ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
K24: Προγραμματισμός Συστήματος – Εαρινό Εξάμηνο 2011
4η Προγραμματιστική Εργασία
Ημερομηνία Ανακοίνωσης: 14/6/11
Ημερομηνία Υποβολής: 14/7/11

Εισαγωγή στην Εργασία:

Η εργασία αυτή θα σας βοηθήσει να να εξοικειωθείτε με τον προγραμματισμό με threads και με την χρήση sockets για την επικοινωνία μεταξύ διεργασιών. Θα υλοποιήσετε μία *multithreaded* εφαρμογή (server) που επιτρέπει σε χρήστες με την βοήθεια ενός απλού client προγράμματος να κάνουν file sharing. Οι χρήστες θα μπορούν όχι μόνο να διαμοιράζονται αρχεία μεταξύ τους αλλά και να αλλάζουν το περιεχόμενο των εν λόγω αρχείων.

Διαδικαστικά:

Το αποτέλεσμα της εργασίας σας θα πρέπει να τρέχει στα μηχανήματα Linux/Unix της σχολής.

- Υπεύθυνοι για την άσκηση αυτή είναι (ερωτήσεις, αξιολόγηση, βαθμολόγηση, κτλ) είναι: ο κ. Αλέξανδρος Αντωνιάδης (grad1097), και ο κ. Γιάννης Γκερμπεσιώτης (grad1131).
- Η άσκηση μπορεί να γίνει είτε **ατομικά** ή από ομάδες **δύο ατόμων**.
- Παρακολουθείτε την ιστοσελίδα του μαθήματος για επιπρόσθετες ανακοινώσεις στο `http://www.di.uoa.gr/~ad/k24/index.html` και την ηλεκτρονική λίστα (mailman).

Βασική περιγραφή του filesystem-sharing της εργασίας:

Ο βασικός στόχος της άσκησης είναι η δημιουργία μιας διαδικτυακής υπηρεσίας που επιτρέπει σε χρήστες την αποθήκευση και διαμοιρασμό αρχείων με άλλους χρήστες στο διαδίκτυο χρησιμοποιώντας (ένα απλό) συγχρονισμό αρχείων.

Τα διαμοιραζόμενα αρχεία (αργά η γρήγορα) φιλοξενούνται σε ένα δικτυακό διακομιστή (server) με τον οποίο μπορούν να επικοινωνήσουν οι χρήστες μέσω ενός client προγράμματος. Ένας χρήστης για να ενημερωθεί και πιθανόν να παραλάβει (εν μέρει) τα περιεχόμενα ενός φακέλου στο server πρέπει πρώτα να συνδεθεί με τον διαμετακομιστή (δηλ. να κάνει join) και να λάβει μια ενημέρωση για τα περιεχόμενα του εν λόγω φακέλου. Για λόγους απλότητας, θεωρούμε ότι ο εξυπηρετητής περιλαμβάνει πάντα τη τελευταία ενημέρωση των αρχείων, αναγκάζοντας τον (client) του χρήστη να συγχρονίσει τα απαραίτητα τοπικά αρχεία με εκείνα του server. Κατόπιν, για οποιαδήποτε αλλαγή που πραγματοποιείται 'τοπικά' στα περιεχόμενα του φακέλου, είναι ευθύνη του client να ενημερώσει αντίστοιχα τον server για το/τα ενημερωμένα/αλλαγμένα αρχεία που πιθανώς υπάρχουν.

Περιοδικά, οι συνδεδεμένοι με τον server, clients θα πρέπει να ενημερώνονται για όσα αρχεία διαθέτουν τοπικά και για τα οποία έχει υπάρξει αλλαγή στον server. Λίστες τοπικών αρχείων θα πρέπει να συγκρίνονται με εκείνη του server ώστε να υπάρχουν οι κατάλληλες ενημερώσεις από τον server σε κάθε client.

Προκειμένου να αναγνωρίζονται μοναδικά τα αρχεία, αλλά και για να ελέγχεται αν έχουν υποστεί κάποιας μορφής αλλοίωση περιεχομένου (δηλ. corruption) κατά τη διαδικασία μεταφοράς τους, χρησιμοποιούνται MD5 sums.

