Λειτουργικά Συστήματα

Προγραμματιστική Εργασία – Εαρινό Εξάμηνο 2020-2021

Ον/μο: Βιτάλης Ιωάννης – ΑΜ: 3150011

Ον/μο: Δαφνομήλης Νικόλαος – ΑΜ: 3140038

Ον/μο: Κατσαρέλης Κωνσταντίνος – ΑΜ: 3170065

Περιγραφή

Για την προγραμματιστική εργασία στο μάθημα των Λειτουργικών Συστημάτων μας ζητήθηκε η υλοποίηση ένος προγράμματος στην γλώσσα προγραμματισμού C με χρήση του πακέτου νημάτων POSIX.

Η εργασία αφορά την προσομοίωση της λειτουργίας μιας πιτσαρίας από την στιγμή που ένας πελάτης καλεί το κατάστημα μέχρι και την στιγμή που παραδίδετε η παραγγελία του. Η πιτσαρία διαθέτει έναν περιορισμένο αριθμό τηλεφωνητών, ψηστών, φούρνων, ατόμων που πακετάρουν τις πίτσες και οδηγών.

Η χρήση threads, mutexes και conditions του πακέτου POSIX μας επιτρέπει να συγχρονίσουμε τις δουλειές που κάνει ο καθένας έτσι ώστε ολες οι παραγγελίες να παραδωθούν το συντομότερο δυνατό.

Δομή Κώδικα

Χρησιμοποιήσαμε το αρχείο header για να δηλώσουμε και να αρχικοποιήσουμε όλες τις σταθερές και μεταβλητές που μας δίνονται από την εκφώνηση, να δηλώσουμε τα Mutexes και τα conditions που θα χρησιμοποιήσουμε καθώς και τις 2 συναρτήσεις που χρειάστηκε να υλοποιήσουμε.

Για την ορθή εκτέλεση του αρχείου κώδικα .c χρειαζόμαστε τουλάχιστον 2 arguments, το $1^{ο}$ είναι ο αριθμός των πελατών που θα εξυπηρετήσει η πιτσαρία και το $2^{ο}$ είναι ένα seed που θα χρησιμοποιηθεί για την ως σπόρος για την γεννήτρια τυχαίων αριθμών, αν δωθούν λιγότερα από δύο arguments το πρόγραμμα εμφάνιζει στην οθόνη κατάλληλο μήνυμα και τερματίζει, αν δωθούν περισσότερα από 2 arguments το πρόγραμμα χρησιμοποιεί τα δύο πρώτα και αγνοεί τα υπόλοιπα.

Χρησιμοποιησάμε malloc για να δεσμεύσουμε δυναμικά μνήμη και να δημιουργήσουμε πίνακα για τα threads και τον πίνακα order_id για την αποθήκευση του αριθμού της παραγγελίας, έπειτα αρχικοποιήσαμε όλα τα mutexes και τα

conditions που δηλώσαμε στο header, στην αρχική τους κατάσταση τα Mutex είναι unlocked.

Για κάθε πελάτη δημιουργούμε ένα καινούριο thread καλώντας την pthread_create με ορίσματα τους δύο πίνακες που δημιουργήσαμε και την συνάρτηση pizza_thread. Κάθε φορά που δημιουργείτε μία παραγγελία η κύρια διεργασία καλεί την sleep για ένα τυχαίο χρονικό διάστημα στην περιοχή [Torderlow,Torderhigh] η οποία σηματοδότει το πόσος χρόνος χρειάζεται για να καλέσει ο επόμενος πελάτης.

Μετά χρησιμοποιήσαμε ένα for loop για να διατρέξουμε τον πίνακα με τα threads και να καλέσουμε την pthread_join() σε κάθε thread, η pthread_join() περιμένει για το αντίστοιχο thread να τερματιστεί άρα χρησιμοποιείται για να βεβαίωθουμε οτι η main δεν θα τερματίσει πριν ολοκληρώσουν όλα τα νήματα τις εργασίες τους.

Πριν τερματίσει το πρόγραμμα τυπώνουμε όλους τους χρόνους που μας ζητήθηκαν, καταστρέφουμε ολα τα mutexes και τα conditions και ελευθερώνουμε την δυναμικά δεσμευμένη μνήμη των πινάκων threadPool και order_id.

