Ευθυμίου Ιωάννης
1115200600296
K24 – Προγραμματισμός Συστήματος
Project 3

Γενικά
Τα δυο προγράμματα έχουν υλοποιηθεί σύμφωνα με τις προδιαγραφές που δόθηκαν.

Σχεδιαστικές επιλογές

1) Αποστολή και λήψη δεδομένων
Τα δεδομένα χωρίζονται σε 2 κατηγορίες, δεδομένα σταθερού και μεταβλητού μήκους. Και στις 2 περιπτώσεις η αποστολή και λήψη γίνεται μέσω ενός δείκτη char* και μετρητές για το πόσα bytes έχουν σταλεί / διαβαστεί. Στην περίπτωση δεδομένων μεταβλητού μήκους αποστέλλεται πρώτα το πλήθος των bytes που θα αποσταλούν. Η σειρά με την οποία αποστέλλονται τα δεδομένα είναι συγκεκριμένη, επομένως η εφαρμογές ξέρουν τι πρέπει να στείλουν και τι περιμένουν να διαβάσουν.

2) remoteClient
Ο client κατά την εκκίνηση του ελέγχει τα ορίσματα που του δόθηκαν. Εφόσον αυτά είναι σωστά, συνδέεται με τον server (η σύνδεση γίνεται σύμφωνα με το παράδειγμα των διαφανειών), στέλνει το αίτημα στον server και περιμένει την απάντηση.

I) Μορφή απάντησης
Ο client λαμβάνει πρώτα των αριθμό των καταλόγων και των αρχείων που θα λάβει. Στην συνέχεια, εφόσον ο server του απαντήσει ότι θα λάβει αρχεία, λαμβάνει τον πρώτο κατάλογο. Σε αυτόν τον κατάλογο, δημιουργεί όλους τους καταλόγους που υπάρχουν στο pathname (με mode 0777), και τέλος τον κατάλογο που ζήτησε (με mode ίδιο με του server). Στην συνέχεια μπαίνει σε ένα for loop και λαμβάνει και δημιουργεί όλους τους υπολοίπους καταλόγους με mode ίδιο με αυτό του server. Στην συνέχεια μπαίνει σε ένα for loop για όλα τα αρχεία που περιμένει να λάβει.

II) Λήψη αρχείων
Ο client στην αρχή δεσμεύει χώρο ίσο με το page size του. Στην συνέχεια διαβάζει το όνομα, το mode και το μέγεθος του αρχείου. Μετά ελέγχει αν το αρχείο προϋπήρχε και το διαγράφει. Τέλος, μπαίνει σε ένα loop μέχρι να διαβάσει filesize bytes. Σε αυτό το loop βλέπει αν το pagesize ή τα bytes που μένουν είναι λιγότερα, και προσπαθεί να διαβάσει τόσα από το socket. Στην συνέχεια γράφει όλα τα bytes που διάβασε στο αρχείο, και ενημερώνει τους μετρητές. Τέλος, στέλνει το σύνολο των byte που διάβασε στον server, για επιβεβαίωση. Σε περίπτωση που οι αριθμοί δεν ταιριάζουν, ενημερώνεται ο χρήστης.

3) dataServer
Ο server στην αρχή ελέγχει τα ορίσματα που δέχθηκε. Εφόσον αυτά είναι σωστά, αρχικοποιεί το socket και όλες τις απαραίτητες ρυθμίσεις (σύμφωνα με τις διαφάνειες). Στην συνέχεια δεσμεύει έναν πίνακα από pthread_t για να κρατήσει τα thread_id των workerThreads. Μετά δημιουργεί δυναμικά την ουρά των εργασιών και τα workerThreads, με όρισμα τον δείκτη στην ουρά. Τέλος μπαίνει σε ένα while loop, στο οποίο περιμένει clients. Όταν έρθει ένας client, ο server δημιουργεί δυναμικά τα ορίσματα του firstLineThread και ένα pthread_t, δημιουργεί το firstLineThread και αποδεσμεύει τον χώρο του pthread_t.

I) workerThread
Το workerThread μπαίνει σε ένα loop, στο οποίο παίρνει ένα job από την ουρά, εκτυπώνει το id και το job, το εκτελεί, εκτυπώνει ότι τελείωσε και το διαγράφει. Αν λάβει NULL από την ουρά, τερματίζει. Όλος ο συγχρονισμός βρίσκεται στην ουρά και τα jobs.

