ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

2022-2023

Αναφορά εργασίας

C file:
Στην αρχή του αρχείου γίνονται οι δηλωσεις όλων των καθολικών μεταβλητών και των mutexes, conds.

Υλοποίηση βοηθητικών συναρτήσεων:

- void rc_check(int rc)
  Γίνεται έλεγχος στο response code. Αν μια πράξη στα νήματα/mutexes/conds επιστρέψει response code διαφορετικό απο 0, τότε εκτυπώνεται μήνυμα λάθους, διαγράφονται τα mutexes και conds που είχαν δημιουργηθεί και τερματίζει το πρόγραμμα.

- void mutex_destroyer(pthread_mutex_t *mutex)
  Διαγράφεται το mutex και γίνεται έλεγχος στο response code του. Αν υπάρξει σφάλμα, τότε εκτυπώνεται μήνυμα λάθους και δεν τερματίζεται το πρόγραμμα για να συνεχίσει η καταστροφή των υπόλοιπων mutexes.
- void cond_destroyer(pthread_cond_t *cond)
  Διαγράφεται το cond και γίνεται έλεγχος στο response code του. Αν υπάρξει σφάλμα, τότε εκτυπώνεται μήνυμα λάθους και δεν τερματίζεται το πρόγραμμα για να συνεχίσει η καταστροφή των υπόλοιπων conds.
- void mutex_lock(pthread_mutex_t *mutex)
  Κλειδώνει το mutex και γίνεται ο έλεγχος στο rc.

- void mutex_unlock(pthread_mutex_t *mutex)
  Ξεκλειδώνει το mutex και γίνεται ο έλεγχος στο rc.
- int initializations()
  Initializations των mutexes, conds κάνοντας τους απαραίτητους ελέγχους και διαγράφοντας τα mutexes, conds που δημιουργήθηκαν προηγουμένως σε περίπτωση σφάλματος. Σε περίπτωση σφάλματος επιστρέφει 1, αλλιώς 0.
- void destructor()
  Destruction των mutexes, conds.

Main:
Στην αρχή γίνεται έλεγχος εγκυρότητας των δεδομένων εισόδου (πλήθος παραμέτρων, αριθμός πελατών) και η μεταροπή των δεδομένων εισόδου σε ακέραιους αριθμούς.
Αν τα δεδομένα είναι έγκυρα τότε καλείται η μέθοδος initializations που δημιουργεί τα mutex και τα cond.
Αν δημιουργηθεί πρόβλημα στο initialization κάποιου από τα προαναφερθέντα, δηλαδή επιστραφεί 1 από την μέθοδο, τότε τερματίζεται το πρόγραμμα.
Αν δεν υπήρξε πρόβλημα στην παραπάνω διαδικασία επιστρέφεται τιμή 0 και συνεχίζει η εκτέλεση της main.
Δημιουργείται πίνακας για την αρίθμηση των threads (παραγγελίες)
Ακολουθεί ένα for loop στο οποίο γίνεται ανάθεση ενός μοναδικού order Id για κάθε πελάτη-νήμα και δημιουργείται το thread της εκάστοτε παραγγελίας με διεξαγωγή αντίστοιχου ελέγχου σε περίπτωση σφάλματος. Ύστερα εκτελείται η order με παράμετρο το orderID. Μετά

τον πρώτο πελάτη, κάθε νήμα-πελάτης περιμένει ένα τυχαίο χρονικό διάστημα πριν δημιουργηθεί. Ο χρόνος καθορίζεται από τη συνάρτηση rand_r με σπόρο το σπόρο που έχει δοθεί με βάση την είσοδο αθροιστικά με το id της παραγγελίας. Αυτό συμβαίνει επειδή αν δινόταν ο ίδιος σπόρος κάθε φορά θα λαμβάναμε συνεχώς τα ίδια αποτελέσματα. Ύστερα, γίνεται το join των νημάτων(όταν αυτά τερματίζουν) ξανά με τον αντίστοιχο έλεγχο. Σε περίπτωση αποτυχίας τα mutexes και τα conditions που έχουν δημιουργηθεί καταστρέφονται, γίνεται pthread_exit(NULL) και τερματίζει το πρόγραμμα.
Στο τέλος, εκτυπώνονται τα μηνύματα που ζητούνται από την εκφώνηση, καταστρέφονται τα mutexes και conds και τερματίζει το πρόγραμμα.

