Λειτουργικά Συστήματα

Αλέξης Φιλιππακόπουλος : 3190212

Γεωργία Καρκάζη : 3190076

Δημήτρης Καββαδάς : 3190064

Ξεκινώντας το πρόγραμμα ζητάει από τον χρήστη να δώσει τον αριθμό των πελατών (Ncust) καθώς και να ορίσει την τιμή του σπόρου για την γεννήτρια τυχαίων αριθμών (seed). Αρχικοποιούμε δύο μεταβλητές (waitingTimeSum και serviceTimeSum) για τα στατιστικά των χρόνων, έναν δυσδιάστατο πίνακα theater [NzoneA + NzoneB] [Nseat] που λειτουργεί ως το πλάνο του θεάτρου, με την χρήση του malloc() δεσμεύουμε χώρο στην μνήμη για έναν δυσδιάστατο πίνακα plan [Ncust] [8] όπου θα χρησημοποιειθεί για το τελικό output για κάθε πελάτη που αντιπροσωπεύεται από κάθε row του πίνακα[0η στήλη: Η ζώνη που δίαλεξε/1η στήλη: Η σειρά που βρίσκονται οι θέσεις/2η στήλη: Ο χρόνος εξυπηρέτησης (service time) για αυτόν/3η στήλη ο χρόνος αναμονής για αυτόν(waiting time)/4η στήλη έως 8η στήλη: Πόσες θέσεις έκλεισε (max 5)].Τον πίνακα τον γεμίζουμε με -1 στην αρχή.Τέλος έχουμε έναν πίνακα customer_id [Ncust] με τα ID των πελατών και έναν πίνακα th[Ncust] για τα threads.

Στη συνάρτηση void* routine(void* arg) περνάμε τα customer IDs και ξεκινάμε κρατώντας τον χρόνο που ξεκινάει η εξυπηρέτηση(double start) με τα κατάλληλα mutex ελέγχουμε την διαθεσιμότητα των τηλεφωνητών, μόλις βρεθεί τηλεφωνητής σταματάμε να μετράμε τον χρόνο για την εύρεσή του (finishWaitOperator), χρησιμοποιούμε τις seatSeed,zoneSeed για ακόμα πιο τυχαίες τιμές και τις seatsToReserve και seatsZone (0 = A και 1 = B) για την επιλογή του

πελάτη όσο αφορά τον αριθμό των θέσεων και της ζώνης.Έπειτα κλειδώνουμε τον πίνακα του θεάτρου με το κατάλληλο mutex και ξεκινά η αναζήτηση θέσεων.Με την rowCounter βλέπουμε αν πάμε σε επόμενη γραμμή ώστε να μηδενίσουμε τις θέσεις για να εξασφαλιστεί ότι θα είναι συνεχόμενες και στην ίδια σειρά,με την seatCounter κρατάμε τις θέσεις προς κράτηση που βρίσκουμε,με την seatsIndex [seatsToReserve] κρατάμε τους αριθμούς των θέσεων και με την ticketCost το κόστος.

Με την συνθήκη (j – seatCounter + seatsToReserve) εξετάζουμε αν στην σειρά που ψάχνουμε οι ελεύθερες θέσεις που υπάρχουν είναι αρκετές για τον συγκεκριμένο πελάτη χωρίς να χρειαστεί να προσπελάσουμε όλες τις στήλες του πίνακα εκμεταλλευόμενοι την προυπόθεση να είναι συνεχόμενες οι θέσεις. Αφού βρούμε τις απαιτούμενες θέσεις (συνθήκη seatsCounter==seatsToReserve βάζουμε την τιμή 1 στις θέσεις που κλείσαμε στον πίνακα και με την goto jmp σταματάμε την προσπέλαση του πίνακα για να εξοικονομίσουμε χρόνο.

Στην jmp αν το flag==0 σημαίνει ότι δεν βρέθηκαν θέσεις οπότε ελευθερώνουμε τον τηλεφωνητή εκτυπώνουμε κατάλληλο μήνυμα και αυξάνουμε την unsuccessfullTransactionSeats που κρατάει τον αριθμό των αποτυχημένων συναλλαγών λόγω μη εύρεσης θέσεων.

Αν βρέθηκαν θέσεις ελευθερώνουμε τον τηλεφωνητή ξεκινάμε να κρατάμε χρόνο αναμονής για ταμία (startWaitCashier) και με το κατάλληλο mutex ελέγχουμε την διαθεσιμότητα τους. Όταν βρεθεί ταμίας σταματάμε τον χρόνο (finishWaitCashier), χρησιμοποιούμε την cardSeed για ακόμα ποιό τυχαία αποτελέσματα και την cardAccepted για να δούμε αν περνάει η κάρτα (0 = Πέρασε 1 = Δεν πέρασε).

Αν η πέρασε εκτυπώνουμε κατάλληλο μήνυμα, ενημερώνουμε τον πίνακα plan για την ζώνη, τις θέσεις και την σειρά, αυξάνουμε τον μετρητή successfullTransactions και βάζουμε τα χρήματα (ticketsCost) στον τραπεζικό λογαριασμό (bankAccount).

Αν η κάρτα δεν πέρασε εκτυπώνουμε κατάλληλο μήνυμα αυξάνουμε τον μετρητή unsuccessfullTransactionsCard και ελευθερώνουμε τις θέσεις που είχαν δεσμευτεί.

Τέλος ελευθερώνουμε τον ταμία,σταματάμε να μετράμε τον χρόνο εξυπηρέτησης (double finish),υπολογίζουμε τον χρόνο εξυπηρέτησης (serviceTime) και τον χρόνο αναμονής (waitingTime) του πελάτη, τα βάζουμε στον πίνακα plan και κάνουμε pthread_exit(NULL).

Πίσω στην main εκτυπώνουμε τα στατιστικά,τα κέρδη, και το πλάνο του θεάτρου, με την χρήση της free() ελευθερώνουμε την μνήμη που είχαμε δεσμεύσει και τέλος κάνουμε destroy() τα mutex και τα conditions που χρησιμοποιήσαμε.