ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Λειτουργικά Συστήματα
Τμήμα Πληροφορικής

Εαρινό Εξάμηνο 2021-2022

Αθανάσιος Τριφώνης-3200298
Νικόλαος Δούρος-3200043

ΑΝΑΦΟΡΑ

Περίληψη

Στην παρούσα εργασία υλοποιήσαμε ένα σύστημα αγοράς θέσεων χρησιμοποιώντας το POSIX threads. Κάθε πελάτης κλείνει τον αριθμό θέσεων που επιθυμεί, στη συνέχεια πληρώνει με πιστωτική κάρτα και τέλος τα χρήματα μεταφέρονται στον λογαριασμό της εταιρείας. Επειδή ο αριθμός των σημείων εξυπηρέτησης πελατών είναι περιορισμένος το πρόγραμμα χρησιμοποιεί αμοιβαίο αποκλεισμό και συγχρονισμό στα νήματα (πελάτες).

Δομή κώδικα

Ο κώδικας είναι χωρισμένος σε δύο αρχεία, εκ των οποίων το header χρησιμοποιείται για δηλώσεις σταθερών, μεταβλητών και κάποιων συναρτήσεων (συμπεριλάβαμε στο ίδιο αρχείο και τις υλοποιήσεις κάποιων μικρών συναρτήσεων ώστε να κρατήσουμε πιο ευανάγνωστο το .c αρχείο). Πιο συγκεκριμένα οι συναρτήσεις στο **header** είναι οι:

```
void print_stats(int case1, int case2, int customers)
```
Η print_stats τυπώνει τα συνολικά έσοδα των πωλήσεων καθώς και τα ποσοστά κάθε μιας από τις 3 περιπτώσεις που μπορεί να καταλήξει ένας πελάτης και στο τέλος τυπώνει τον μέσο χρόνο αναμονής και εξυπηρέτησης. Οι παράμετροι case1 και case2 είναι το πλήθος των αποτυχιών λόγω ανεπαρκών θέσεων και το πλήθος σφαλμάτων κατά την πληρωμή. Αφαιρόντας αυτές τις 2 περιπτώσεις από το σύνολο των πελατών βρίσκουμε το πλήθος που ανήκει στις επιτυχημένες κρατήσεις. Εμφανίζουμε τα παραπάνω σε ποσοστά του συνολικού αριθμού πελατών.

```
int get_cost(char zone, int num)
```
Η συνάρτηση get_cost επιστρέφει το κόστος της κράτησης για τον πελάτη, το οποίο υπολογίζεται ως η η τιμή της θέσης(εξαρτάται από την ζώνη) επί τον αριθμό των θέσεων.

`void print_booking(int cid, int rc, int seats, int cost)` Η συνάρτηση print_booking εμφανίζει τα στοιχεία της επιτυχημένης κράτησης του πελάτη με κωδικό cid, με κόστος cost, και για τις θέσεις από rc (κωδικός πρώτης θέσης στη κράτηση) μέχρι την τελευταία θέση της κράτησης.

Στο **.c αρχείο** εντός της συνάρτησης main αρχικά γίνεται έλεγχος στον αριθμό ορισμάτων που δίνονται κατά την εκτέλεση, ώστε το πρόγραμμα να τρέχει μόνο όταν δίνονται 2 ακριβώς ορίσματα(με το όνομα του εκτελέσιμου πρέπει argc ==3). Σε διαφορετική περίπτωση εμφανίζεται μήνυμα σφάλματος και το πρόγραμμα τερματίζει.

Στην συνέχεια διαβάζονται τα περιεχόμενα των ορισμάτων εκτέλεσης, το πρώτο καθορίζει τον αριθμό πελατών και το δεύτερο είναι ο «σπόρος» της γενήτριας τυχαίων αριθμών. Δημιουργείται ένας πίνακας έπειτα που θα περιλαμβάνει το id του κάθε πελάτη, καθώς και ένας πίνακας με threads για τον κάθε πελάτη. Επίσης έχουμε 2 global pointers, το seat_plan και row_free_seats τα οποία χρησιμοποιούνται ως πίνακες και αρχικοποιούνται σε αυτό το σημείο δεσμεύοντας δυναμικά μνήμη με βάση το πλήθος των θέσεων στην αίθουσα και τον αριθμό των σειρών. Ο πίνακας seat_plan έχει τιμή -1 στις κενές θέσεις ενώ στις κρατημένες έχει το id του πελάτη που τις κράτησε. Ο row_free_seats έχει τον αριθμό των ελεύθερων θέσεων της κάθε σειράς και ο ρόλος του είναι να επιταχύνει την εύρεση ελεύθερων θέσεων για έναν πελάτη, καθώς ο τηλεφωνητής δεν θα ψάχνει σε σειρές που δεν έχουν αρκετές θέσεις σε σχέση με αυτές που θέλει να κρατήσει στην ζώνη που επέλεξε ο πελάτης.

