Αναστάσιος Παπαπαναγιώτου

3200143

Αναφορά Λειτουργικών Συστημάτων

Δομή Κώδικα

main

Υπάρχουν 5 συναρτήσεις συνολικά. Η main() είναι η αρχική συνάρτηση και δέχεται δύο παραμέτρους, τον αριθμό των CLI arguments και ένα dynamic array με όλες τις τιμές των παραμέτρων. Στην main() αρχικοποιούμε όλα τα mutexes, condition variables, και καθολικές μεταβλητές, όπως τον αριθμό των διαθέσιμων τηλεφωνητών και ταμείων και το πλάνο του θεάτρου. Για κάθε πελάτη, αρχίζουμε να μετράμε τον χρόνο αναμονής και εξυπηρέτησης (clock_gettime) και μετά αν δεν είναι ο πρώτος πελάτης περιμένει για τυχαίο χρόνο στο διαστήμα $[t_{reslow}, t_{reshigh}]$. Επειδή η rand_r χρειάζεται pointer δεσμεύουμε χώρο στην μνήμη χρησιμοποιώντας το malloc και μετά ελευθερώνουμε τον χώρο με την free. Με αυτόν τον τρόπο μοιάζει να έρχεται ένας πελάτης μετά από τυχαίο χρόνο δευτερολέπτων. Όταν έρχεται ένας πελάτης, προσθέτουμε όλα τα arguments που θα περάσουμε στο thread σε ένα stuct. Ένα thread χρειάζεται το id του πελάτη το οποίο θα είναι το i + 1, τη διεύθυνση μνήμης που θα αποθηκευτεί ο χρόνος που πέρασε μέχρι να συνδεθεί με τηλεφωνητή και με ταμία, και ο συνολικός χρονος εξυπηρέτησης, και τη διεύθυνση για να ξεκινήσουμε τον χρόνο από τότε που τελείωσε η κλήση με τον τηλεφωνητή (ώστε να μετρήσουμε τον χρόνο αναμονής για σύνδεση με ταμεία). Τέλος, δημιουργούμε το thread και περνάμε τη διεύθυνση του stuct με τα arguments. Όταν όλα τα threads τελειώσουν, τυπώνονται όλα τα αποτελέσματα στην οθόνη και μετά καταστρέφουμε τα mutexes και condition variables.

connect_with_tel

Όταν καλεί ένας πελάτης (δηλαδή δημιουργείται ένα thread) κλειδώνει ένα mutex για να μπορεί να αυξομειώνει τους διαθέσιμους τηλεφωνητές και όσο δεν υπάρχουν διαθέσιμοι τηλεφωνητές περιμένει. Χρησιμοποιείται μια while όσο περιμένει για να ελέγχεται συνέχεια και να αποφύγουμε τα spurious wakeups. Όταν συνδεθεί με τηλεφωνητή (δηλαδή υπάρει τουλάχιστον ένας διαθέσιμος), τον δεσμεύει (μειώνει τους διαθέσιμους τηλεφωνητές κατά 1). Μέσω κλήσης στην συνάρτηση bernoulli_distr διαλέγει ένα zone. Αν επιστραφεί 1 τότε έχει επιλεχθεί η ζώνη B ενώ αν επιστραφεί 0 η ζώνη A (για αυτό και 1 - πιθανότητα ζώνης A). Αφού βρεθεί η ζώνη χρειάζεται να ψάξει ένα τυχαίο αρίθμο δευτερολέπτων στο διάστημα $[t_{seatlow},\ t_{seathigh}]$ για να βρεί αν υπάρχει σειρά με τόσες θέσεις όσες και τα συνολικά εισητήρια (τυχαίος αριθμός). Για να βρεθούν οι θέσεις χρησιμοποιείται ένας αλγόριθμος που χρησιμοποιεί την τεχνική του sliding window και ελέγχει κάθε σειρά αν υπάρχουν συνεχόμενες θέσεις. Επειδή το πλάνο του θεάτρου αποθηκεύεται ως matrix με τιμές bool μπορούμε απλά να δούμε αν υπάρχουν συνεχομένες θέσεις (σε μια σειρά) με άθροισμα ίσο με τον συνολικό αριθμό εισητηριών (αφού αν matrix[i][j] = 1 τότε θα είναι διαθέσιμη η θέση), στο τέλος της αναφοράς υπάρχει παράδειγμα αυτού του αλγορίθμου. Αν βρέθεί τέτοια σειρά αποθηκεύται στο info η σειρά και η που ξεκινάει η πρώτη θέση του πελάτη. Δηλαδή αν ξεκινάει στην i, τότε θα δεσμευτούν οι θέσεις i, i + 1,..., i + total tickets. Αλλίως, το info[0] και το info[1] θα είναι -1. Αν <u>δεν βρεθούν</u> θέσεις, δεσμεύεται lock για να αυξηθεί ο μετρητής των αποτυχημένων συναλλαγών λόγω μη διαθέσιμων θέσεων. Εμφανίζεται μήνυμα λάθους, αποδεσμεύεται ο τηλεφωνητής και γίνεται signal για να συνδεθεί κάποιο άλλο thread, και τέλος σταματάμε τον χρόνο αναμονής και εξυπηρέτησης, αποδεσμέουμε τον χώρο του struct που έχει τα arguments και τερματίζεται το νήμα. Αν <u>βρεθούν</u> θέσεις, δεσμεύεται ένα lock για να δεσμευτούν οι θέσεις στο πλάνο του θεάτρου (matrix),

