Report 1ης προγραμματιστικής εργασίας Λειτουργικά συστήματα.

Στην πρώτη εργασία υλοποιήσαμε το σύστημα με τους τηλεφωνητές όπως αυτό ζητήθηκε από την εκφώνηση. Στόχος μας ήταν να κάνουμε το σύστημα να λειτουργεί παράλληλα όσο το δυνατόν αυτό ήταν δυνατό καθώς και να ομαδοποιήσουμε τις λειτουργίες του κώδικα έτσι ώστε να είναι ευανάγνωστος και να αποφευχθεί η πολλαπλή αντιγραφή κοινού κώδικα. Επίσης για την καλύτερη κατανόηση του χρήστη που το διαβάζει έχουμε ορίσει αρκετές σταθερές που εμείς χρησιμοποιούμε ως flags για να διακρίνουν την λειτουργικοτήτα ορισμένων μεθόδων. Τέλος αντιμετοπίσαμε ένα πρόβλημα με την συναρτηση clock\_gettime, η οποία ζητούσε ως πρώτο όρισμα την σταθερά CLOCK\_REALTIME. Για κάποιο λόγο το vm μας δεν μπορούσε με τίποτα να αναγνωρίσει αυτή την σταθερά παρόλο που προσπαθήσαμε να κάνουμε include το time.h. Για αυτό για να επιλύσουμε το πρόβλημα, ψάχνοντας στο διαδίκτυο βρήκαμε ότι αυτή η σταθερά είναι 0 και απλά το βάλαμε κατευθείαν σαν παράμετρο στην συνάρτηση.

Το πρόγραμμα ξεκινάει από την main αρχικοποιόντας ορισμένους πίνακες που χρειάζεται στην διάρκεια του, καθώς και τα mutexes και το Condition που χρειαζόμαστε και μετά ώστε να μπλοκάρουμε τα threads. Χρησιμοποιήσαμε τα εξής mutexes:

1. Operators – οπου χρησιμεύει στο να μπλοκάρει κάθε φορά την ανάγνωση και την αγγραφή της μεταβλητής που αναφέρεται στους ελεύθερους τηλεφωνητές.
2. expression\_of\_interest – όπου χρησιμεύει για την ανάγνωση και εγγραφή στον πίνακα των θέσεων του θεάτρου και βοηθάει και κατά την αναζήτηση των θέσεων αλλα και κατά την ολοκλήρωση της κράτησης.
3. Payment – όπου χρησιμεύει κατά την διαδικασία της πληρωμής για την μεταβλητή που συγκεντρώνει τα συνολικά έσοδα του θεάτρου.
4. seats\_availability – χρησιμεύει κατά την επιτυχημένη ολοκλήρωση της συναλλαγής για να αλλάξει την τιμή του counter των ελεύθερων θέσεων.
5. Condition – το οποίο χρησιμεύει κατά την αναμονή των χρηστών για τους τηλεφωνητές ώστε να μην μπλοκάρει το πρόγραμμα.

Επόμενο βήμα είναι να αρχικοποιηθούν όλα τα threads, που εξικονίζουν τα τηλεφωνήματα ώστε να ξεκινήσουν οι διαδικασίες των κρατήσεων. Σε αυτό το κομμάτι του κώδικα θα βρείτε σε σχόλιο μια εντολή sleep(1) ώστε να προσομοιώσουμε ότι δεν ήρθαν όλα τα τηλεφωνήματα αμέσως. Τέλος στην main περιμένουμε τα threads να τελειώσον και ελευθερώνουμε την μνήμη των threads, των mutexes και του condition που χρησιμοποιήσαμε.

Στο κώδικα μας θα βρείτε τις εξής μεθόδους:

1. void mutex\_handle(pthread\_mutex\_t\* mutex, int flag) – Αυτή χρησιμεύει για οποιαδήποτε διαδικασία θα πραγματοποιήσουμε με τα mutexes βάση του κατάλληλου flag που θα περάσουμε ως όρισμα. Δηλαδή χρησιμεύει για το init, lock, unlock και destroy.
2. void condition\_handle (pthread\_cond\_t\* cond, int flag) – Αυτή χρησιμεύει για οποιαδήποτε διαδικασία θα πραγματοποιήσουμε με το condition βάση του κατάλληλου flag που θα περάσουμε ως όρισμα. Δηλαδή χρησιμεύει για το init και destroy.
3. int rand\_prob(float percent) – Αυτή χρησιμεύει για την έρευση πιθανότητας για την επιτυχία της πληρωμής και επιστρέφει SUCCESS με πιθανότητα percent
4. void wait\_operator(TRANSACTION\_INFO\* info) – Αυτή χρησιμεύει για την διαδικασία όπου το σύστημα αναζητάει διαθέσιμο τηλεφωνητή για τους πελάτες που καλούν. Αναλαμβάνει να κάνει lock και unlock το κατάλληλο Mutex καθώς και να το ελευθερώσει σε περίπτωση που δεν υπάρχει διαθέσιμος τηλεφωνητης μέχρι να λάβει σήμα από το πρόγραμμα ότι τελείωσε ένα transaction.
5. int request\_seats(int\* clientID) – Αυτή χρησιμεύει για την προσομοίωση της ομιλίας του τηλεφωνητή με τον πελάτη για να του ζητήσει πόσες θέσεις θέλει. Τέλος επιστρέφει στην μέθοδο transaction τον αριθμό των θέσεων που ζήτησε το πελάτης.
6. int find\_seats (TRANSACTION\_INFO\* info) – Αυτή χρησιμεύει για την προσομοίωση της αναζήτησης των θέσεων από τον τηλεφωνητή. Μόλις βρεθούν οι κατάλληλες θέσεις το σύστημα τις μαρκάρει με SEAT\_OCCUPIED ώστε να μην είναι διαθέσιμες από άλλους τηλεφωνητές. Αν τελικά δεν συμπληρωθούν οι κατάλληλες θέσεις που θέλει ο πελάτης τότε ξαναελευθερώνει τις θέσεις που έκανε occupied. Τέλος επιστρέφει στην μέθοδο transaction ένα flag για το αν πέτυχε η διαδικασία.
7. int pay\_seats(int amount) – Αυτή χρησιμεύει για την προσομοίοση της διαδικασίας πληρωμής από τον χρήστη και την ανανέωση της μεταβλητής που κρατάει τα κεντρικά έσοδα του θεάτρου. Επιστρέφει στην μέθοδο transaction ένα flag για το αν πέτυχε ή όχι η κράτηση.
8. void change\_seats\_state (int new\_state, TRANSACTION\_INFO\* info) – Αυτή χρησιμεύει ώστε στο τέλος της κράτησης αν αποτύχει να ελευθερώνει πάλι τις θέσεις που έγιναν occupied είτε πλέον να βάζει μέσα το id του πελάτη δείχνοντας ότι αυτός έχει κλήσει την θέση.
9. void change\_seats\_availability (TRANSACTION\_INFO\* info) – Αυτή χρησιμεύει στο τέλος της κράτησης αν αυτή είναι επιτυχημένη να ανανεώνει έναν counter που έχουμε ο οποίος μετράει τις διαθέσιμες θέσει του θεάτρου.
10. void\* transaction (void\* clientID) – Αυτή είναι κεντρική μέθοδος όπου καλεί το thread και αναλαμβάνει την διαδικασία του τηλεφωνήματος. Μαρκάρει τους εξής χρόνους: όταν ξεκινάει το transaction, όταν συνδέεται ο χρήστης με κάποιον τηλεφωνητή και όταν τελειώνει το transaction. Αυτά γίνονται για να υπολογιστούν τα στατιστικά όπως τα ζητάει το πρόγραμμα. Αναλαμβάνει να καλέσει όλες τις επιμέρους μεθόδους που έχουν να κάνουν με το τηλεφώνημα και διαθέτει ένα struct TRANSACTION\_INFO το οποίο χρησιμοποιείται από αρκετές επιμέρους μεθόδους και κρατάει τις πληροφορίες της εκάστοτε κράτησης. Στο τέλος με το που τελειώσει ένα transaction κάνει broadcast ένα μήνυμα ώστε όσοι είναι σε αναμονή για τηλεφωνητή να μπορέσουν να προχωρήσουν.
11. void arguments\_check(int argc, char\* argv[]) – Αυτή χρησιμεύει για την ανάγνωση των arguments το console όταν ξεκινάει το πρόγραμμα.