Χρησιμοποιείται επίσης μία συνάρτηση initializer η οποία απλώς κάνει init όλα τα mutexes και conditions και στο τέλος βρίσκεται η αντίστοιχη destroyer που τα καταστρέφει.

Αφού αρχικοποιηθούν τα mutexes και τα conditions, σε μία for δημιουργούμε ένα thread ανά πελάτη και εκτελεί το κάθε ένα την ρουτίνα routine με όρισμα το id του πελάτη(τα id ξεκινούν από το 0). Το νήμα του πρώτου πελάτη δημιουργείται απευθείας την στιγμή 0, αλλά για κάθε επόμενο νήμα κάνουμε ένα sleep για ένα τυχαίο χρονικό διάστημα πριν το δημιουργήσουμε και πάρει τηλέφωνο.

Στην **routine** που εκτελούν τα νήματα αποθηκεύουμε τοπικά το id του πελάτη και δημιουργούμε ένα τοπικό seed = argSeed + id, όπου argSeed είναι το seed που δίνεται ως argument κατά την εκτέλεση του προγράμματος. Δημιουργούνται οι δομές timespec start, tel_start, tel_end, cash_start, finish και χρησιμοποιούνται στην συνέχεια για να υπολογίσουμε χρόνο αναμονής και εξυπηρέτησης ανάμεσα στα στάδια που θα περάσει ένας πελάτης μέχρι να ολοκληρωθεί η εκτέλεση της ρουτίνας. Με την κλήση της clock_gettime και με 1ο όρισμα το CLOCK_REALTIME και 2^{ο} το timespec που θέλουμε, αποθηκεύουμε κάθε φορά την χρονική στιγμή στο επιθυμητό timespec για χρήση αργότερα.

Σε αυτό το σημείο χρησιμοποιούμε μία μεταβλητή available_tel η οποία δείχνει τον αριθμό των ελεύθερων τηλεφωνητών, όταν δεν υπάρχουν ελεύθεροι τηλεφωνητές οι πελάτες περιμένουν ένα σήμα. Στην περιοχή που ενημερώνεται η τιμή των ελεύθερων τηλεφωνητών χρησιμοποιείται το mutex_tel για αμοιβαίο αποκλεισμό. Όταν ένας πελάτης βρεί ελεύθερο τηλεφωνητή κλειδώνει το mutex μειώνει τους ελεύθερους τηλεφωνητές κατά 1 και ξεκλειδώνει το mutex.

Έπειτα ο πελάτης επιλέγει μία ζώνη και έναν αριθμό εισητηρίων και ο τηλεφωνητής κλειδώνει την αντίστοιχη ζώνη ώστε να μην μπορεί κάποιος άλλος να έχει πρόσβαση σε αυτές τις σειρές που αντιστοιχούν στην ζώνη(το υλοποιούμε με την δημιουργία ενός mutex_zone το οποίο δείχνει στο mutex της επιλεγμένης ζώνης κάθε φορά, mutex_zoneA ή mutex_zoneB). Στην συνέχεια συμβαίνει

μία κλήση της sleep για να υποδηλώσουμε πως ο τηλεφωνητής χρειάζεται κάποιο χρόνο για να ελέγξει την διαθεσιμότητα θέσεων, και καλείται η συνάρτηση booking.

```
int booking(char zone, int seats, int cid, int *free_seats, int *plan)
```

Η συνάρτηση booking παίρνει ως ορίσματα έναν χαρακτήρα Α ή Β που υποδηλώνει την ζώνη, τον αριθμό των εισητηρίων που θέλει ο πελάτης, τον κωδικό του πελάτη, τον πίνακα με τις ελεύθερες θέσεις ανά σειρά και το πλάνο των θέσεων.

