Λειτουργικά Συστήματα Ι, Ασκηση 2

Σπύρος Σειμένης 5070

4 Ιανουαρίου 2013

1 Εισαγωγή

Η πλήρης λειτουργία του κώδικα περιγράφεται αναλυτικά με σχόλια μέσα στα αρχεία. Παρακάτω περιγράφω τις δομές που έχω χρησιμοποιήσει και μια εκτέλεση του προγράμματος, δεν θα αναφερθώ σε λεπτομέρειες υλοποιήσης.

2 Περιγραφή λειτουργίας

2.1 common.h

Στο παραπάνω αρχείο υπάρχουν:

- Δήλωση κοινών βιβλιοθηκών που χρειάζονται τα άλλα δύο αρχεία
- Συναρτήσεις ελέγχου fatal και debug
- Γενικές μεταβλητές του προγράμματος

2.2 client.c

Ενέργειες του client:

- Άνοιγμα κοινού socket για επικοινωνία με τον server
- Δημιουργία παραγγελίας
  - Αν υπάρχουν ορίσματα δημιουργεί μια τυχαία παραγγελία
  - Αν όχι, ο χρήστης δίνει είσοδο
- Παραμένει ανοιχτός για να λαμβάνει μηνύματα του server (πχ cocacoles ή έλλειψη μνήμης) και κλείνει μόλις λάβει το μήνυμα DONE

2.3 server.c

Η κεντρική main του server το μόνο που κάνει είναι συνεχώς να ακούει για παραγγελίες και μόλις λάβει μία παραγγελία να καλεί την _init_proc η οποία θα δημιουργήσει ένα νέο thread για την διαχείρισή της (2.3.1) και συνεχίζει να κάνει accept. Ο έλεγχος για καθυστερημένες παραγγελίες αντί να είναι βάρος του server είναι αρμοδιότητα του thread της κάθε παραγγελίας.

2.3.1 Thread παραγγελίας

Εσωτερικά το κάθε παιδί λαμβάνει την παραγγελία απο τον client και την κάνει parse σε ένα πίνακα κωδικών (0-2). Αοφύ λάβει την παραγγελία δημιουργεί και ένα νέο thread delay το οποίο μετράει σε διαστήματα TVERYLONG και ενημερώνει για κοκακολα τον client, έως ότου το thread παραγγελίας το καταστρέψει(μετά την ολοκλήρωση του delivery).

Έπειτα προσθέτει την παραγγελία στην λίστα παραγγελιών (λεπτομέρειες στο section 2.3.3) και ξεκινάει η διαδικασία ανάθεσης σε ψήστη. Η παραγγελία εκτελεί ένα loop μέχρι να δώσει επιτυχώς όλες τις πίτσες της σε ψήστες. Σε κάθε loop κλειδώνεται το mutex baker_lock έτσι ώστε να μειωθεί το num_bakers ή να αναμένει στο condition variable free_bakers εως ότου ένας ψήστης ελευθερωθεί. Αφού βρεθεί ψήστης δημιουργεί ένα αυτόνομο thread ψήστη με όρισμα τον τύπο της πίτσας που καλείται να ψήσει (2.3.2).

Μόλις δώσει όλες τις πίτσες της σε ψήστες βγαίνει απο το loop ανεξαρτήτως αν έχει τελειώσει το ψήσιμο και περιμένει τους ψήστες(παιδιά της) να τελειώσουν με join. Μόλις όλοι οι ψήστες τελειώσουν ακολουθείται η ίδια διαδικασία για την ανάθεση σε delivery (χωρίς το loop προφανώς). Μόλις κάνει join και τον deliverά ακυρώνει το thread καθυστέρησης και διαγράφει την παραγγελία απο την λίστα.

2.3.2 Baker/delivery process

Το thread του baker με αυτό του delivery είναι πανομοιότυπα και ο μόνος λόγος που είναι ξεχωριστές συναρτήσεις είναι για να είναι πιο ξεκάθαρο.

Το thread του baker/delivery δημιουργεί ένα δικό του mutex και condition variable στο οποίο περιμένει μέχρι να γίνει timeout, διότι η sleep παρατηρήθηκε ότι δεν δίνει ακριβή αποτελέσματα. Έπειτα αυξάνει τον αριθμό διαθέσιμων ψηστών και στέλνει signal στην free_bakers για να ενημερώσει τις παραγγελίες που περιμένουν ελεύθερο ψήστη.

2.3.3 Λίστα παραγγελιών

Η λίστα παραγγελιών υλοποιείται με μια διπλά συνδεδεμένη λίστα για γρηγορότερους χρόνους σβησίματος και εισαγωγής αλλά και για να εκτελούνται παράλληλα οι παραγγελίες που φτάνουν.

Η λίστα παραγγελιών υλοποιείται σε ένα σταθερού μεγέθους πίνακα απο structs order_info. Διαθέτει condition variable και mutex για την ελεγχόμενη εισαγωγή/διαγραφή σε αυτήν.

Μιας και η λίστα είναι σταθερού μεγέθους θα τοποθετεί σειριακά τις παραγγελίες που εισάγονται μέχρι να γεμίσει MAX_ORDERS. Παρόλ' αυτά μόλις διαγραφεί μια παραγγελία απο αυτήν, εισάγει την, πλέον ελεύθερη, θέση της σε μία στοίβα ελεύθερων θέσεων. Έτσι ο server μπορεί να εξυπηρετήσει όσες παραγγελίες και να έρθουν αφού όταν μια παραγγελία

εισάγεται στην λίστα τσεκάρει αν έχει γεμίσει ήδη μια φορά και αν ναι θα μπεί σε θέση που θα κάνει pull απο την στοίβα.

2.3.4 Στοίβα διαχείρισης μνήμης

Η στοίβα διαχείρισης της μνήμης της λίστας είναι σταθερού μεγέθους ίδιου με το μέγεθος της λίστας. Διαθέτει mutex για τον έλεγχο πρόσβασης σε αυτή και ένα condition variable το οποίο σηματοδοτείται όταν υπάρχει κάποιο στοιχείο στην λίστα.

Οι διαδικασίες του pull και του push ελέγχονται απο το mutex. Επιπρόσθετα στην pull υπάρχει έλεγχος για το αν η στοίβα έχει στοιχεία με το condition var opt.full επειδή έτσι αν και το max_orders είναι 100 μπορώ να στείλω 200 παραγγελίες και αυτές να περιμένουν στην pull για ελεύθερη θέση. Στην push δεν υπάρχει τέτοιος έλεγχος διότι η λογική του προγράμματος εκτελεί παράλληλα MAX_ORDERS παραγγελίες άρα πάντα υπάρχει κενή θέση στην στοίβα. Η opt.full γίνεται signal μετά το πέρας μιας push.

2.3.5 Καθυστερημένες παραγγελίες

Ο έλεγχος καθυστερημένων παραγγελιών είναι αρμοδιότητα της κάθε παραγγελίας και όχι του server για λόγους ταχύτητας στην εξυπηρέτηση νέων πελατών.

Με το που ξεκινάει ένα thread παραααγγελίας, αφού λάβει την παραγγελία του ξεκινάει και ένα δεύτερο thread. Αυτό το thread μόλις περάσει tverylong χρόνος θα αναλάβει να ενημερώσει τον client ότι θα πάρει cocacola, και θα συνεχίσει το loop έως ότου να γίνει cancel απο το thread παραγγελίας.