Εργαστήριο Δικτύων Υπολογιστών

Άσκηση 1

Ονοματεπώνυμο: Κυριακού Ανδρόνικος

AM: 5806

mail: kyriakou@ceid.upatras.gr

Στα πλαίσια της άσκησης, υλοποιήθηκε η επικοινωνία ενός client και ενός server βασισμένη σε ένα πρωτόκολλο για την ανταλλαγή μηνυμάτων μεταξύ των δυο και την επίτευξη της αποθήκευσης ζευγαριών κλειδιού και τιμής (key-value).

Παρακάτω, παρατίθενται και αναλύονται ο client και οι διάφορες υλοποιήσεις διαφορετικών αρχιτεκτονικών του server που αναπτύχθηκαν για την άσκηση αυτή.

1. Client

Ο client, αρχικά, περιέχει όλες τις ενέργειες που χρειάζονται προκειμένου να ανοίξει ένα TCP socket και να αρχίσει να συνδέεται με τους servers που θα αναλυθούν παρακάτω. Για να το πετύχει αυτό, αρχικοποιείται μια δομή τύπου sockaddr_in για να αποθηκευτεί η δικτυακή διεύθυνση του server με τον οποίο θέλουμε να συνδεθούμε και έπειτα μετατρέπεται η διεύθυνση – domain name, σε IP διεύθυνση με χρήση της συνάρτησης gethostbyname(). Ύστερα ,η συνδεση ρυθμίζεται με τις κατάλληλες παραμέτρους του server στη δομή sockaddr_in και με χρήση της connect γίνεται προσπάθεια για σύνδεση σε αυτόν.

Όσον αφορά την κυρίως λειτουργία του client, διατρέχονται με χρήση μιας δομής επανάληψης όλα τα command line arguments και αν βρεθεί εντολή put δημιουργείται ένα μήνυμα το οποίο στην πρώτη θέση περιέχει τον χαρακτήρα ‘p’, στις επόμενες το όρισμα (τιμή του key) ακολουθούμενες από ‘\0’ και έπειτα το όρισμα που αφορά το value το οποίο και τερματίζει με ‘\0’. Για την αξιόπιστη αποστολή στον server του μηνύματος, χρησιμοποιείται η συνάρτηση writen από το βιβλίο του Stevens η οποία επαναληπτικά προσπαθεί να στείλει το εκάστοτε μήνυμα μέχρι να βεβαιωθεί ότι δεν έχει άλλα εναπομείναντα bytes. Η λειτουργία του get για την αποστολή είναι παρόμοια ενώ για λήψη τιμής έχει δημιουργηθεί μια επαναληπτική δομή για να εγγυηθούμε ότι θα ληφθεί ολόκληρη και σωστή η απάντηση του server. Η επανάληψη λειτουργεί μέχρι είτε να διαβάσει ‘\0’ και άρα να έχει ανιχνευθεί ότι βρέθηκε η τιμή που ζητήθηκε είτε στην περίπτωση που δεν βρέθηκε, αν ο πρώτος χαρακτήρας είναι ‘n’ να σταματήσει να περιμένει για την υπόλοιπη απάντηση του server. Για την εκτύπωση των τιμών ακολουθείται το πρωτόκολλο που περιγράφεται στην εκφώνηση, δηλαδή αν βρεθεί τιμή, τυπώνεται σε μια γραμμή και ακολουθείται από newline, ενώ διαφορετικά τυπώνεται μόνο newline.

2. Iterative Server (serv1)

Ο πρώτος server είναι iterative,υποστηρίζει δηλαδή ένα client κάθε φορά. Πιο συγκεκριμένα, στον κώδικα που αναπτύχθηκε, αρχικά ορίζονται όλοι οι παράμετροι για την δημιουργία και την λειτουργία του server. Λεπτομέρειες που αξίζει να αναφερθούν είναι η χρήση του INADDR_ANY προκειμένου να μπορεί να δεχθεί συνδέσεις σε όλα τα interfaces του υπολογιστή καθώς και η χρήση του αριθμού 5 ως listen backlog.

Πιο αναλυτικά, για την λειτουργία του συγκεκριμένου server δημιουργείται ένα infinite loop το οποίο δέχεται συνδέσεις οι οποίες αν είναι επιτυχείς αντιγράφουν το περιεχόμενο της read σε μια μεταβλητή buffer. Στη συνέχεια, μια δομή επανάληψης for αναλαμβάνει να προσπελάσει όλα τα στοιχεία του buffer και να διαχωρίσει την εκάστοτε εντολή από τα keys και τα values. Έπειτα γίνεται η εκτέλεση της κάθε εντολής. Για τις εντολές τύπου put απλά προσπελαύνεται η συνάρτηση put η οποία ελέγχει την μνήμη και αν βρει το κλειδί, αντικαθιστά την τιμή, ενώ διαφορετικά αυξάνει την μεταβλητή counter η οποία δείχνει την τελευταία θέση στην οποία έγινε εγγραφή και γράφει σε αυτή. Για τις εντολές τύπου get προσπελαύνεται ο πίνακας ο οποίος περιέχει τα κλειδιά και αν βρεθεί αντιστοιχία επιστρέφεται το περιεχόμενο της αντίστοιχης θέσης του πίνακα με τα values.