αποδεσμεύεται ο τηλεφωνητής και γίνεται signal για να συνδεθεί κάποιο άλλο νήμα, και μετά προχωράμε στην πληρωμή (make_payment).

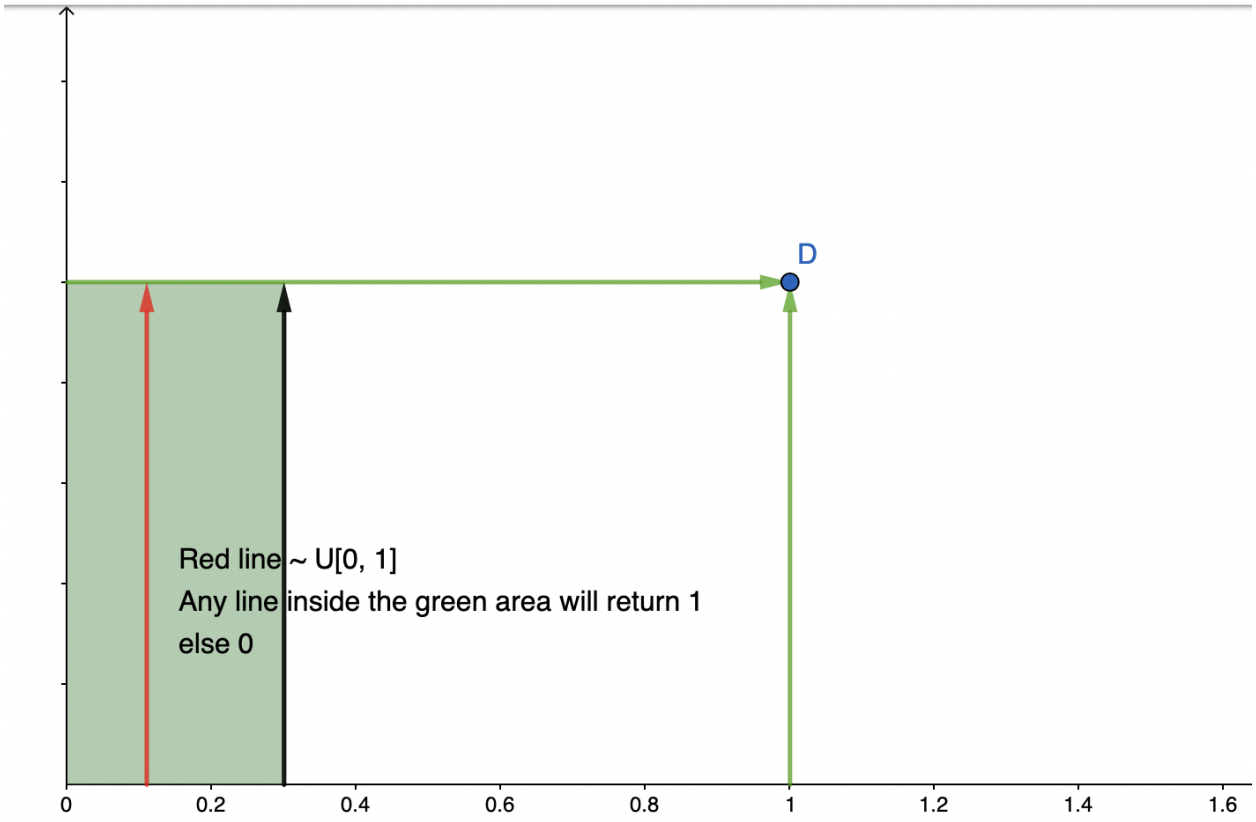
make_payment

Ξεκινάει να μετράει ο χρόνος αναμονής για σύνδεση με ταμία. Παίρνουμε lock και με χρήση while για αποφυγή spurious wakeup περίμενουμε μέχρι να βρεθεί διαθέσιμος ταμίας και όταν βρεθεί δεσμεύεται και σταματάει ο χρόνος αναμονής ταμία. Ο ταμίας χρειάζεται ένα τυχαίο αριθμό δευτερολέπτων για να προσπαθήσει την πληρωμή. Αφού τελειώσει το sleep με χρήση της συναρτήσης bernoulli_distr επιστρέφεται 1 για επιτυχή πληρωμή και 0 για μη επιτυχή. Αν αποτύχει η πληρωμή, παίρνουμε lock για να αυξήσουμε τον μετρητή για το πλήθος των αποτυχημένων πληρωμών, εμφανίζουμε μήνυμα λάθος και αφού πάρουμε lock αποδεσμέυουμε τις θέσεις από το πλάνο του θεάτρου (matrix), αποδεσμεύουμε τον ταμία και κάνουμε signal για την χρήση άλλου νήματος. Αν είναι επιτυχής η πληρωμή, παίρνουμε lock για να αυξήσουμε τον μετρητή για τις επιτυχείς συναλλαγές, βρίσκουμε το συνολικό κόστος για τις θέσεις, παίρνουμε lock για να μεταφέρουμε τα χρήματα στον τραπεζικό λογαριασμό και εμφανίζουμε μήνυμα με τη γραμμή, τις θέσεις, τη ζώνη, και το συνολικό κόστος. Τέλος αποδεσμεύουμε τον ταμία και κάνουμε signal για να χρησιμοποιηθεί από άλλο νήμα, αποδεσμεύουμε την μνήμη του struct με τα arguments της συνάρτησης και σταματάμε τον χρόνο εξυπηρέτησης και τερματίζουμε το νήμα.

