

Εργασία 1

Δομή

Αρχείο Κεφαλίδας

Το αρχείο κεφαλίδας περιέχει τις δηλώσεις των σταθερών, την δημιουργία ενός τύπου Boolean (παρόλο που διατίθεται κατάλληλη βιβλιοθήκη κρίθηκε πιο αποδοτικό), την δήλωση δυο συναρτήσεων (μιας custom atoi και την συνάρτησης που κάνει την παραγγελιοληψία και την παράδοση). Επίσης στο αρχείο κεφαλίδας περιέχονται και οι συμπεριλήψεις όλων των βιβλιοθηκών που είναι αναγκαίες για το κυρίως πρόγραμμα.

Αρχείο C

Το πρόγραμμα ξεκινάει με τις απαραίτητες δηλώσεις μεταβλητών, mutexes και cond. Λόγω της δομής του προγράμματος θα ακολουθήσουμε μη γραμμική περιγραφή.

Αρχικά η συνάρτηση custAtoi ελέγχεται έναν πίνακα χαρακτήρων για το αν είναι αριθμός και τον μετατρέπει σε ακέραιο. Ελέγχει αν υπάρχει '-' σαν χαρακτήρας (κάτι που μπορεί να δείχνει αρνητικό αριθμό) και αν κάθε χαρακτήρας είναι αριθμητικός όπως και αν ο αριθμός των χαρακτήρων δεν είναι πάνω από 10 (αφού σίγουρα δεν θα χωράει σε μεταβλητή τύπου int). Αν κάποια από τις δυο τελευταίες συνθήκες είναι ψευδής τότε τυπώνεται μήνυμα λάθους και το πρόγραμμα τερματίζει, αλλιώς επιστρέφεται μια ακέραια τιμή (μέσω atoi).

Η κύρια συνάρτηση ελέγχει αρχικά τον αριθμό των console arguments (2+1) και επιστρέφει μήνυμα λάθους αν ο αριθμός είναι λάθος και τερματίζει το πρόγραμμα. Αλλιώς συνεχίζει και αναθέτει στην μεταβλητή τυχαίου σπόρου και πελατών κατάλληλο αριθμό μέσω της custAtoi ανάλογα με το εισαχθέν console argument. Ελέγχουμε αν οι πελάτες είναι θετικός αριθμός (και αν δεν είναι επιστρέφουμε μήνυμα λάθους και τερματίζουμε). Ύστερα αρχικοποιούμε τα mutex και τα conditions και ξεκινάμε να δημιουργούμε threads (που συμβολίζουν πελάτες) με τυχαία αναμονή μετά την δημιουργία κάθε thread. Τα threads δημιουργούνται σε μια συνάρτηση (που θα αναφερθεί αμέσως μετά). Τελικά εκτυπώνουμε συνολικούς αριθμούς χρόνου όπως και μέσο χρόνο (σε δευτερόλεπτα αλλά και σε λεπτά – αντιμετωπίζοντας όλους τους χρόνους ως λεπτά και απελευθερώνουμε την μνήμη από ένα pointer και mutexes/conds.

Η συνάρτηση που εκτελείται για κάθε thread ξεκινάει μετρώντας τον χρόνο (ώστε να ξέρουμε πόση ώρα κάνει κάθε πελάτης αλλά και την συνολική ώρα – η οποία εισάγεται μέσω pointer). Ύστερα, κάθε φορά που γίνεται χρήση κοινού resource (π.χ. οθόνη, bakers, ovens, time) κάνουμε lock με κατάλληλο mutex. Πιο συγκεκριμένα ελέγχουμε αν υπάρχουν διαθέσιμοι παρασκευαστές (αφού έχουμε κλειδώσει) και αν δεν υπάρχουν περιμένουμε.

Μόλις υπάρξουν δεσμεύουμε έναν κατασκευαστή και ξεκλειδώνουμε. Φυσικά περιμένουμε χρόνο που ισούται με τον χρόνο παρασκευής επί τον αριθμό πίτσας που έχει ο κάθε πελάτης(που είναι τυχαίος). Ακριβώς ανάλογα δρούμε και για τους φούρνους μόνο που περιμένουμε σταθερό χρόνο αφού όλες οι πίτσες ψήνονται μαζί. Τελικά αποδεσμεύουμε έναν παρασκευαστή και έναν φούρνο μόλις ετοιμαστεί μια παραγγελία (αποδεσμεύονται μαζί αφού ένας παρασκευαστής είναι δεσμευμένος καθ'όλη την διάρκεια παρασκευής). Ξαναμετράμε τον χρόνο και βρίσκουμε πόσος χρόνος πέρασε για έναν πελάτη και επίσης τον αθροίζουμε στο συνολικό χρόνο. Τέλος κάνουμε exit το thread.

Να σημειωθεί πώς το .sh δημιουργεί κάποια προβλήματα. Η εργασία δουλεύει επιθυμητά αν γίνει το compilation και το run από το shell χωρίς την χρήση script (υπάρχει και αντίστοιχη επικοινωνία με τον κ. Καρακωνσταντή).