγορών λόγω αδυναμίας εκτέλεσης πληρωμής με κάρτα. Διαθέτει mutex.
customersCountCashLine: Ο συνολικός αριθμός των πελατών που πέρασαν από το ταμείο. Διαθέτει mutex.
sumTicketLineTime: Τα συνολικά δευτερόλεπτα που περιμένουν όλοι οι πελάτες για να εξυπηρετηθούν απο τους τηλεφωνητές. Διαθέτει mutex.
sumCashLineTime: Τα συνολικά δευτερόλεπτα που περιμένουν όλοι οι πελάτες για να εξυπηρετηθούν απο τους ταμίες. Διαθέτει mutex.
sumCustomersTime: Τα συνολικά δευτερόλεπτα που πέρνει σε όλους τους πελάτες να πραγματοποιήσουν επιτυχως ή ανεπιτυχώς μια αγορά. Διαθέτει mutex.

1.3.2 Main Function

Στη main συνάρτηση του κώδικά μας αρχικά ελέγχουμε αν τα arguments που έδωσε ο χρήστης είναι σωστά στο πλήθος. Στη συνέχεια αρχικοποιούμε τους δυο πίνακες των θέσεων με -1 και τους δυο πίνακες των θέσεων ανά σειρά με τις αντίστοιχες σειρές ανά ζώνη. Ακολούθως αρχικοποιούμε όλα τα mutexes και τα conditions και αρχικοποιούμε τη μεταβλητή του πλήθους των πελατών και του τυχαίου σπόρου με τα δοθέντα arguments του χρήστη. Μετά κάνουμε allocate την απαραίτητη μνήμη για τον πίνακα των ID των πελατών και δημιουργούμε τον πίνακα για τα pthreads. Μέσα σε ένα for loop γίνεται το sleep για να πετύχουμε την καθυστέρηση των τηλεφωνημάτων, κατασκευάζουμε τα threads των πελατών τα οποία εκτελούν την threadRunner συνάρτηση παίρνοντας ως όρισμα το ID του συγκεκριμένου πελάτη αυτού του thread και αφού τελειώσει, ένα ακόμα for loop ξεκινάει και κάνει join τα threads έτσι ώστε το parent thread να περιμένει τα υπόλοιπα threads πριν τελειώσουν την εκτέλεσή τους για να συνεχίσει.

Αφού επιστρέψουν όλα τα threads από τις διεργασίες τους εκτυπώνεται το πλάνο των δύο ζωνών όπου κάθε θέση αντιστοιχεί σε ένα ID πελάτη ή αν η θέση είναι κενή τότε μένη και η θέση χωρίς αριθμό. Μετά το πλάνο των θέσεων εκτυπώνονται όλα τα στατιστικά που έχουν ζητηθεί στην εκφώνηση της εργασίας. Τέλος καταστρέφονται τα mutexes και τα conditions και απελευθερώνεται η μνήμη του πίνακα ID των πελατών.

1.3.3 Thread Runner Function

Η παρούσα μέθοδος αποτελεί τη μέθοδο την οποία εκτελεί κάθε thread με όρισμα το id του τρέχοντος πελάτι. Στην αρχή της μεθόδου δημιουργούμε τα απαραίτητα structs για να μπορέσουμε να πάρουμε τα χρονικά δεδομένα έτσι ώστε να παράξουμε τα ανάλογα στατιστικά, δημιουργούμε μια μεταβλητή με το ID του πελάτη, δημιουργούμε μια μεταβλητή με το πλήθος των θέσεων που θέλει ο πελάτης με τη βοήθεια του τυχαίου σπόρου και δημιουργούμε κάποιες int μεταβλητές που θα μας βοηθήσουν να βλέπουμε την πορεία της αγοράς του πελάτη και από ποια στάδια πέρασε, πιο αναλυτικά.

zoneA: Ο πελάτης έχει επιλέξει τη ζώνη Α.
zoneB: Ο πελάτης έχει επιλέξει τη ζώνη Β.
isAvailableA: Βρέθηκαν διαθέσιμες θέσεις στην περίπτωση αν έχει επιλεγεί η ζώνη Α.
isAvailableB: Βρέθηκαν διαθέσιμες θέσεις στην περίπτωση αν έχει επιλεγεί η ζώνη Β.
foundSeats: Βρέθηκαν θέσεις.

