

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΝΙΚΟΛΑΣ ΚΟΥΝΤΟΥΡΙΩΤΗΣ 3170195

ΣΑΜΑΡΤΖΗΣ ΓΕΩΡΓΙΟΣ 3180168

ΗΛΙΑΣ ΑΠΟΣΤΟΛΑΚΗΣ 3180012

Δομή Κώδικα & Ροή Προγράμματος :

Ξεκινάμε με το header file , όπου γίνονται οι δηλώσεις των μεταβλητών που θα χρησιμοποιεί το C file μας . Δηλώνουμε τις μεταβλητές και μετά δηλώνουμε τις σταθερές όπως γράφει η εκφώνηση . Από κάτω δηλώνουμε τις mutex & cond μεταβλητές και καταστάσεις , όπου θα χρειάζονται για να ελέγχουμε τους πόρους που θα επεξεργάζονται τα threads , έτσι ώστε να μην υπάρχουν συγκρούσεις και τυχαίες μεταβολές στις τιμές τους . Υπάρχουν συνολικά 12 mutex & 5 cond μεταβλητές .

Περνώντας στο C file , κάνουμε import το header file και αμέσως μετά αρχικοποιούμε στο global scope τις μεταβλητές που θα χειρίζονται τα threads . Μετά περνάμε στη main όπου αρχικά αρχικοποιούμε τις υπόλοιπες μεταβλητές , γίνεται ο έλεγχος για σωστές παραμέτρους και μετά τις περνάμε σε μεταβλητές . Μετά αρχικοποιούμε ένα order id array για να κρατάμε τα id των threads , κάνουμε allocate την ανάλογη μνήμη , δηλαδή για 100 πελάτες , και μετά περνάμε στην δημιουργία τους μέσω ενός for loop και μετά τα κάνουμε join . Περνάμε στην συνάρτηση pthread_create την συνάρτηση startOrder και το συγκεκριμένο order id , όπου εκεί διαχειρίζεται το thread .

Αφού το πρόγραμμα μπει στην συνάρτηση startOrder τότε αρχικοποιεί τους χρόνους που θα χρειαστούμε , ένα μοναδικό σπόρο για κάθε thread και περνάει την μεταβλητή order id σε μια τοπική μεταβλητή . Ξεκινάει ο χρόνος και αφού πάντα γίνονται τα σωστά lock / unlock mutex και cond wait και signal , πραγματοποιείται η παραγγελία του χρήστη . Πριν να γίνει η οποιαδήποτε μετατροπή σε κάποιο πόρο γίνεται lock και αλλάζει η τιμή του και πριν γίνει unlock , ξαναγίνεται διαθέσιμη , δηλαδή αυξομειώνεται κατά 1 με κάθε διαδικασία . Όταν δεν υπάρχουν διαθέσιμοι τηλεφωνητές , φούρνοι , μάγειρες , διανομείς , μπαίνει στο ανάλογο while loop όπου περιμένει το ανάλογο condition να αλλάξει , μέσω signals . Για τον σκοπο της αποτυχίας της συναλλαγής δημιουργούμε μια τυχαία μεταβλητή απο το 1-100 και ελέγχουμε αν η συγκεκριμένη τιμή είναι μικρότερη από το 5 και αν αυτο ισχύει τερματίζει το thread αφού πρώτα βγάλει το σωστό μήνυμα με lock / unlock lock output (screen lock) .

Η φόρμουλα στην οποία πραγματοποιείται η τυχαιοποίηση των αριθμών έχει ως εξής :

$(rand_r(&seedp) \% (high - low + 1)) + low;$

Έτσι μπορούμε να μένουμε πάντα στα προκαθορισμένα όρια που δίνονται στην εκφώνηση λαμβάνοντας υπόψη τα ανάλογα άνω και κάτω όρια (delivery time , pizza bake time , delivery time etc.) .

Το πρόγραμμα κάνει sleep τις στιγμές που χρειάζεται ανάλογα με τις τιμές στο header file ή τις τιμές που υπολογίσαμε με την παραπάνω φόρμουλα , σε δευτερόλεπτα , αποτέλεσμα να λειτουργεί κανονικά το πρόγραμμα .

Για τον υπολογισμό των τιμών των που έχουν να κανουν με το timespec αφαιρούμε την μία με την άλλη για να φανούν οι σωστοί χρόνοι , όπως έτσι συμβουλευεται στη περιγραφή της <time.h> .

