## Εισαγωγή

Στόχος αυτής της εργασίας είναι να εξοικειωθείτε με τον προγραμματισμό νημάτων και τη δικτυακή επικοινωνία. Στα πλαίσια αυτής της εργασίας θα υλοποιήσετε μία απλουστευμένη έκδοση του dropbox όπου διαφορετικοί clients θα συγχρονίσουν ένα σύνολο από αρχεία. Ο κάθε client θα συνδέεται σε ένα dropbox server, θα μαθαίνει ποιοι άλλοι clients έχουν συνδεθεί καθώς επίσης και πληροφορίες για το πώς να επικοινωνεί μαζί τους. Κάθε client θα συνδέεται με κάθε έναν από τους άλλους clients με στόχο να συγχρονίσουν τα αρχεία τους, έτσι ώστε τελικά να έχουν όλοι ενημερωμένα αντίγραφα από όλα τα αρχεία.

### Α) Η εφαρμογή dropbox_server (30%)

Σε αυτή την εργασία θα υλοποιήσετε την εφαρμογή dropbox_server η οποία θα χρησιμοποιείται ως εξής:

```
./dropbox_server -p portNum
```

`portNum`: είναι ένας αριθμός θύρας όπου θα ακούει ο dropbox_server

Η λειτουργία του dropbox server είναι η εξής: θα δέχεται συνδέσεις από clients και θα ακούει στο port για αιτήματα από τους clients. Ο server θα δέχεται και θα εξυπηρετεί τα εξής αιτήματα τα οποία θα έρχονται με στο socket που έχει ανοιχτεί:

1. LOG_ON <IP address, portNum> Αυτό το μήνυμα το στέλνει ένας client για να δηλώσει πως συνδέθηκε στο σύστημα ένας καινούργιος client. Ο server θα πρέπει να διατηρεί μια λίστα μαζί με τις εξής πληροφορίες για το συγκεκριμένο client:
   `client IP address, client portNum`
   Το port number είναι το port όπου ακούει ο συγκεκριμένος client ώστε να μπορεί κάποιος άλλος client να συνδεθεί μαζί του προκειμένου να συγχρονιστούν τα αρχεία. Δεν χρειάζεται να υλοποιήσετε κάποια δομή δεικτοδότησης για τη λίστα, όποτε χρειάζεστε κάτι μπορείτε να την ψάχνετε σειριακά. Υποθέτουμε πως ο client στέλνει τη δική του IP. Επίσης, αν ο συγκεκριμένος client με τα συγκεκριμένα IP/port υπάρχει ήδη στη λίστα δεν τον ξαναβάζετε δεύτερη φορά. Στη λίστα δηλαδή όλες οι εγγραφές είναι μοναδικές.
   Όταν συνδεθεί κάποιος καινούργιος client, ο server θα πρέπει να στέλνει αυτή την πληροφορία που μόλις αποθήκευσε, δηλαδή το `client IP address, client portNum` tuple, σε όλους τους άλλους συνδεδεμένους clients. Η αποστολή θα γίνεται στέλνοντας το string USER_ON <IP address, portNum> σε όλους τους αντίστοιχους clients.
2. GET_CLIENTS στέλνει στον client που έστειλε το request μια λίστα από tuples <IP address, portNum> για τους άλλους πελάτες που βρίσκονται μέσα στο σύστημα. Το πρωτόκολλο για να επιστραφούν οι clients θα είναι CLIENT_LIST N <IP1, port1> … <IPn, portn> όπου N είναι ο αριθμός των tuples που θα επιστραφούν.
3. LOG_OFF Βρίσκει τον client που έστειλε την εντολή στη λίστα, τον αφαιρεί από τη λίστα, και στέλνει ειδοποίηση στους υπόλοιπους clients με μήνυμα USER_OFF <IP address, portNum>. Αν ο client δε βρεθεί επιστρέφεται το μήνυμα: ERROR_IP_PORT_NOT_FOUND_IN_LIST.

Οι τιμές των IP και port θα πρέπει να στέλνονται σε binary μορφή πάνω από το socket (θα χρειαστείτε δηλαδή `htonl/s()` και `ntohl/s()`) ενώ όλα τα υπόλοιπα στοιχεία θα πρέπει να είναι με τη μορφή strings. Μπορείτε να βάλετε όσα κενά χρειάζεστε (αν χρειάζεστε) και επίσης τα < και , μπορείτε να τα θεωρήσετε προαιρετικά αν θέλετε. Τα ίδια ισχύουν και για τις πληροφορίες για τα sockets του dropbox_client πιο κάτω.