Στη συνέχεια εκτυπώνεται ένα μήνυμα του τύπου:
“New customer ID: “ το ID του πελάτη.

Για να γίνει πιο κατανοητή η επεξήγηση του υπόλοιπου κομματιού του κώδικα θα χωρίσουμε τη μέθοδο σε δυο μέρη, το μέρος του Ticket Line και του Reservation και το μέρος του Cash Line και του Payment.

1.3.3.1 Ticket Line | Reservation

Αρχικά κρατάμε τη χρονική στιγμή στην οποία ξεκίνησε ο customer και με τη βοήθεια ενός while loop και του condition wait σε περίπτωση που δεν υπάρχει κάποιος ελεύθερος τηλεφωνητής ο πελάτης περιμένει. Μόλις περάσει το while loop σημαίνει ότι ο πελάτης έχει βρει διαθέσιμο τηλεφωνητή οπότε μειωνεται ο αριθμος των διαθέσιμων τηλεφωνητών και κρατάμε τη χρονική στιγμή την οποία ο πελάτης βρήκε διαθέσιμο τηλεφωνητή και προσθέτουμε τη διαφορά των κρατουμένων χρόνων στη μεταβλητή με το πλήρος του χρόνου που οι πελάτες πέρασαν στην αναμονή για τηλεφωνητή. Στη συνέχεια γίνεται το sleep για την προσομοίωση της διαδικασίας του reservation και μεταβαίνουμε στην επιλογή ενός εκ των δυο zones από τον πελάτη με τη χρήση του τυχαίου σπόρου.

Είτε επιλεγεί η α ζώνη είτε η β ζώνη η διαδικασία είναι παρόμοια. Με λίγα λόγια αυτό που κάνουμε είναι να κοιτάμε γραμμή γραμμή στον πίνακα των διαθέσιμων θέσεων ανά γραμμή αν υπάρχουν οι απαραίτητες ελεύθερες θέσεις και μόλις βρούμε μια γραμμή στην οποία υπάρχουν ελεύθερες θέσεις για τον πελάτη τότε τις αφαιρούμε και σε αυτή τη γραμμή του πίνακα των θέσεων της ζώνης καταχωρούμε το ID του.

Υπάρχει και η περίπτωση να μην βρεθούν θέσεις οπότε το πρόγραμμα θα συνεχίσει χωρις να καταχωρήσει κάτι στον πίνακα.

Σε περίπτωση που βρεθούν θέσεις θα εμφανιστεί μήνυμα του τύπου: “Customer ID: “ το ID του πελάτη “ | “ η ζώνη που έχει επιλέξει “ | Amount of seats: “ το πλήθος των θέσεων που θέλει να κλείσει “ | Row: “ το νούμερο της σειράς των εισητιριων “ | Seat: “ οι αριθμοί των θέσεων που έχει κρατήσει “ | Seats Reserved ”. Και θα θέσουμε τη μεταβλητή εύρεσης θέσης σε επιτυχή.

Σε περίπτωση που δεν βρεθούν θέσεις θα εμφανιστεί μήνυμα του τύπου: “Customer ID: “ το ID του πελάτη “ | “ η ζώνη που έχει επιλέξει “ | Amount of seats: “ το πλήθος των θέσεων που θέλει να κλείσει “ | Row: “ το νούμερο της σειράς των εισητιριων “ | Seat: “ οι αριθμοί των θέσεων που έχει κρατήσει “ | Reservation Failed | No More Available Seats in ” και η ζώνη που έχει επιλέξει. Και θα θέσουμε τη μεταβλητή εύρεσης θέσης σε ανεπιτυχή.