Για όλες τις εξόδους του προγράμματος υλοποιείται η `lock_output mutex` για τον έλεγχο της ροής εξόδου . Για όλα τα `mutex & cond` συνθήκες ελέγχουμε μέσω της `rc` για τυχόν σφάλμα και αν βρεθεί κάνει `exit` απο το `thread` και δίνει μήνυμα σφάλματος :

Mutex: Error %d \n, rc;

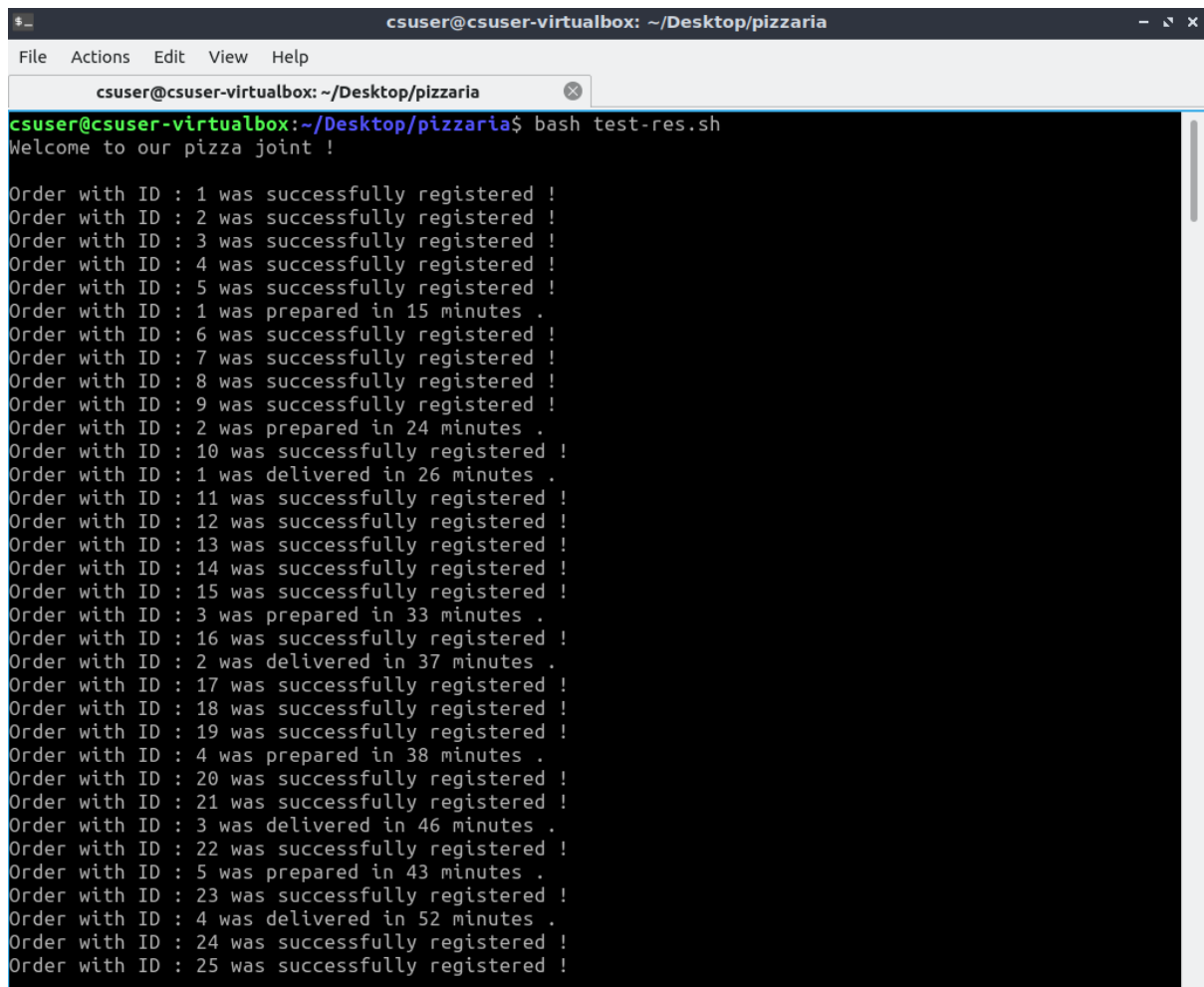
Αφού τυπώσει τα πρώτα 2 μηνύματα σε ανάλογο χρόνο πως η παραγγελία απέτυχε / καταχωρήθηκε επιτυχώς , και πως ετοιμάστηκε , μετα μόλις διεκπαιρωθεί η παραγγελία τυπώνει το ανάλογο μήνυμα πως παραδόθηκε στα συγκεκριμένα λεπτά από την ώρα που άρχισε (το πρόγραμμα) , ελέγχει για `max` τιμές και κάνει `pthread_exit(idOfOrder)` .

