ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Εαρινό Εξάμηνο 2018-2019

2η Προγραμματιστική Εργασία

Δανοπούλου Έμυ [3170033]

Μπαλή Νίκη [3170114]

Χαβιατζή Ελένη [3170172]

Header file

Στο header file p3170033-p3170114-p3170172-res2.h μαζί με τις σταθερές που υπήρχαν ήδη από την πρώτη εργασία προστέθηκαν και οι: Ncash, NzoneA, NzoneB, NzoneC, PzoneA, PzoneB, PzoneC, CzoneA, CzoneB, CzoneCTcashlow, Tcashhigh και η σταθερά Nseat απέκτησε νέα τιμή γιατί τώρα εκφράζει το πλήθος θέσεων σε κάθε σειρά. Επίσης, προστέθηκε η ακέραια μεταβλητή cash_waiting_time που θα έχει τον χρόνο αναμονής του ταμία, και οι double μεταβλητές st, ep, fz και cs. Η st έχει το πλήθος επιτυχών συναλλαγών, η ep το πλήθος σφαλμάτων κατά την πληρωμή, η fz το πλήθος συναλλαγών που τερμάτισαν λόγω κάποιας γεμάτης ζώνης και η cs το πλήθος συναλλαγών που τερμάτισαν λόγω μη διαθέσιμων συνεχόμενων θέσεων. Ακόμη, αντί για το seats_mutex έχουμε τρία mutexes για τις τρεις ζώνες θέσεων αντίστοιχα (seatsA_mutex, seatsB_mutex, seatsC_mutex) και προστέθηκε ο cash_mutex για τον ταμία. Επιπλέον, ορίσαμε τρεις πίνακες (SeatsA, SeatsB, SeatsC) που περιλαμβάνουν τις θέσεις κάθε ζώνης. Πέρα από αυτές τις αλλαγές, το header file περιέχει ό,τι άλλο περιείχε και το header file της πρώτης εργασίας.

C file

Αρχικοποιούμε τις μεταβλητές seatsA, seatsB, seatsC με τις θέσεις κάθε ζώνης, τη μεταβλητή tele με το πλήθος διαθέσιμων τηλεφωνητών και τη μεταβλητή cash με το πλήθος διαθέσιμων ταμείων. Επίσης αρχικοποιούμε τις συνθήκες μεταβλητών tele_cond και cash_cond.

Συνάρτηση main

Η main έχει την ίδια δομή όπως και στην πρώτη εργασία, με μοναδική διαφορά τα mutexes τα οποία αρχικοποιούμε και τελικά καταστρέφουμε, και τα μηνύματα που εκτυπώνουμε.

Συνάρτηση Client

Η δομή της Client δεν διαφέρει με αυτή της πρώτης εργασίας, ως το σημείο όπου πρέπει να γίνει η κράτηση των θέσεων. Τότε συγκρίνουμε την τυχαία ακέραια μεταβλητή zone με την πιθανότητα να θέλουμε τη ζώνη Α (PzoneA) και εάν είναι μικρότερη ή ίση, τότε σημαίνει πώς ο πελάτης ζήτησε εισιτήρια για τη ζώνη Α. (Η zone λαμβάνει τιμές από το 1 μέχρι το 10.) Εάν η ζώνη Α είναι πλήρης (seatsA==0), τότε ακυρώνεται η κράτηση και αυξάνεται ο μετρητής fz.

Εάν όμως η ζώνη Α δεν είναι γεμάτη, τότε κλειδώνουμε το seatsA_mutex και μπαίνουμε σε ένα διπλό for loop το οποίο διατρέχει τον πίνακα SeatsA, δηλαδή τη ζώνη Α πρώτα κατά σειρά και μετά κατά θέση. Όταν επεξεργαζόμαστε το πρώτο εισιτήριο (ή το μοναδικό) από όσα ζήτησε ο πελάτης (SeatsA[j + i*Nseat]==0 && t==tickets), γίνεται αμέσως η κράτηση. Όταν επεξεργαζόμαστε εισιτήριο που δεν είναι το πρώτο, τότε πρώτα ελέγχουμε αν η αμέσως προηγούμενη θέση είναι κρατημένη για τον ίδιο πελάτη. Αν ναι, τότε γίνεται κανονικά η κράτηση, και αν όχι, τότε επιστρέφονται όσες θέσεις έχουν κρατηθεί ως στιγμής για τον συγκεκριμένο πελάτη και θα ψάξουμε το υπόλοιπο της σειράς για διαθέσιμες και συνεχόμενες θέσεις.

Στο κομμάτι της πληρωμής, αρχικά περιμένουμε μέχρι κάποιος ταμίας να είναι διαθέσιμος (pthread_cond_wait(&cash_cond, &cash_mutex)) και μετά κλειδώνουμε το cash_mutex, ελέγχοντας και για πιθανό σφάλμα. Αναμένουμε για χρόνο cash_waiting_time και ξεκλειδώνουμε το cash_mutex. Όσο είναι κλειδωμένο το time_mutex, προσμετράμε το cash_waiting_time στο συνολικό χρόνο αναμονής total_waiting_time.

Αν η πληρωμή είναι επιτυχής (payment_success <= Pcardsuccess), τότε υπολογίζεται το κατάλληλο κόστος (cost = CzoneA*tickets), εκτυπώνεται το κατάλληλο μήνυμα στην οθόνη για όσο είναι κλειδωμένο το screen_mutex και ενημερώνουμε το υπόλοιπο στην τράπεζα (balance = balance + cost) για όσο είναι κλειδωμένο το bank_mutex.

Αν η πληρωμή δεν είναι επιτυχής, τότε εκτυπώνουμε το κατάλληλο μήνυμα στην οθόνη και ακυρώνουμε τη συναλλαγή, επιστρέφοντας τις θέσεις που είχαμε κρατήσει. Επίσης, αυξάνουμε το μετρητή ep.

Τέλος, υπολογίζουμε το συνολικό χρόνο εξυπηρέτησης για αυτή τη συναλλαγή (total_service_time), αυξάνουμε το μετρητή st.

Με τον ίδιο ακριβώς τρόπο λειτουργεί και στην περίπτωση όπου ο πελάτης ζητήσει εισιτήρια για τη ζώνη Β ή τη ζώνη C. Οι μόνες διαφορές είναι στο πλήθος θέσεων κάθε ζώνης και στην τιμή του εισιτηρίου ανά ζώνη.