Συνάρτηση check_rc:

Οι pthread_mutex_lock(), pthread_mutex_unlock(), pthread_cond_signal(), pthread_cond_broadcast() και pthread_cond_wait() επιστρέφουν την τιμή 0 αν είναι πετυχημένες, αλλιώς επιστρέφεται ένας αριθμός σφάλματος που υποδεικνύει το σφάλμα.

Αποθηκεύουμε τον αριθμό που επιστρέφουν σε μια μεταβλητή rc και καλούμε την check_rc(rc) για να ελέγξουμε αν υπάρχει σφάλμα, σε περίπτωση σφάλματος εμφανίζεται αντίστοιχο μήνυμα και το thread τερματίζεται.

Συνάρτηση pizza_thread:

Κάθε νήμα που δημιουργείται χρησιμοποιεί την pizza_thread για να εκτελέσει της εξής λειτουργίες:

Αρχικά δημιουργεί τις απαραίτητες τοπικές μεταβλητές που θα χρειαστούν, μία από αυτές η seed_thr προκύπτει από το seed που δώθηκε στα arguments επί το ID της παραγγελίας για την σωστή παραγωγή των ψευδοτυχαίων και για να μην έχει κάθε νήμα την ίδια ακολουθία.

Κρατάμε την χρονική στιγμή που ξεκίνησε η παραγγελία και κλειδώνουμε το νήμα με την χρήση της pthread_mutex_lock και ελέγχουμε αν υπάρχει διαθέσιμος τηλεφωνητής, αν δεν υπάρχει χρησιμοποιούμε ένα while loop με την pthread_cond_wait η οποία περιμένει μέχρι κάποιος τηλεφωνητής να γίνει διαθέσιμος,

μόλις ο τηλεφωνητής γίνει διαθέσιμος, ο αριθμός των τηλεφωνητών μειώνεται κατα 1, ξεκλειδώνει το νήμα και χρησιμοποιούμε την sleep για να προσομοιώσουμε τον χρόνο που χρειάζεται για να χρεώσει ο τηλεφωνητής την πιστώτικη κάρτα. Μόλις ολοκληρωθεί η sleep ξανακλειδώνουμε το νήμα, απελευθερώνουμε τον τηλεφωνητή αυξάνοντας τον αριθμό των τηλεφωνητών κατα 1 και με την χρήση της pthread_cond_signal στέλνουμε σήμα σε ένα άλλο κλειδωμένο thread που περιμένει κάποιον τηλεφωνητή να γίνει διαθέσιμος.

Ξεκλειδώνουμε τους τηλεφωνητές και υπολογίζουμε την πιθανότητα η παραγγελία να απέτυχε, αν απέτυχε εμφανίζεται κατάλληλο μήνυμα και η παραγγελία τερματίζει.

Παρόμοια διαδικασία ακολουθούμε για τους παρασκευαστές, τους φούρνους, τον μοναδικό υπάλληλο που πακετάρει τις παραγγελίες και τον οδηγό.

Σημαντική διαφορά παρατηρείται κάθε φορά που οι φούρνοι τελειώνουν το ψήσιμο και απελευθερώνεται αριθμός φούρνος ίσο με τον αριθμό από πίτσες που ψήθηκαν. Σε αυτή την περίπτωση δεν χρησιμοποιούμε την pthread_cond_signal() για να ειδοποιήσουμε ένα νήμα οτι υπάρχουν διαθέσιμοι φούρνοι, χρησιμοποιούμε την pthread_cond_broadcast() για να ειδοποιήσουμε όλα τα νήματα για να ελέγξουν αν υπάρχουν αρκετοί φούρνοι για την διεκπεραίωση της παραγγελίας τους.

Οταν η παραγγελία φτάσει στον πελάτη, κλειδώνουμε το mutex για να ενημερώσουμε όλες τις μεταβλητές που κρατάνε τους απαραίτητους χρόνους τους οποίους θα εμφανίσουμε στην οθόνη.

Ξεκλειδώνουμε το mutex και χρησιμοποιούμε το sleep άλλη μια φορά για τον χρόνο που χρειάζεται ο οδηγός να επιστρέψει στο μαγάζι, αυξάνουμε τον αριθμό των οδηγών κατα 1 και χρησιμοποιούμε την pthread_cond_signal(), έπειτα το thread μας τερματίζει.