Αφού τελειώσουν όλα τα `threads` υπολογίζουμε τις `average` τιμές και τις προβάλλουμε μόλις τελειώσουν οι διαδικασίες όλων των `threads` , και μετά αποδεσμεύουμε τη μνήμη που αρχικά ορίσαμε για τα `threads` και επιστρέφουμε την τιμή 1 για να λήξει το πρόγραμμα .

Υπάρχει και το `.sh` file όπου το τρέχουμε με την εντολή `bash test-res.sh` , και έχει μέσα τον κώδικα , :

```
gcc -pthread -o output 3170195-3180168-3180012.c  
./output 100 1000
```

Έτσι θα τρέξει το πρόγραμμα όπως ζητά η εκφώνηση με 100 πελάτες και 1000 για αριθμό τυχαίου σπόρου .Screenshot του προγράμματος όταν τρέξαμε το `bash test-res.sh` :



```
csuser@csuser-virtualbox: ~/Desktop/pizzaria
File Actions Edit View Help
csuser@csuser-virtualbox: ~/Desktop/pizzaria
csuser@csuser-virtualbox:~/Desktop/pizzaria$ bash test-res.sh
Welcome to our pizza joint !

Order with ID : 1 was successfully registered !
Order with ID : 2 was successfully registered !
Order with ID : 3 was successfully registered !
Order with ID : 4 was successfully registered !
Order with ID : 5 was successfully registered !
Order with ID : 1 was prepared in 15 minutes .
Order with ID : 6 was successfully registered !
Order with ID : 7 was successfully registered !
Order with ID : 8 was successfully registered !
Order with ID : 9 was successfully registered !
Order with ID : 2 was prepared in 24 minutes .
Order with ID : 10 was successfully registered !
Order with ID : 1 was delivered in 26 minutes .
Order with ID : 11 was successfully registered !
Order with ID : 12 was successfully registered !
Order with ID : 13 was successfully registered !
Order with ID : 14 was successfully registered !
Order with ID : 15 was successfully registered !
Order with ID : 3 was prepared in 33 minutes .
Order with ID : 16 was successfully registered !
Order with ID : 2 was delivered in 37 minutes .
Order with ID : 17 was successfully registered !
Order with ID : 18 was successfully registered !
Order with ID : 19 was successfully registered !
Order with ID : 4 was prepared in 38 minutes .
Order with ID : 20 was successfully registered !
Order with ID : 21 was successfully registered !
Order with ID : 3 was delivered in 46 minutes .
Order with ID : 22 was successfully registered !
Order with ID : 5 was prepared in 43 minutes .
Order with ID : 23 was successfully registered !
Order with ID : 4 was delivered in 52 minutes .
Order with ID : 24 was successfully registered !
Order with ID : 25 was successfully registered !
```

```
Order with ID : 18 was prepared in 136 minutes .  
Order with ID : 63 was successfully registered !  
Order with ID : 16 was delivered in 146 minutes .  
Order with ID : 64 was successfully registered !  
Order with ID : 18 was delivered in 142 minutes .  
Order with ID : 65 has failed !  
Order with ID : 19 was prepared in 142 minutes .  
Order with ID : 66 was successfully registered !  
Order with ID : 20 was prepared in 141 minutes .  
Order with ID : 17 was delivered in 151 minutes .  
Order with ID : 67 was successfully registered !  
Order with ID : 19 was delivered in 150 minutes .
```

```
Order with ID : 100 was prepared in 474 minutes .  
Order with ID : 100 was delivered in 486 minutes .
```

```
Total income from pizzas is : 2720.00  
Number of failed payments is : 4  
Number of successful payments is : 96
```

```
Average waiting time is 1.560000  
Max waiting time is 4.000000
```

```
Average service time is 224.650000  
Max service time is 486.000000
```

```
Average cold time is 9.840000  
Max cold time is 15.000000
```

```
csuser@csuser-virtualbox:~/Desktop/pizzeria$
```

-----EYXAPISTOYME-----