Με βάση την ζώνη αποφασίζουμε σε ποιό τμήμα των πινάκων θα ψάξουμε για να ελέγξουμε την διαθεσιμότητα. Με μία for τρέχουμε την κάθε σειρά της ζώνης και ψάχνουμε να βρούμε την πρώτη σειρά στην οποία υπάρχουν τουλάχιστον όσες ελεύθερες θέσεις ζήτησε ο πελάτης. Αν δεν υπάρχει τέτοια σειρά η συνάρτηση booking επιστρέφει με κωδικό -1 που λέει έπειτα στην ρουτίνα στην οποία βρίσκεται να τρέξει την περίπτωση αποτυχίας λόγω ανεπαρκών θέσεων. Σε περίπτωση που μία σειρά έχει αρκετές θέσεις με μια for αρχίζουμε από την πρώτη θέση της σειράς και κοιτάμε στο seat_plan αν είναι ελεύθερη. Κάθε φορά που βρίσκουμε ελεύθερη αυξάνουμε το streak κατά 1, ενώ αν βρουμε μία που δεν είναι ελεύθερη το streak μηδενίζεται καθώς ψάχνουμε για συνεχόμενες ελεύθερες θέσεις. Αν φτάσουμε σε ένα σημείο που ξέρουμε πως με βάση την τρέχουσα θέση και streak και τον αριθμό των θέσεων που έχει η κάθε σειρά και τον αριθμό εισητηρίων που θέλει ο πελάτης δεν θα μπορέσει να χωρέσει σε αυτή την σειρά τότε πάμε στην επόμενη.

(Ο παραπάνω έλεγχος υλοποιείται με την σχέση j - streak <= NSEAT – seats και ένα παράδειγμα που χρήσης είναι όταν ένας πελάτης θέλει 5 θέσεις και εμείς φτάνουμε στην θέση 6 και βλέπουμε πως είναι κρατημένη, αυτό σημαίνει πως το streak θα μηδενιστεί και δεν υπάρχει λόγος να ψάξουμε τις υπόλοιπες θέσεις της σειράς γιατί ακόμα και αν είναι ελεύθερες το streak δεν θα μπορέσει να γίνει ίσο με τον αριθμό εισητηρίων που ζήτησε ο πελάτης)

Αν τελειώσουν οι σειρές της ζώνης πάμε σε έξοδο με -1 που σηματοδοτεί αποτυχία λόγω ανεπαρκών θέσεων. Διαφορετικά αν καταφέρουμε να βρούμε όσες συνεχόμενες θέσεις ζητάει ο πελάτης(streak == seats) τότε με μία for πάμε και τις κλείνουμε βάζοντας στο πλάνο τον κωδικό του πελάτη σε κάθε μία από τις θέσεις που ζήτησε. Έπειτα εφόσον κλείσαμε τις θέσεις τις αφαιρούμε από τις ελεύθερες της τρέχουσας σειράς και τέλος επιστρέφουμε τον αριθμό της πρώτης θέσης στο seat_plan από αυτές που μόλις κλείσαμε(Θα χρησιμοποιηθεί στη συνέχεια από την ρουτίνα σε περίπτωση πιθανής ακύρωσης ώστε να γνωρίζουμε σε ποιο σημείο του seat_plan μπορεί να χρειαστεί να ελευθερωθούν θέσεις από την cancel_booking).

Πίσω στην ρουτίνα, ανάλογα με το πως πήγε η εύρεση θέσεων οδηγούμαστε είτε σε αποτυχία είτε συνεχίζουμε στην επόμενη φάση. Στο πρώτο ενδεχόμενο πάμε όταν η συνάρτηση booking επιστρέφει τιμή -1. Κλειδώνουμε την οθόνη με το mutex_screen και τυπώνουμε το αντίστοιχο μήνυμα αποτυχίας. Επίσης ξεκλειδώνουμε την ζώνη και την οθόνη, κλειδώνουμε το mutex_tel για να αυξήσουμε τους τηλεφωνητές κατα 1 και έπειτα ξεκλειδώνουμε τους τηλεφωνητές καταγράφουμε την περίπτωση αποτυχίας στο case_not_enough_seats κλειδώνοντας την περιοχή πριν μπούμε να ανανεώσουμε την μεταβλητή και ξεκλειδώνοντας αφού βγούμε. Στέλνεται σήμα με το cond_tel αφού ελευθερώθηκε τηλεφωνητής και τέλος κάνουμε το αντίστοιχο με τις μεταβλητές των χρόνων αναμονής και εξυπηρέτησης με αυτό που κάναμε για την μεταβλητή με τον αριθμό αποτυχιών λογω μη εύρεσης θέσεων, έπειτα το νήμα καλεί το pthread_exit(NULL) και επιστρέφει στην join NULL.