bernoulli_distr

Δέχεται την πιθανότητα αποδοχής και το seed για παραγωγή τυχαίου αριθμού ο οποίος διαιρείται με το RAND_MAX για να βρίσκεται στο διάστημα [0, 1] δηλαδή παράγουμε μια

τυχαία τιμή με χρήση ομοιόμορφης κατανομής στο [0, 1]. Αν αυτή η τυχαία μεταβλητή είναι μικρότερη από το p (πιθανότητα επιτυχίας) τότε επιστρέφουμε 1 αλλιώς 0.



Πως λειτουργεί η Bernoulli Distribution με δείγμα U[0, 1]

<u>Sliding Window</u>

Total tickets = 3

Every iteration we subtract the front of the window (before moving it) and adding the last item of the window (after moving it).

1. Sliding Window Sum = 0 + 1 + 1 = 2 < 3

0	1	1	0	0	1	0	1	1	1

2. Sliding Window Sum = 2 + 0 - 0 = 2 < 3

0	1	1	0	0	1	0	1	1	1

3. Sliding Window Sum = 2 + 0 - 1 = 1 < 3

0	1	1	0	0	1	0	1	1	1

4. Slinding Window Sum = 1 + 1 - 1 = 1 < 3

0	1	1	0	0	1	0	1	1	1

5. Sliding Window Sum = 1 + 0 - 0 = 1 < 3

0	1	1	0	0	1	0	1	1	1

6. Sliding Window Sum = 1 + 1 - 0 = 2 < 3

0	1	1	0	0	1	0	1	1	1

7. Sliding Window Sum = 2 + 1 - 1 = 2 < 3

0	1	1	0	0	1	0	1	1	1

8. Sliding Window Sum = 2 + 1 - 0 = 3 == 3

0	1	1	0	0	1	0	1	1	1

Το sum == total tickets αρα βρέθηκε σειρά με 3 συνεχόμενες θέσεις

Αν δεν βρεθεί το sum = 0 και γίνεται η ίδια διαδικασία για την επόμενη σειρά.