Σενάριο Λειτουργίας Πρωτοκόλλου Εφαρμογής:

Αρχικά ας υποθέσουμε ότι ενας client θέλει να συνδεθεί με τον server και γνωρίζει εκ των προτέρων την ip διεύθυνση και port που ακούει για νέες συνδέσεις. Τότε, προκειμένου να συνδεθεί, ο client θα στείλει μία αίτηση τύπου `JoinRequest` στον (απομακρυσμένο) server όπως παρακάτω:

```
MessageType: JoinRequest\r\n
\r\n
```

Η ακολουθία χαρακτήρων `\r\n` υποδηλώνει αλλαγή γραμμής και είναι ένας τρόπος να ξεχωρίζει το πρωτόκολλο επικοινωνίας τα πεδία μιας αίτησης ή μιας απάντησης που θα χρησιμοποιηθούν. Κάθε client αλλά και ο server που επιθυμεί να επικοινωνήσει χρησιμοποιώντας αυτό το πρωτόκολλο, πρέπει να τερματίζει τις γραμμές του με την προαναφερθείσα ακολουθία χαρακτήρων και μόνο με αυτή. Ο καθορισμός μίας συγκεκριμένης ακολουθίας αλλαγής γραμμής έχει γίνει επειδή διαφορετικά λειτουργικά συστήματα χρησιμοποιούν διαφορετικές ακολουθίες αλλαγής γραμμής, για παράδειγμα τα Unix συστήματα χρησιμοποιούν μόνο έναν χαρακτήρα, τον `\n`. Ας σημειωθεί ότι η αίτηση που περιγράφηκε παραπάνω αποτελεί μία συνεχή ακολουθία χαρακτήρων, αλλά έχει τυπωθεί σε διαφορετικές γραμμές για λόγους αναγνωσιμότητας. Αποφύγετε λοιπόν να προσθέτετε επιπλέον χαρακτήρα αλλαγής γραμμής μετά την ακολουθία `\r\n` .

Η τελευταία γραμμή της αίτησης, όπως φαίνεται, είναι μία κενή γραμμή. Με αυτό τον τρόπο καταλαβαίνουν οι συμμετέχοντες στην επικοινωνία ότι η αίτηση έφτασε στο τέλος της και δεν χρειάζεται να διαβάσουν επιπλέον πληροφορία. Πιο συγκεκριμένα, τα μηνύματα του πρωτοκόλλου αποτελούνται από μία επικεφαλίδα (header) και ένα σώμα (body) και η ακολουθία `\r\n` υποδηλώνει το τέλος της επικεφαλίδας του μηνύματος. Το παραπάνω μήνυμα απλά συμβαίνει να μην έχει σώμα, όπως και άλλες αιτήσεις του πρωτοκόλλου. Ορισμένα μηνύματα απάντησης περιέχουν και σώμα, όπως για παράδειγμα ένα μήνυμα αποστολής αρχείου στο οποίο το σώμα του μηνύματος περιέχει τα δεδομένα του αρχείου. Παρακάτω θα περιγραφεί και ο τρόπος επεξεργασίας του σώματος των μηνυμάτων, καθώς δεν είναι ομοιόμορφος για όλα τα μηνύματα.

Συνεχίζοντας στο παράδειγμα, ο εξυπηρετητής server θα λάβει την αίτηση τύπου `JoinRequest` που του έστειλε ο client και θα καταχωρήσει τα απαραίτητα στοιχεία επικοινωνίας (διεύθυνση ip και θύρα port). Ο server από τη μεριά του θα απαντήσει με ένα μήνυμα τύπου `JoinResponse.` Το μήνυμα `JoinResponse` που αντιστοιχεί στην αίτηση που υποθέσαμε ότι έστειλε ο client φαίνεται παρακάτω:

```
MessageType: JoinResponse\r\n
\r\n
```

Εν συνεχεία, ο client αποστέλλει ένα μήνυμα τύπου `AskFileList` για να ενημερωθεί για τα περιεχόμενα του φακέλου που έχει στη διάθεση του ο server. Το μήνυμα έχει ως εξής:

```
MessageType: AskFileList\r\n
\r\n
```