**B) Η εφαρμογή dropbox_client (70%)**

θα υλοποιήσετε και μία εφαρμογή την dropbox_client η οποία θα χρησιμοποιείται ως εξής:

```
./dropbox_client -d dirName -p portNum -w workerThreads -b bufferSize -sp
        serverPort -sip serverIP
```

- `dirName`: είναι το directory με τα αρχεία που ο client θα μοιραστεί με τους άλλους πελάτες.
- `portNum`: είναι η θύρα όπου θα ακούει ο dropbox_client για να δεχθεί μηνύματα από το server και από άλλους πελάτες.
- `workerThreads`: είναι ο αριθμός των WorkerThread νημάτων που θα δημιουργήσει ο dropbox_client. Τα threads μπορούν να δημιουργηθούν μια φορά στην αρχή όταν θα ξεκινάει ο dropbox_client.
- `bufferSize`: είναι το μέγεθος ενός **κυκλικού** buffer που θα μοιράζεται ανάμεσα στα νήματα που δημιουργούνται από το dropbox_client process.
- `serverPort`: είναι ο αριθμός θύρας όπου ακούει ο dropbox_server στον οποίο θα συνδεθεί ο dropbox_client.
- `serverIP`: είναι η IP διεύθυνση του dropbox_server στον οποίο θα συνδεθεί ο dropbox_client.

Η λειτουργία του multithreaded dropbox_client είναι η εξής. Το αρχικό main thread θα συνδέεται στον dropbox_server για να δηλώσει την παρουσία του πελάτη στο σύστημα με LOG_ON μήνυμα. Κατόπιν, θα στέλνει GET_CLIENTS μήνυμα ώστε να λάβει πληροφορίες, συγκεκριμένα ένα tuple <IP address, portNum> για όλους τους άλλους clients που βρίσκονται ήδη μέσα στο σύστημα. Για κάθε client για τον οποίο λαμβάνει ένα tuple, το αρχικό νήμα θα τοποθετεί το tuple σε μία client_list λίστα η οποία θα είναι κοινή για όλα τα threads (χρειάζεται synchronized access στη λίστα).

Στη συνέχεια, θα δημιουργεί τον κοινόχρηστο **κυκλικό** buffer, ο οποίος θα αποτελείται απο `bufferSize` στοιχεία, όπου το κάθε στοιχείο αποτελείται από τέσσερα πεδία: {pathname (128 bytes), version, IP address, portNum}. Τέλος, θα ξεκινάει `worker_threads` αριθμό από Worker Threads και θα περιμένει συνδέσεις από άλλους πελάτες (ή τον server) για να εξυπηρετεί τριών ειδών αιτήματα. Για κάθε αίτημα που θα έρχεται από κάποιον που συνδέθηκε στο αρχικό main thread, θα πρέπει να ελέγχεται αν η IP και το port αυτού που συνδέθηκε βρίσκεται στη client_list λίστα.

Το αρχικό main νήμα θα χειρίζεται τα εξής εισερχόμενα αιτήματα:

1. GET_FILE_LIST στέλνει στον πελάτη μια λίστα ονομάτων (pathnames) όλων των αρχείων που βρίσκονται στο φάκελο `dirName` . Τα αρχεία μπορούν να είναι σε υποκαταλόγους κάτω από το dirName. Το πρωτόκολλο για να επιστραφούν τα ονόματα θα είναι FILE_LIST n <pathname1, version1> … <pathnameN,versionN> όπου n είναι ο αριθμός των ονομάτων αρχείων που θα επιστραφούν. Θεωρούμε πως το pathname δεν μπορεί να περιέχει το κόμμα.

2. GET_FILE <pathname, version> Ελέγχει αν το αρχείο με όνομα `dirName`/pathname υπάρχει (το `dirName` έχει δοθεί ως παράμετρος στο πρόγραμμα). Αν δεν υπάρχει τότε στέλνεται το string FILE_NOT_FOUND. Αν υπάρχει, ελέγχει αν έχει αλλάξει το αρχείο σε σχέση με την έκδοση που ζητείται. Αν δεν ἐχει αλλάξει το αρχείο (δηλαδή η version αντιστοιχεί στην τελευταία εκδοχή του αρχείου), τότε στέλνεται το string FILE_UP_TO_DATE στον πελάτη. Αν η τοπική έκδοση διαφέρει από τη ζητούμενη, τότε θα πρέπει να επιστραφεί το αρχείο. Το τι θα περιέχει η πληροφορία version είναι δική σας σχεδιαστική επιλογή: μπορεί να είναι π.χ., ένα hash, ένα timestamp, η ένας αύξοντας αριθμός. Αν έχει αλλάξει το αρχείο, τότε επιστρέφεται το string FILE_SIZE version n byte0byte1...byten, όπου version είναι η παρούσα έκδοση του

