ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
Λειτουργικά Συστήματα
Εαρινό Εξάμηνο 2019-2020

Ελένη Σαξώνη, Α.Μ: 3160153
Αγγελική Φέκα, Α.Μ: 3140290
Αριστείδης Χρονόπουλος, Α.Μ: 3160194

2η Προγραμματιστική Εργασία

1. Δομή πηγαίου κώδικα:

- Αρχικά καλείται η μέθοδος **main(int argc, char *argv[ ])**, η οποία παίρνει ως όρισμα από τον χρήστη τον αριθμό των πελατών και τον σπόρο (seed) για την **rand_r**. Με **rand_r** παράγουμε τυχαία τον χρόνο που δεχόμαστε κάθε νέα παραγγελία και τον αριθμό από πίτσες που περιέχει η κάθε παραγγελία. Στην περίπτωση που ο χρήστης δεν δώσει τον απαραίτητο αριθμό ορισμάτων, του εμφανίζεται το κατάλληλο μήνυμα σφάλματος.
- Ανάλογα με τον αριθμό των πελατών δημιουργούμε και τα αντίστοιχα νήματα με την χρήση της **pthread_create**.
- Επίσης, σύμφωνα με τον αριθμό των πελατών δημιουργούμε και τρείς(3) πίνακες με τον ανάλογο μέγεθος προκειμένου να αποθηκεύσουμε σε αυτούς τους τα απαραίτητα δεδομένα για την εργασία μας. Συγκεκριμένα δημιουργούμε έναν πίνακα για τις παραγγελίες, έναν πίνακα για να αποθηκεύουμε τον χρόνο παράδοσης των παραγγελιών και έναν πίνακα για να αποθηκεύσουμε τον χρόνο κρυώματος των παραγγελιών.
- Με την **pthread_create** καλείται η **execute(void * customer_id)** για κάθε νήμα που δημιουργούμε, η οποία υλοποιεί τα εξής:
    - Δέχεται κάθε παραγγελία μετά από ένα τυχαίο χρόνο *accept_delay_time* και κρατά την χρονική στιγμή που δέχεται την παραγγελία στην μεταβλητή **order_start** με την χρήση της **clock_gettime**.
    - Κρατάμε τον χρόνο όταν μια παραγγελία παραδίδεται σε κάποιον παρασκευαστή στην μεταβλητή **prep_start**.
    - Υπολογίζει τυχαία με την χρήση της **rand_r** των αριθμό των πιτσών που θα έχει η κάθε παραγγελία και περιμένει **T_prep** για κάθε μια από αυτές.
    - Ύστερα όλες οι πίτσες ψήνονται όλες μαζί εφόσον έχει βρεθεί διαθέσιμος φούρνος και περιμένουμε για χρόνο **T_bake** μέχρι να ολοκληρωθεί το ψήσιμο.
    - Εφόσον ολοκληρωθεί το ψήσιμο ο πρώτος διαθέσιμος delivery βγάζει την παραγγελία από τον φούρνο και ξεκινάει για να την παραδώσει. Ο χρόνος που χρειάζεται ο delivery για την παράδοση της παραγγελίας στο σπίτι υπολογίζεται τυχαία με την χρήση της **rand_r.**
    - Τέλος, υπολογίζεται ο χρόνος που χρειάζεται από την εισαγωγή μιας παραγγελίας μέχρι την παράδοσή της στον πελάτη, όπως και ο χρόνος που

χρειάζεται από την ολοκλήρωση του ψησίματος της παραγγελίας μέχρι και την παράδοση της στον πελάτη. Οι χρόνοι αυτοί αποθηκεύονται στις μεταβλητές **time_X** και **time_Y** αντίστοιχα.

- Όταν τα νήματα ολοκληρώνονται και έλεγχος επιστρέφει στην main διασχίζουμε του πίνακες που δημιουργήσαμε έτσι ώστε να βγάλουμε τα τελικά αποτελέσματα. Συγκεκριμένα, στους πίνακες *max_delivery* και *max_cool,* τους οποίους διαχειριζόμαστε μέσω global μεταβλητής τύπου double*, και όταν ο έλεγχος επιστρέφει στην main υπολογίζουμε την μέγιστη τιμή χρόνου παράδοσης παραγγελιών και την μέγιστη τιμή χρόνου κρυώματος αντίστοιχα.

2. Περιγραφή Υλοποίησης Μεθόδων:

- Για τη δημιουργία των νημάτων κάναμε χρήση της int **pthread_create**(pthread_t * thread , const pthread_attr_t * attr , void *(* start_routine) (void *), void * arg); Το νέο thread που δημιουργείται ξεκινά εκτελώντας την μέθοδο που καλείται ως start_routine().
- Προκειμένου να πετύχουμε τον συγχρονισμό των νημάτων χρησιμοποιήσαμε mutexes. Ουσιαστικά ένα mutex είναι ένα lock, το οποίο θέτουμε πριν την χρήση των κοινών δεδομένων των threads, και το ελευθερώνουμε εφόσον τελειώσουμε με την χρήση αυτή. Συγκεκριμένα χρησιμοποιούμε τα mutex **mutex_available_cooks**, **mutex_available_ovens**, **mutex_available_deliverer**, **mutex_count**, **mutex_console**, **mutex_total_order_completion, mutex_sum** και **mutex_avg** πριν από κάθε υπολογισμό των κοινών δεδομένων, δηλαδή των μετρητών **available_cooks** (μετρητής για τον αριθμό των διαθέσιμων cooks κάθε χρονική στιγμή), **available_ovens** (μετρητής για τον αριθμό των διαθέσιμων φούρνων κάθε χρονική στιγμή), **available_deliverer** (μετρητής για τον αριθμό των διαθέσιμων deliverer), **count** (μετρητής για τον αριθμό των παραγγελιων) και πριν από τον υπολογισμό του συνολικού χρόνου προετοιμασίας της παραγγελίας. Κάθε φορά που υπολογίζουμε έναν από αυτούς τους μετρητές χρησιμοποιούμε **pthread_mutex_lock** προκειμένου να κλειδώσουμε το επιθυμητό mutex έτσι ώστε να μην είναι προσβάσιμο εκείνο το σημείο του κώδικα στα υπάρχοντα threads. Όταν τελειώσουμε με τους υπολογισμούς μας χρησιμοποιούμε **pthread_mutex_unlock** έτσι ώστε να διαθέσουμε τους μετρητές με τις νέες τιμές στα threads μας.
- Με την χρήση της **int pthread_join**(pthread_t thread, void **retval): περιμένει το συγκεκριμένο thread που δέχεται σαν όρισμα να τελειώσει την εκτέλεση του. Αν το retval δεν είναι NULL, τότε η pthread_join αντιγράφει το exit status του thread στην θέση που υποδεικνύει η *retval. Στην περιπτωση μας κάθε thread επιστρέφει τον χρόνο ολοκλήρωσης του.

3. Αποτελέσματα: