ΑΜ : 1115201400024
Δημήτριος Γάγγας

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΥΣΤΗΜΑΤΟΣ

Εργασία 3

Υλοποιήθηκε σε C++11 με λειτουργικό Linux χωρίς τη χρήση STL ή άλλων βιλιοθηκών και τεσταρίστηκε σε περιβάλλον Ubuntu 18.04 . Χρησιμοποιήθηκε localhost κατά τις δοκιμές.

- Υλοποιήθηκαν όλα τα απαιτούμενα ερωτήματα.
- Δημιουργήθηκαν επιτυχώς τοσο το πρόγραμμα για τον client οσο για το πρόγραμμα του server.
- Χρησιμοποιήθηκε το git και το gitkraken για οπτικοποίηση των εκάστοτε αλλαγών αλλά και για την ευκολία επιστροφής σε προηγούμενα στάδια ανάπτυξης του κώδικα.
- Χρησιμοποιήθηκαν τα self-made template δομές για linked list και HashTable απο την Εργασία1.
- Περιέχονται ξεχωριστά makefile που μεταγλωττίζεται με make για τον client και server.
- Έχουν υλοποιηθεί όλες οι απαιτήσεις της εκφώνησης για τα ορίσματα με όποια σειρά και να δοθούν τα flags καθώς και οι απαραίτητοι έλεγχοι. Π.χ

  ./dropbox_client -d 5_input -p 8005 -w 3 -b 1000 -sp 8882 -sip localhost

  ./dropbox_server -p 8882
- Επίσης, αν δε δωθούν κάποια flags τοτε το προγραμμα ζητάει την ενημέρωση τους.
- Για την επικοινωνία server και clients χρησιμοποιήθηκε η **select() syscall** με σκοπό να παρακολουθεί το listening socket και όλα τα sockets που δημιουργούνται μέσω της **accept()** κλήσης για επικοινωνία με άλλους πελάτες ή τον server.
- **Αλλαγή που έγινε στο δοθέν πρωτόκολλο** είναι να στέλνει στο LOG_OFF ο εκάστοτε client και το tuple <ip,port> ετσι ώστε να ξέρει ο server ποιος έφυγε.Γιατι αν για παράδειγμα βρίσκονταν 2 clients στο ιδιο μηχάνημα σε διαφορετικά ports ο server δε θα γνώριζε ποιος ακριβώς client έφυγε ,καθως δεν ξέρει το port.

Αρχεία που υλοποιήθηκαν

Κοινα αρχεία για client και server:

- clientTuple.cpp/.h :
- socketManipulation.cpp/.h

Ξεχωριστά αρχεία για server:

- serverProtocol.cpp/.h

Ξεχωριστά αρχεία για server:

- clientProtocol.cpp/.h
- hashFunction.cpp/.h
- criticalSection.cpp/.h
- circularBuffer.cpp/.h

Βασικές Δομές - Classes που υλοποιήθηκαν

- **Class Protocol** .
  Η class αυτή είναι διαφορετική για server και client και ουσιαστικά περιέχει τις συναρτήσεις στις οποιες ο server καθώς και το main thread του εκάστοτε client δέχονται requests και απαντούν.
  Αντίστοιχα , για τα worker threads των clients το σκοπό αυτό εξυπηρετεί η thread_protocol thr.

- ***linkedList*<*clientsTuple*> *clients_list***
  Δημιουργήθηκε μια linked list για τους Clients που είναι online.
  [όπου ***clientsTuple*** = <ip,port>]
  Σε κάθε δραστηριότητα server και clients ενημερώνουν τη λίστα.

- **pthread_t workerThrsIds_array[num_thr]**
  Ο array δημιουργήθηκε με σκοπό να κρατάμε τα id των threads έτσι ώστε να μπορέσουμε να αναστείλουμε τη λειτουργία τους κατα τον τερματισμό του προγράμματος.

- **struct** circularBuffer
  Δημιουργήθηκε ο κυκλικός buffer με σκοπό να εξυπηρετεί τους worker threads απο τον οποιον κάνουν place και obtain αιτήματα.

- **struct** info
  Ουσιαστικα κάθε κελί του circularBuffer είναι τύπου **struct** info.
  Η δομή περιέχει <ip,port,path,version>.Σε περίπτωση που είναι απλα ενας καινούργιος client τοτε path = **"None"** και version = 0.

- **struct** CS [aka Critical Section]

  Η δομή ουσιαστικά ενθυλακώνει τις παραπάνω πληροφορίες σε μια έτσι ώστε να περαστεί σαν παράμετρος στη συνάρτηση **void*** worker_function(**void*** shared); για να έχουν access και τα threads στην client_list και στον κυκλικό buffer.

Σημαντικές Παρατηρήσεις

- Στις πληροφορίες της CS δομης ,δηλαδη στην client_list και τον κυκλικό buffer χρησιμοποιήθηκαν mutexes και condition Variables.
  Συγκεκριμένα για τον κυκλικό buffer χρησιμοποιήθηκαν τα εξης cond variables:
  1. pthread_cond_t cond_nonempty; = για οταν δεν είναι άδειος ο buffer.
  2. pthread_cond_t cond_nonfull; = για οταν δεν είναι γεμάτος ο buffer.

- Τέλος για το versioning των αρχείων χρησιμοποιήθηκε το hashing των περιεχομένων των αρχείων.
  Στην αρχή στέλνεται version =0 για να ληφθούν τα αρχεία.
  - Αν ειναι directory στελνεται version =1.
  - Αν ειναι file γίνεται load το περιεχόμενο των αρχείων και έπεται ότι Version = hash(περιεχόμενο αρχείου).

Future Approach

Να βάλω σε ολες τις read while για τα bytes που θέλω να διαβάσω.
Πρακτική αμυντικού προγραμματισμού.