II) firstLineThread
Το firstLineThread αρχικά τοποθετεί τα στοιχεία που παίρνει από το το όρισμα σε τοπικές μεταβλητές. Στην συνέχεια προσπαθεί να ανεβάσει τον αριθμό των workerThreads στην ουρά. Αν αυτό αποτύχει σημαίνει ότι ο server έχει λάβει εντολή εξόδου, επομένως επιστρέφει 0 αρχεία και καταλόγους στον πελάτη και τερματίζει αφού κάνει τις κατάλληλες ενέργειες. Αν πετύχει και αυτό που έλαβε είναι εντολή εξόδου, απενεργοποιεί την ουρά, μειώνει τον αριθμό των ενεργών workerThreads, καλεί τον terminator και τερματίζει αφού κάνει τις κατάλληλες ενέργειες.

Σε διαφορετική περίπτωση, ελευθερώνει την μνήμη που είχε δεσμεύσει το main thread για τα arguments, και τοποθετεί αναδρομικά όλα τα αρχεία και τους υποφακέλους από το αίτημα του πελάτη σε 2 λίστες. Ενημερώνει τον πελάτη για τα αρχεία και τους φακέλους που θα λάβει και στέλνει τα ονόματα των φακέλων στον πελάτη. Μετά δημιουργεί το mutex για τον πελάτη και έναν ακέραιο στον οποίο θα έχουν πρόσβαση όλα τα workers, το οποίο δείχνει τον συνολικό αριθμό των αρχείων που έχουν αποσταλεί. Στην συνέχεια τοποθετεί τα αρχεία στην ουρά, μειώνει τον αριθμό των active thread και τερματίζει. Πάλι ο συγχρονισμός είναι υπόθεση της ουράς.

III) terminator
Ο terminator περιμένει να τελειώσουν όλα τα active workerThreads από την ουρά, και κλείνει το sock του mainThread. Μετά κάνει join όλα τα workerThreads, στέλνει σήμα στο mainThread και το κάνει join. Στην συνέχεια αποδεσμεύει τον χώρο του argument που πήρε, τον χώρο του πίνακα με τα pthread_id των worker και τον χώρο της ουράς, και τερματίζει.

IV) MyJob
Το MyJob είναι η δουλεία που μπαίνει στην ουρά. Περιέχει έναν δείκτη σε MyINode, που είναι το αρχείο που πρέπει να στείλει, το socket του πελάτη, το mutex για πρόσβαση στο socket και τον αριθμό των ολοκληρωμένων δουλειών, έναν δείκτη στον αριθμό με τις ολοκληρωμένες δουλείες και το σύνολο των δουλειών. Όταν καλεστεί η συνάρτηση sendFileToClient(), κλειδώνεται το mutex, τυπώνονται τα στοιχεία, και αποστέλλεται το αρχείο. Όταν ολοκληρωθεί η δουλειά, ανεβάζει τον μετρητή των ολοκληρωμένων δουλειών και εκτυπώνει ότι τελείωσε. Πάντα μετά από την κλήση αυτής της συνάρτησης καλείται ο destructor. Στον destructor καταστρέφεται το myinode, και γίνεται έλεγχος αν είναι το τελευταίο job. Στην συνέχεια ξεκλειδώνεται το mutex. Αν ήταν η τελευταία δουλειά για τον client, διαγράφονται ο μετρητής και το mutex που έχουν δεσμευτεί δυναμικά και κλείνει το socket.

V) Αποστολή Αρχείων
Η αποστολή των αρχείων γίνεται με τον εξής τρόπο. Στην αρχή δεσμεύεται ένας buffer με μέγεθος όσο το pagesize του server. Στην συνέχεια αποστέλλεται το όνομα, το mode και το μέγεθος του αρχείο. Μετά ανοίγεται το αρχείο, και διαβάζονται έως και pagesize bytes από αυτό και γράφονται στο socket όλα όσα διαβάστηκαν όπως περιγράφηκε πιο πριν.

VI) MyINode
Απλή κλάση που περιέχει το path του αρχείου, το μέγεθος του, το mode του καθώς και άλλα στοιχεία.