αρχείου, n είναι το μέγεθος του αρχείου σε bytes, και ακολούθως τα bytes του αρχείου. **Σημείωση:** Για λόγους απλότητας, θεωρούμε πως μόνο ο πελάτης που αρχικά διαθέτει το αρχείο (δηλαδή, μοιράζεται το αρχείο με τους άλλους πελάτες) μπορεί να αλλάζει το αρχείο. Η μεταφορά θα είναι πάντα από τον πελάτη που αρχικά μοίρασε το αρχείο προς τους άλλους πελάτες που συνδέονται μαζί του για να ζητήσουν την τελευταία εκδοχή του.

3. LOG_OFF <IP, port> Αυτό το μήνυμα από τον dropbox server υποδεικνύει πως έχει φύγει ενας client από το σύστημα. Σε αυτήν την περίπτωση, η client_list λίστα θα πρέπει να γίνεται updated όταν υπάρξει κάποιο LOG_OFF μήνυμα από τον server. Δεν χρειάζεται να διαγράψετε τα αρχεία του client που έφυγε.

Το αρχικό main νήμα θα πρέπει με select() call να παρακολουθεί το listening socket του dropbox_client και όλα τα sockets που δημιουργούνται μέσω της accept() κλήσης για επικοινωνία με άλλους πελάτες.

Κάθε WorkerThread θα λειτουργεί ως εξής. Αν ο κοινός buffer δεν είναι αδειανός, το WorkerThread θα αφαιρεί ένα αντικείμενο. Υπάρχουν δύο είδη αντικειμένων που θα πρέπει να χειριστεί:

Αν το αντικείμενο είναι tuple <IP addr, portNum>, (που σημαίνει τα πεδία filename και version θα είναι empty ή null ανάλογα με το σχεδιασμό σας), θα συνδέεται στον απομακρυσμένο πελάτη (αν είναι μέσα στο σύστημα), και θα του στέλνει ένα GET_FILE_LIST αίτημα. Στη συνέχεια, για κάθε (pathname, version) που του στέλνει ο απομακρυσμένος πελάτης θα το τοποθετεί στον κοινό buffer, μαζί με τις αντίστοιχες πληροφορίες του απομακρυσμένου πελάτη (IP addr, portNum). Όταν λάβει και τοποθετήσει το τελευταίο pathname στον κοινό buffer, αποσυνδέεται από τον απομακρυσμένο χρήστη και ξεκινάει πάλι την λειτουργία του, αφαιρώντας αντικείμενο από τον κοινό buffer, εφόσον δεν είναι αδειανός. Αν ο buffer γεμίσει κατά την εισαγωγή των αντικειμένων, το thread θα πρέπει να περιμένει να αδειάσουν κάποιες θέσεις στον buffer για να συνεχίσει την εισαγωγή των αντικειμένων.

Αν το αντικείμενο είναι <pathname, version, IP addr, portNum>, ελέγχει αν ο πελάτης με IP addr είναι μέσα στο σύστημα ελέγχοντας την κοινή λίστα με τα IP, portNum tuples. Στη συνέχεια, ελέγχει αν υπάρχει τοπικά αποθηκευμένο το clientName/pathname. Αν δεν υπάρχει τοπικά, συνδέεται στο IP addr/portNum που αναφέρεται και στέλνει ένα GET_FILE αίτημα για το συγκεκριμένο pathname. Αν δεν υπάρχει το αρχείο θα πρέπει πιθανώς να χρησιμοποιήσετε ένα συγκεκριμένο version (π.χ. -1) προκειμένου να γίνει σωστά η σύγκριση και να σταλεί το αρχείο. Αν υπάρχει το clientName/pathname τοπικά αποθηκευμένο, στέλνεται το GET_FILE request με το τοπικό version και, εφόσον η τοπική έκδοση είναι πιο παλιά, θα πρέπει να αρχίσει και η αντιγραφή του αρχείου αλλιώς θα ληφθεί το FILE_UP_TO_DATE. Φυσικά, και στις δύο περιπτώσεις που λαμβάνετε ένα αρχείο θα πρέπει να ενημερωθούν όποιες πληροφορίες κρατούνται τοπικά για την εκδοχή του αρχείου. Το clientName μπορεί να είναι της μορφής clientIP_clientPort ή κάποιο άλλο όνομα (χωριστό όμως για κάθε client) που θα ορίσετε εσείς.

Το client πρόγραμμα θα πρέπει να χειρίζεται την SIGINT signal. Όταν η διεργασία λάβει ένα SIGINT σήμα, το πρόγραμμα θα πρέπει πρώτα να στέλνει ένα LOG_OFF μήνυμα στον server και στη συνέχεια να τερματίζει.