Αφού γίνει όλα το παραπάνω, για να επιστραφούν στον client οι τιμές χρησιμοποιείται η written η οποία αναλύθηκε παραπάνω.

Έπειτα δημιουργήθηκε ένα κομμάτι κώδικα για την περίπτωση που ο buffer που δέχεται δεδομένα από τον client έχει γεμίσει. Στην περίπτωση αυτή, αδειάζουμε το αρχικό κομμάτι του buffer μέχρι το σημείο που αποτελούσε την τελευταία εντολή που εκτελέστηκε. Έπειτα, μεταθέτουμε τα περιεχόμενα του buffer έτσι ώστε να πάνε στην αρχή του (να ξεκινάνε ουσιαστικά από τη θέση 0) και στον χώρο που απελευθερώνεται στο τέλος του, τοποθετούμε ότι επιστρέφει μια νέα read –η οποία λειτουργεί επαναληπτικά όπως και στην περίπτωση του client για να εγγυηθούμε ότι θα έρθουν όσα bytes περιμένουμε.

Η διαδικασία αυτή επαναλαμβάνεται μέχρι η read να σταματήσει να επιστρέφει αριθμό bytes και άρα ουσιαστικά να δείξει ότι τελείωσε η εντολή.

Σημείωση:

Οι παρακάτω servers διαμοιράζονται την λειτουργικότητα που περιγράφηκε για τον serv1 και οι μοναδικές διαφορές που αναλύονται βασίζονται στο διαφορετικό αρχιτεκτονικό μοντέλο του καθενός και τις παραμετροποιήσεις που ήταν αναγκαίες για να λειτουργήσει.

3. On demand forking server (serv2)

Ο δεύτερος server αποτελεί on – demand forking server και δημιουργεί μια διεργασία παιδί για κάθε client που συνδέεται.

Στην περίπτωση του συγκεκριμένου server δημιουργήθηκε μια δομή η οποία αποτελείται από τον αριθμό των εγγραφών καθώς και από τους δυο πίνακες που φιλοξενούν keys και values. Καθώς οι διεργασίες δεν έχουν κοινό χώρο διευθύνσεων, η δομή αυτή φιλοξενείται σε ένα κομμάτι shared memory το οποίο αρχικοποιείται στην συνάρτηση main και

προσπελαύνεται χωρίς πρόβλημα από όλες τις διεργασίες αλλά και από τις συναρτήσεις. Προκειμένου να μπορεί κάθε διεργασία να συμπεριφέρεται ξεχωριστά και να διαχειρίζεται κάποιον client, όλη η λειτουργικότητα που ακολουθεί το accept μεταφέρθηκε στην συνάρτηση worker στην οποία στέλνεται ως όρισμα το file descriptor της κάθε σύνδεσης. Για να μην υπάρχουν προβλήματα συγχρονισμού, η συνάρτηση worker προφυλάσσεται από ένα σημαφόρο ο οποίος κατεβαίνει πριν την προσπέλαση της και ανεβαίνει αφού τελειώσει την διαδικασία.

Τέλος, για να περιμένει ο πατέρας τα παιδιά-διεργασίες και να μην δημιουργηθούν zombies δημιουργήθηκε η συνάρτηση sig_child η οποία αποτελεί signal handler και ενεργοποιείται από παιδιά τα οποία τερματίζουν ενώ σημαντικό είναι και το γεγονός ότι κατά την αρχικοποίηση του server ενεργοποιήθηκε η επιλογή SO_REUSEADDR για να υποστηριχθούν οι πολλαπλές διεργασίες.

4. Preforking server (serv3)

Ο server αυτός έχει πανομοιότυπη λειτουργία με τον προηγούμενο (serv2) με την μοναδική διαφορά ότι δημιουργεί αριθμό παιδιών ίσο με ένα νούμερο δοσμένο σαν command line argument τα οποία στη συνέχεια αναλαμβάνουν να διεκπεραιώσουν την λειτουργία του server.

Στην περίπτωση αυτή, όπως και παραπάνω, χρησιμοποιούμε shared memory προκειμένου να υπάρχει συνέπεια στις τιμές που βλέπει κάθε διεργασία αλλά και ένα σημαφόρο για να ρυθμίσουμε την πρόσβαση στους κοινούς πόρους.

5. Multi-threaded server (serv4)

Ο τέταρτος και τελευταίος server έπρεπε να υποστηρίζει τη δημιουργία ενός thread για κάθε client που συνδέεται.

Είναι γνωστό ότι τα threads έχουν κοινό χώρο διευθύνσεων μεταξύ τους και μοιράζονται αυτόν του πατέρα τους άρα δεν κατέστη αναγκαίο να χρησιμοποιηθεί shared memory. Αντίθετα, ακολουθήθηκε η λογική του serv1 όπου οι κοινόχρηστες μεταβλητές δηλώθηκαν σαν global και μπορούσαν να προσπελαστούν τόσο από όλα τα threads όσο και από τις συναρτήσεις put και get στις οποίες μεταβαλλόντουσαν οι τιμές τους. Τέλος, δημιουργήθηκε μια νέα συνάρτηση η doit προκειμένου τα threads να γίνονται detach και όχι join (η οποία περιμένει να τελειώσει το thread που ξεκίνησε) για να εξασφαλίσουμε ότι θα υπάρχει παραλληλία.