ΟΝΟΜΑ: Πέτρος
ΕΠΩΝΥΜΟ: Βασιλόπουλος
Α.Μ.: 1115201000218

Η εργασία υλοποιεί όλα τα ζητούμενα της άσκησης. Στην ουσία ο MirrorInitiator κάνει σύνδεση στον MirrorServer με sockets και του περνάει το string με τα στοιχεία για τους content servers στους οποίους θα πρέπει να κάνει σύνδεση ο MirrorServer. Μαζί με αυτά περνάει και τον αριθμό τον content server για να γνωρίζει ο MirrorServer πόσα MirrorManager να φτιάξει. Ο MirrorServer στην εκκινησή του φτιάχνει έναν αριθμό απο workers τα οποία ορίζονται απο το threanum που δίνεται σαν όρισμα και θα λειτουργούν και στην επόμενη σύνεση του MirrorServer. Στην συνέχεια καλεί τον server_assistant.

Ο server_assistant παίρνει απο τον initiator το πόσα νήματα θα φτιάξει και στην συνέχεια αρχίζει και διαβάζει ένα ένα τα strings που περιέχουν τις πληροφορίες για τους content server. Κάθε φορά που διαβάζει και ένα string αυτό το περνάει σαν όρισμα στον MirrorManager που θα φτιάξει. Οτάν τελειώσει με όλα αυτά, περιμένει όλα τα νήματα να κάνουν join μεταξύ τους.

Ο MirrorManager παίρνει το string που του έχει περαστεί σαν όρισμα και το σπάει με την βοήθεια της συνάρτησης analyze με το delimeter “:”. Στη συνέχει αποθηκεύει τις πληροφορίες που πήρε με malloc στο thread αλλά και το port και το address σε μια struct για να τις ξαναχρησμοποιήσει. Κάνει σύνδεση στο content server που μόλις πήρε απο το string και στέλνει List,id,delay καθώς το “,” βοηθάει περισσότερο απο το κενό. Μετά μπαίνει σε μια επανάληψη που για κάθε όνομα φακέλου που θα παίρνει απο τον content server κοιτάζει αν έχει σχέση με τον φάκελο που είχε ζητήσει ο initiator με την βοήθεια της dir_exists. Αν το string που μόλις πήρε έχει σχέσει με τον φάκελο που θέλουμε να αντιγράψουμε τον βάζει σε μια ουρά για να το τραβήξει μετά ο worker. Αν η ουρά είναι γεμάτη τότε κοιμάται με την pthread_cond_wait αλλιώς το βάζει στην ουρά και στέλνει σήμα στον worker. Αν δεν είχε σχέσει το string με τον φάκελο τότε αυξάνει μια μεταλητή count2 ή οποιά αν είναι ίση με την count1 που μετράει πόσα συνολικά string θα πάρει, τότε αυτό σημαίνει οτι ο content server δεν προσφέρει τον φάκελο που θέλουμε για αντιγραφή. Βγαίνει απ την επναλάληψη όταν ο content του στείλει “end”.

Ο content server στην συνέχεια φτιάχνει shared memory για την struct η οποία θα κρατά το delay και το id. Shared memry επίσης για μια μεταβλητή ή οποία μετράει πόσα list έχουμε για να βάλει το κάθε delay,id στην σωστή θέση του struct αλλά και τις μεταβλητες bytestTransfered, filestransfered και τα semaphores που θα τα προστατεύουνε. Μετά φτιάχνει ένα fixed size αριθμό απο παιδιά και για κάθε σύνδεση που δέχεται κοιτάει αν είναι για να σταματησει ή όχι. Αν δεν είναι κάνει fork και το παιδί κοιτάει αν θα του δωθεί αίτημα LIST ή FETCH, ο πατέρας προχωράει στην επόμενη σύνδεση. Αν είναι LIST καλέι την make_list μια αναδρομική συνάρτηση η οποία στέλνει στον manager την λίστα με τους φακέλους που εξυπηρετέι ο server. Αν ειναι fetch περιμένει delay seconds και στέλνει τα αρχεία

απο τον φάκελο που ζητήθηκε. Αν έχει πάρει αίτημα τερματισμού τότε στέλνει στον mirror server τα bytes και file transferred και τερματίζει.

Ο worker περιμένει αν η ουρά είναι άδεια, αν δεν είναι τότε σπάει το string που μόλις πήρε απο την ουρά και κάνει σύνδεση στον content server με FETCH στη συνέχεια κοιτάει αν του δόοθηκε φάκελος ή αρχείο. Αν του δόθηκε φάκελος τον δημιουργεί και πηγαίνει στο επόμενο string απο την ουρά, αν του δόθηκε αρχείο το δημιουργέι μέσα στον φάκελο που πρέπει και το ανοίγει για να γράψει σε αυτό ότι έχει και ο κλώνος του στον content server.

Αν ο MirrorServer δεχθεί απο τον initiator μήνυμα τερματισμού τότε κάνει true την μεταβλητή end και στέλνει σήμα σε όλα τα worker με την pthread_signal_broadcast. Το worker οταν ξυπνήσει κοιτάει αν η end είναι true και η ουρά είναι άδεια, αυτό σημαίνει ότι μπορεί αν τερματίσει αφού στην ουρά δεν υπάρχει κάτι άλλο για να τραβηχτεί. Στην συνέχεια ο MirrorServer περιμένει να τελειώσουν τα worker και μετά καλέι την terminator που για κάθε content server συνδέεται σε αυτόν πέρνει τα bytes και file transferred που τα στέλνει στον initiator και στέλνει end στον content.

Για την εργασία έχω διαβάσει όλα τα παραδείγματα του κυρίου δελή και έχω χρησημοποιήσει πολλά απο αυτά όπως τα shared memory αλλά και την σύνδεση με τα socket. Έχω κάνει ερωτήσεςι στο stack overflow και έτσι έδειξα λίγο κώδικα όπως https://stackoverflow.com/questions/44192063/cant-pass-different-strings-to-multiple-threads, https://stackoverflow.com/questions/44319134/transfer-files-through-sockets-in-c, https://stackoverflow.com/questions/44250581/pthread-mutex-lock-works-only-with-sleep, αυτές είναι ερωτήσεις τις οποίες τις έχω κάνει εγώ. Επίσης εμπνεύστικα για την κατασκευή της ουράς από εδω https://stackoverflow.com/questions/35486632/c-queue-char-array .