## Παρατηρήσεις

- Στα προγραμματά σας, κοινές μεταβλητές που μοιράζονται ανάμεσα σε πολλαπλά νήματα θα πρέπει να προστατεύονται με τη χρήση mutexes. Τονίζεται πως το busy-waiting δεν είναι αποδεκτή λύση για τη παραμονή πρόσβασης στον κοινό buffer ανάμεσα στα νήματα του dropbox_client προγραμματός σας. Η άσκηση δεν περιγράφει όλες τις λεπτομέρειες και τις δομές για τον απλό λόγο πως οι σχεδιαστικές επιλογές είναι αποκλειστικά δικές σας (βεβαιωθείτε φυσικά πως τις περιγράφετε αναλυτικά στο README). Αν έχετε διάφορες επιλογές για κάποιο σημείο της άσκησης σκεφτείτε τα υπέρ και τα κατά,

τεκμηριώστε τα στο README, επιλέξτε αυτό που θεωρείτε σωστό και λογικό και περιγράψτε γιατί το επιλέξατε στο README.

### Παραδοτέα

- Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματός σας. 1-2 σελίδες ASCII κειμένου είναι αρκετές. Συμπεριλάβετε την εξήγηση και τις οδηγίες για το compilation και την εκτέλεση του προγράμματός σας σε ένα αρχείο README μαζί με τον κώδικα που θα υποβάλετε.
- Οποιαδήποτε πηγή πληροφορίας, συμπεριλαμβανομένου και κώδικα που μπορεί να βρήκατε στο Διαδίκτυο ή αλλού θα πρέπει να αναφερθεί στον πηγαίο κώδικά σας αλλά και στο παραπάνω README.
- Όλη η δουλειά σας (πηγαίος κώδικας, Makefile και README) σε ένα tar.gz file με ονομασία OnomaEponymoProject3.tar.gz. Προσοχή να υποβάλετε μόνο κώδικα, Makefile, README και όχι τα binaries.
- Η σωστή υποβολή ενός σωστού tar.gz που περιέχει τον κώδικα της άσκησής σας και ό,τι αρχεία χρειάζονται είναι αποκλειστικά ευθύνη σας. Άδεια tar.gz ή tar.gz που έχουν λάθος και δεν γίνονται extract δεν βαθμολογούνται.

### Διαδικαστικά

- Για επιπρόσθετες ανακοινώσεις, παρακολουθείτε το forum του μαθήματος στο piazza.com. Η πλήρης διεύθυνση είναι https://piazza.com/uoa.gr/spring2019/k24/home. Η παρακολούθηση του φόρουμ στο Piazza είναι υποχρεωτική.
- Το πρόγραμμά σας θα πρέπει να γραφεί σε C (ή C++). Στην περίπτωσή που χρησιμοποιήσετε C++ δεν μπορείτε να χρησιμοποιήσετε τις έτοιμες δομές της Standard Template Library (STL). Σε κάθε περίπτωση το πρόγραμμά σας θα πρέπει να τρέχει στα Linux workstations του Τμήματος.
- Ο κώδικάς σας θα πρέπει να αποτελείται από τουλάχιστον δύο (και κατά προτίμηση περισσότερα) διαφορετικά αρχεία. Η χρήση του separate compilation είναι επιτακτική και ο κώδικάς σας θα πρέπει να έχει ένα Makefile.
- Βεβαιωθείτε πως ακολουθείτε καλές πρακτικές software engineering κατά την υλοποίηση της άσκησης. Η οργάνωση, η αναγνωσιμότητα και η ύπαρξη σχολίων στον κώδικα αποτελούν κομμάτι της βαθμολογίας σας.

### Άλλες σημαντικές παρατηρήσεις

- Οι εργασίες είναι ατομικές.
- Όποιος υποβάλει / παρουσιάσει κώδικα που δεν έχει γραφτεί από την ίδια/τον ίδιο μηδενίζεται στο μάθημα.
- Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πώς θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιασδήποτε μορφής) είναι κάτι που δεν επιτρέπεται. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικα απλά παίρνει μηδέν στο μάθημα. Αυτό ισχύει για όσους εμπλέκονται ανεξάρτητα από το ποιος έδωσε/πήρε κλπ.
- Οι ασκήσεις προγραμματισμού μπορούν να δοθούν με καθυστέρηση το πολύ 3 ημερών και με ποινή 5% για κάθε μέρα αργοπορίας. Πέραν των 3 αυτών ημερών, δεν μπορούν να κατατεθούν ασκήσεις.