Στην περίπτωση που βρήκαμε θέσεις για τον πελάτη τότε ακολουθείται μία παρόμια διαδικασία με αυτή των τηλεφωνητών με παρόμια mutexes, μεταβλητές και conditions, στην οποία οι πελάτες αντιστοιχίζονται σε ελεύθερους ταμίες. Ο ταμίας χρειάζεται ένα τυχαίο χρόνο να κάνει την πληρωμή το υλοποιούμε με την sleep, και παράγεται μία τυχαία τιμή βάση της οποίας η πληρωμή επιτυγχάνεται ή αποτυγχάνει. Στην περίπτωση αποτυχίας πληρωμής ακολουθείται παρεμφερή διαδικασία με αυτή της αποτυχίας από μη εύρεσης θέσεων, ενημερώνοντας τις αντίστοιχες μεταβλητές κάνοντας τα απαραίτητα κλειδώματα και ξεκλειδώματα, αλλά με την διαφορά πως τώρα θα πρέπει οι ταμίες να ακυρώσουν τις κρατημένες θέσεις από το seat_plan, οπότε καλείται η συνάρτηση **cancel_booking** που κάνει ακριβώς αυτό.

```
*/
void cancel_booking(int starting_seat, int seats, int *free_seats, int *plan)
```

Η συνάρτηση αυτή παίρνει ως 1^{ο} όρισμα την τιμή που επέστρεψε το booking νωρίτερα στην εκτέλεση του, 2^{ο} όρισμα τον αριθμό εισητηρίων που έκλεισε ο πελάτης και 3^{ο}, 4^{ο} όρισμα τους δύο πίνακες με τις ελεύθερες θέσεις ανά σειρά και το πλάνο των θέσεων. Τρέχει μια for η οποία πηγαίνει στην θέση starting_seat στο πλάνο(πρώτη θέση της συγκεκριμένης κράτησης) και ορίζει ως ελεύθερες αυτή και τις υπόλοιπες θέσεις που είχαν κρατηθεί από τον πελάτη(θέτωντας τιμή -1 στο πλάνο). Τέλος η συνάρτηση ανανεώνει τις ελεύθερες θέσεις της σειράς αυξάνοντας τες κατά αριθμό ίσο με seats.

Στην περίπτωση που η συναλλαγή πραγματοποιήθηκε τότε το ποσό προστίθεται στον λογαριασμό της εταιρείας(κάνοντας το απαραίτητο κλείδωμα και ξεκλείδωμα πριν και μετά την πρόσβαση στην μεταβλητή) και εν συνεχεία ελευθερώνεται ο ταμίας και στέλνεται σήμα(με παρόμοιο τρόπο με τους τηλεφωνητές) ενημερώνονται τα κατάλληλα timespec και χρόνοι αναμονής και εξυπηρέτησης(πάλι με παρόμοιο τρόπο όπως στις προηγούμενες φορές) γίνεται η αντίστοιχη εκτύπωση για την επιτυχημένη κράτηση και πληρωμή των θέσεων του πελάτη και τελικά το νήμα κάνει pthread_exit(NULL) και επιστρέφει.

Αφού επιστρέψουν όλα τα νήματα – πελάτες το αρχικό νήμα κάνει τις προβλεπόμενες εκτυπώσεις οι οποίες αφορούν το ποσό του λογαριασμού της εταιρείας, τα στατιστικά για την κάθε μία εκ των 3 περιπτώσεων που μπορεί να έφτασε ένας πελάτης και έπειτα καλεί την destroyer συνάρτηση για καταστροφή των mutexes και conditions που χρησιμοποιηθηκαν και την συνάρτηση free για να απελευθερώσει την δυναμική μνήμη των seat_plan και row_free_seats πινάκων.