Εργασία Λειτουργικά Συστήματα

Ονοματεπώνυμο: Καββαδάς Δημήτριος

AM:3190064

Αρχείο p3190064-pizza.h:

Συμπεριλαμβάνει όλες τις constant μεταβλητές που χρειάζονται και που ζητούνται από την εκφώνηση της άσκησης(αριθμό resources, τυχαίοι και μη χρόνοι για διεργασίες).

Επίσης, περιλαμβάνει επιπρόσθετες μεταβλητές όπως μετρητές για τον αριθμό των resources(cooks,ovens etc.) ,τον αριθμό των πετυχημένων και μη παραγγελιών ,τα έσοδα του μαγαζιού και όλα τα timers(max ,average και χρόνοι αναμονής etc. για κάθε παραγγελία ξεχωριστά).

Το header file περιέχει και τα mutexes και conditions για την αναγκαία δέσμευση και αποδέσμευση πόρων αλλά και για την εμφάνιση τελικών αποτελεσμάτων και για τα update των timers.

Τέλος ,εκεί βρίσκονται 2 functions την void

successful_mutex_action(int rs);

και την unsigned int sleep(unsigned int seconds); όπου χρησιμοποιούνται και οι 2 στο .c file πολύ.

Η πρώτη ελέγχει αν ένα mutex ενεργεί σωστά και η δεύτερη απλώς κάνει sleep συγκεκριμένο χρόνο που ζητάμε κάθε φορά αναλόγως την διεργασία που γίνεται(baking ,delivering etc.)

p3190064-pizza.c:

Έχει την function που γίνεται ένα order και την main.

void *order(void * order_id){…}:

Εδώ υλοποιείται κάθε φορά ένα order. Κάθε φορά αρχικοποιώ το rs και το id του order.

Έπειτα για κάθε παραγγελία γίνονται τα κατάλληλα lock και unlock στα mutexes για δέσμευση και αποδέσμευση πόρων.

Επίσης υπάρχουν κατάλληλες while οι οποίες ελέγχουν αν υπάρχουν συγκεκριμένα resources έτσι ώστε να υλοποιούνται οι παραγγελίες ταυτόχρονα ,όμως να σταματάνε κάθε φορά όταν δεν υπάρχουν πόροι. Γι'αυτό χρησιμοποιώ την cond_wait.

Απαραίτητο είναι να καλείται η συνάρτηση sleep() μετά από κάθε διεργασία μίας παραγγελίας(bake etc.).

Επίσης χρειάζονται και structs timespecs kai clock_gettime() για τα average και max timers αλλά για να υπολογιστούν αυτές οι τιμές χρειάζεται να υπολογίζω τους χρόνους για κάθε παραγγελία ξεχωριστά.

Δηλαδή π.χ struct timespec order_start; struct timespec order_finish; rs=clock_gettime(CLOCK_REALTIME, &order_start); rs=clock_gettime(CLOCK_REALTIME, &order_finish); waiting_time = order_finish.tv_sec - order_start.tv_sec;

Ενώ το ίδιο κάνω σχεδόν σε διαφορετικά σημεία για άλλους χρόνους.

Επίσης χρησιμοποιώ την rand_r(int *seed); Όπου seed δίνεται από τον χρήστη για του τυχαίους χρόνους etc. .Τέλος κάνω print κάθε πληροφορία που χρειάζομαι , και κάνω exit το pthread.

int main(int argc, char *argv[]){}:

Αρχικά ,τοποθετώ τα 2 input του χρήστη για τον αριθμό των πελατών και το seed σε δύο μεταβλητές καθώς ελέγχω αν τα input είναι σωστά.

Έπειτα ,αρχικοποιώ όλες τις μεταβλητές που έχω δηλώσει από το .h file.

Βάζω σε ένα array τους το id του κάθε πελάτη σε αύξων αριθμό ξεκινώντας από το 1. Σε μία for φτιάχνω thread για κάθε παραγγελία οπότε καλείται και η παραπάνω συνάρτηση όσες φόρες αναλόγως με τους πελάτες που έχει δώσει ο χρήστης ως input.

Κάνω join threads και περιμένω να τελειώσουν όλες οι διεργασίες για το καθένα από αυτά.

Επίσης ,εμφανίζω όλα τα τελικά αποτελέσματα που ζητούνται από την εκφώνηση της άσκησης(incomings etc.).

Τέλος διαγράφω όλα τα mutexes και conditions που έχω δηλώσει στο .h file και κάνω return. Εκεί τελειώνει το πρόγραμμά μου!