VII) MyINodeList
Απλή λίστα με MyINodes

VIII) MyThreadArgs
Κλάση που περιέχει δείκτη στην ουρά, το socket του client, το socket του server, των αριθμό των worker και τα pthread_id τους, και το pthread_id του mainThread. Χρησιμοποιείται από τα firstLineThread και τον terminator.

IX) MyQueue
Η ουρά περιέχει έναν αριθμό με το μέγιστο μέγεθος της, έναν μετρητή των αντικειμένων που περιέχει, έναν μετρητή για τα firstLineThreads που είναι ενεργά, καθώς και μια μεταβλητή bool που δείχνει αν είναι ενεργή ή όχι. Έχει επίσης 3 mutex, ένα για την πρόσβαση στα δεδομένα της ουράς, ένα για τον μετρητή των firstLineThread και ένα για το activeStatus και 3 condition variables, ένα για άδεια ουρά, ένα για γεμάτη και ένα για όταν είναι έτοιμη για κλείσιμο.

a) Εισαγωγή στην ουρά
Κατά την εισαγωγή, κλειδώνεται το κατάλληλο mutex και όσο η ουρά είναι γεμάτη και ενεργή, το thread περιμένει στο κατάλληλο condition variable. Σε περίπτωση που η ουρά δεν είναι ενεργή, ο περιορισμός μήκους της παρακάμπτεται, για να επισπευσθεί η έξοδος. Όταν περάσει την κλειδαριά, τοποθετεί το στοιχείο, και αν η ουρά ήταν άδεια στέλνει σήμα. Μετά ανεβάζει τον μετρητή και ξεκλειδώνει το mutex.

b) Εξαγωγή από την ουρά
Κατά την εξαγωγή, κλειδώνεται το mutex και όσο η ουρά είναι άδεια, το thread περιμένει μέσα σε ένα condition variable. Μέσα στο σώμα του while γίνεται και ένας έλεγχος για το αν η ουρά παραμένει ενεργή. Όταν βγει από την “κλειδαριά”, εφόσον η ουρά είναι άδεια και όχι ενεργή, επιστρέφει NULL. Αλλιώς, αφαιρεί την κορυφή και μειώνει τον μετρητή Σε περίπτωση που η ουρά ήταν γεμάτη, ενημερώνει τα FirstLineThreads. Στην συνέχεια ξεκλειδώνει το mutex και επιστρέφει την κορυφή.

c) Άλλες λειτουργίες
Ο terminator χρησιμοποιεί την ουρά για να περιμένει σε ένα condition variable μέχρι να τελειώσουν όλα τα active firstLineThreads και να ξυπνήσει όλα τα worker για να τερματίσουν.

X) Έξοδος Server
Η έξοδος γίνεται με την αποστολή της λέξης exit από κάποιον client. Μόλις ληφθεί αυτή η λέξη, ο server σταματάει να δέχεται νέα αιτήματά, τα firstLineThreads βάζουν τα jobs στην ουρά ανεξάρτητα από το μήκος της για την επίσπευση της εξόδου και μόλις είναι όλα έτοιμα, αποδεσμεύεται όλη η μνήμη και τερματίζει το πρόγραμμα.

4) Διαχείρηση μνήμης
Επειδή το πρόγραμμα είναι λίγο περίπλοκο, περιγράφω που δεσμεύεται και που αποδεσμεύεται η μνήμη δυναμικά.

I) mainThread
Στο mainThread δεσμεύεται χώρος για τα pthread_id των worker, η ουρά αρχικά, και ένα pthread_id και ένα MyThreadArgs για κάθε firstLineThread. Τα πρώτα 2 διαγράφονται στο terminator, το pthread_id στο mainThread και το MyThreadArgs στο firstLineThread (σε περίπτωση που ο πελάτης στείλει exit, το MyThreadArg περνάει στον Terminator και διαγράφεται εκεί).

II) firstLineThread
Στο firstLineThread δημιουργούνται τα MyINode, το mutex και ο μετρητής για τον κάθε πελάτη καθώς και τα jobs για την ουρά. Τα MyINode που περιέχουν φακέλους καταστρέφονται από το ίδιο thread, τα jobs από τα workerThreads και όλα τα υπόλοιπα από το τελευταίο job για τον πελάτη.