Πριν τελειωσει η διαδικασία με τον τηλεφωνητή πρέπει να αυξήσουμε τις διαθέσιμες θέσεις στους τηλεφωνητές και να στείλουμε το απαραίτητο signal στο thread που κάνει wait για τηλεφωνητή.

1.3.3.2 Cash Line | Payment

Στην διαδικασία της πληρωμής θα μπουν μόνο οι πελάτες που έκαναν επιτυχή εύρεση θέσεων.

Αρχικά κρατάμε τη χρονική στιγμή στην οποία μπήκε ο πελάτης στην ουρά του ταμείου και με τη βοήθεια ενός while loop και του condition wait σε περίπτωση που δεν υπάρχει κάποιος ελεύθερος ταμίας ο πελάτης περιμένει. Μόλις περάσει το while loop σημαίνει ότι ο πελάτης έχει βρει διαθέσιμο ταμείο οπότε μειωνεται ο αριθμος των διαθεσίμων ταμείων και κρατάμε τη χρονική στιγμή την οποία ο πελάτης βρήκε διαθέσιμο ταμείο και προσθέτουμε τη διαφορά των κρατουμένων χρόνων στη μεταβλητή με το πλήρος του χρόνου που οι πελάτες πέρασαν στην αναμονή για ταμείο. Στη συνέχεια γίνεται το sleep για την προσομοίωση της διαδικασίας του payment με τη χρήση του τυχαίου σπόρου.

Πάλι με τη βοήθεια του τυχαίου σπόρου αποφασίζουμε αν η πληρωμή θα γινει με επιτυχία ή όχι. Σε περίπτωση που γίνει με επιτυχία τότε θα αυξήσουμε τις επιτυχείς πληρωμές για τα στατιστικά, θα αυξήσουμε τα earnings με βάση τη ζώνη του πελάτη και θα εμφανίσουμε μηνυμα του τύπου: “Customer ID: “ το ID του πελάτη “ | Successful payment | Total : “ το συνολικό ποσό “ euros”. Σε περίπτωση που είναι ανεπιτυχής η πληρωμή θα αυξήσουμε τις ανεπιτυχής πληρωμές για τα στατιστικά και θα επιστρέψουμε τις θέσεις. Οι θέσεις θα επιστραφούν με προσπέλαση του πίνακα

των θέσεων της ζώνης του πελάτη και όπου βρεθεί το ID του θα αλλάξουμε την τιμή σε -1, δηλαδή ότι η θέση είναι κενή.

Πριν τελειωσει η διαδικασία με το ταμείο πρέπει να αυξήσουμε τις διαθέσιμες θέσεις στα ταμεία και να στείλουμε το απαραίτητο signal στο thread που κάνει wait για ταμείο.

Τέλος κρατάμε τη χρονική στιγμή που τελείωσε ο πελάτης και τη διαδικασία του ταμείου ή τη χρονική στιγμή μετά τον τηλεφωνητή εφόσον δεν βρηκε εισητηρια και προσθέτουμε τη διαφορά των κρατουμένων χρόνων στη μεταβλητή με το πλήθος του ολικού χρόνου εξυπηρέτησης του πελάτη. Μετά το τέλος όλων των παραπάνω διαδικασιών το thread κάνει exit.

2 Εκτέλεση Προγράμματος

Βήμα 1ο: Αποσυμπιέστε το αρχείο p3190112-p3150016-res.7z

Βήμα 2ο: Μέσω του terminal μεταβείτε στον φάκελο που έγινε η αποσυμπίεση

Βήμα 3ο: Πληκτρολογήστε στο terminal: chmod +x test-res.sh

Βήμα 4ο: Πληκτρολογήστε στο terminal: ./test-res.sh

Βήμα 5ο: Στην διεπαφή του terminal θα εμφανιστούν οι έξοδοι του προγράμματος, περιμέντε μέχρι να ολοκληρωθεί η εκτέλεσή του και αφού τελειώσουν οι διεργασίες όλων των threads θα τυπωθεί το πλάνο του θεάτρου και τα στατιστικά.