Order:
Η order είναι η συνάρτηση που καλείται κατά την δημιουργία των νημάτων και εκτελείται για κάθε νήμα με παράμετρο το OrderID.
Σε αυτή την συνάρτηση γίνονται όλες οι διεργασίες που απαιτεί μια παραγγελία.
Αρχικά στη μεταβλητή OrderID αναθέτουμε το id της κάθε παραγγελίας που περνάει σαν παράμετρος και με βάση αυτό υπολογίζουμε το τοπικό σπόρο (local_seed) που είναι μοναδικός για κάθε νήμα. Ύστερα, γίνονται οι αρχικοποιήσεις των αριθμών από πίτσες που παρήγγειλε ο κάθε πελάτης, καθώς και το είδος της κάθε πίτσας.
Ακολουθεί ο υπολογισμός του κόστους της παραγγελίας και η προσπάθεια πληρωμής. Για την πληρωμή ο πελάτης περιμένει (sleep) 1-3 secs για να δεχτεί την κάρτα. Αν η πληρωμή αποτύχει τότε αυξάνονται οι αποτυχημένες παραγγελίες κατά ένα και κάνουμε exit από το thread. Αν είναι επιτυχημένη τότε συνεχίζεται η διαδιασία της παραγγελίας και ανανεώνουμε τα στατιστικά στοιχεία.
Για την ενημέρωση των στατιστικών στοιχείων, καθώς και για τις αποτυχημένες/πετυχημένες παραγγελίες, χρησιμοποιούμε mutex (operator_lock, fail_lock) γιατί οι μεταβλητές που χρησιμοποιούμε είναι global και θέλουμε να τις ενημερώνει ένα νήμα κάθε φορά (το νήμα ζητά/έχει αποκλειστική πρόσβαση στις μεταβλητές).
Έπειτα ακολουθεί το ψήσιμο , το πακετάρισμα και το delivery.

Επόμενο στάδιο μετά την επιτυχημένη πληρωμή είναι το ψήσιμο των πιτσών.
Με την χρήση mutex και condition μεταβλητών, κάθε πελάτης περιμένει μέχρι να βρει διαθέσιμο ψήστη. Με τη χρήση του while loop, όσο δεν υπάρχει διαθέσιμος ψήστης η παραγγελία μπαίνει σε αναμονή και περιμένει (pthread_cond_wait) μέχρι να σταλθεί σήμα από άλλο νήμα που τελείωσε με τη διαδικασία του ψήστη, ότι βρέθηκε διαθέσιμος ψήστης. Όταν βρεθεί διαθέσιμος ψήστης τότε, μειώνονται οι διαθέσιμοι ψήστες (n_available_cooks) κατά 1.
Έπειτα, η παραγελία χρειάζεται Tprep * number_of_pizzas δευτερόλεπτα να ετοιμαστεί από τον ψήστη. Μόλις ολοκληρωθεί η προετοιμασία περνάμε στο επόμενο στάδιο, που είναι οι φούρνοι, δηλαδή το ψήσιμο των πιτσών.

Αντίστοιχα (με mutexes/conds) ο ψήστης περιμένει μέχρι να βρει αρκετούς φούρνους διαθέσιμους, έτσι ώστε να βάλει όλες τις πίτσες σε κάθε φούρνο και να ψηθούν παράλληλα. Όταν βρεθούν όλοι οι διαθέσιμοι φούρνοι, τότε ο ψήστης τις τοποθετεί στους φούρνους και συνεχίζει με την επόμενη παραγγελία. Τότε είναι που απελευθερώνεται ο ψήστης, χρησιμοποιώντας το κατάλληλο mutex, αυξάνεται ο αριθμός των διαθέσιμων ψηστών κατά 1 και στέλνεται σήμα σε κάποιο νήμα ότι βρέθηκε διαθέσιμος ψήστης.

Οι πίτσες ψήνονται (sleep) για 10 δευτερόλεπτα (Tbake).

Όμοια γίνονται και οι διαδικασίες του πακεταρίσματος και του delivery.
Υποθέτουμε ότι τη στιγμή που βρεθεί υπάλληλος για πακετάρισμα, αφού ετοιμάσει τις παραγγελίες για παράδοση (sleep Tprep*number_of_pizzas sec) δεν τις παραδίδει ο ίδιος στον ντελιβερά, αλλά τις αφήνει (π.χ. σε κάποιο πάγκο του καταστήματος) και τις παίρνει ο επόμενος διαθέσιμος ντελιβεράς.

Μόλις βρεθεί διαθέσιμος διανομέας, παραδίδει την παραγγελία (sleep) σε ένα τυχαίο χρονικό διάστημα 5-15 δευτερόλεπτα. Ύστερα, περιμένουμε το ίδιο τυχαίο χρονικό διάστημα να επιστρέψει (sleep) ο διανομέας στο κατάστημα, και τότε τον αποδεσμεύουμε.
Τέλος, με τη χρήση mutex, ενημερώνουμε τις global μεταβλητές για τους χρόνους και καλείται η pthread_exit για σωστό τερματισμό του νήματος.

Στο αρχείο ***pizzeria.h*** δηλώνονται οι σταθερές του προγράμματος μας σύμφωνα με την εκφώνηση, οι μέθοδοι που θα υλοποιήσουμε και θα χρησιμοποιήσουμε, και εισάγωνται όλες οι απαραίτητες βιβλιοθήκες.

Στο ***results.txt*** εμφανίζονται τα αποτελέσματα της εκτέλεσης του προγράμματος μας με είσοδο 1000 σπόρο 100 πελάτες.

Στο αρχείο ***test-res.sh*** αναγράφονται οι εντολές που χρησιμοποιήθηκαν για την μεταγλώττιση και εκτέλεση του προγράμματος με συνολικούς πελάτες 100 και αρχικό σπόρο 1000.