Όποτε ο server αποκρίνεται με ένα μήνυμα τύπου `FileList` :

```
MessageType: FileList\r\n
\r\n
<FileName1 File-md5sum-1>\r\n
```

```
<FileName2 File-md5sum-2>\r\n
...
<FileNameN File-md5sum-N>\r\n
\r\n
```

Στο παραπάνω μήνυμα απάντησης, όπως φαίνεται, αποτελείται από την επικεφαλίδα που απλά περιγράφει το τύπο του μηνύματος, ενώ στο σώμα του μηνύματος παρατηρούμε μία λίστα ονομάτων αρχείων μαζί με το md5 sum του κάθε αρχείου.

Ο τύπος μηνύματος `GetFile` , του οποίου τη σύνταξη τη βλέπουμε παρακάτω, αποστέλλεται από τον client στο server για την ανάκτηση ενός συγκεκριμένου αρχείου.

```
MessageType: GetFile\r\n
Filename: <Filename>\r\n
File-md5sum: <md5 sum>\r\n
\r\n
```

Στο οποίο ο server αποκρίνεται με ένα μήνυμα τύπου `SendFile`, στη περίπτωση που διαθέτει το συγκεκριμένο αρχείο:

```
MessageType: GetFileResponse\r\n
ResponseStatus: FileFound\r\n
Filename: <Filename>\r\n
File-md5sum: <md5 sum>\r\n
FileSizeInBytes: <File size>\r\n
\r\n
<File data>
```

Το παραπάνω μήνυμα απάντησης, όπως φαίνεται, αποτελείται από μία επικεφαλίδα, η πρώτη γραμμή της οποίας πληροφορεί τον client ότι ο απομακρυσμένος server διαθέτει το αρχείο που του ζητήθηκε. Στις επόμενες γραμμές περιλαμβάνονται πληροφορίες (metadata) σχετικές με το αρχείο. Μετά την κενή γραμμή αρχίζει το σώμα του μηνύματος, το οποίο περιέχει τα δεδομένα που ζητήθηκαν. Ο client ξέρει πόσα bytes να διαβάσει από το πεδίο `FileSizeInBytes` στην τελευταία γραμμή της επικεφαλίδας. Το πεδίο αυτό χρησιμοποιείται για να γνωρίζει ο τοπικός client σε ποιο σημείο τελειώνουν τα δεδομένα του σώματος του μηνύματος.

Το πεδίο File-md5sum αντιστοιχεί σε ένα checksum ελέγχου του αν η διαδικασία λήψης ολοκληρώθηκε χωρίς σφάλματα. Αν συμβεί κάποιο πρόβλημα στην επικοινωνία και τα δεδομένα αλλάξουν, θα αλλάξει και το md5 sum και σε αυτή την περίπτωση ο client θα πρέπει να το αγνοήσει και να το ζητήσει ξανά κάποια στιγμή στο μέλλον. Το md5 sum στην περίπτωση των αρχείων χρησιμεύει και ως αναγνωριστικό των αρχείων. Ο συνδυασμός ονόματος αρχείου και md5 sum έχει πολύ μικρή πιθανότητα να είναι ίδιο για δύο διαφορετικά αρχεία (όχι μηδενική, αλλά το θεωρούμε αποδεκτό σχήμα ταυτοποίησης στα πλαίσια της άσκησης).

Μία ακόμα σημαντική λειτουργία, είναι αυτή της ενημέρωσης των αρχείων του server από τον client, κατά την οποία ο πρώτος στέλνει ένα μήνυμα τύπου `SendFile` στο δεύτερο ως εξής:

```
MessageType: SendFile\r\n
Filename: <Filename>\r\n
File-md5sum: <md5 sum>\r\n
FileSizeInBytes: <File size>\r\n
\r\n
<File Data>
```

Όταν ο server λάβει ένα αρχείο από έναν client , ενημερώνει κατάλληλα τις εσωτερικές δομές του, είτε αντικαθιστώντας τυχόν υπάρχον αρχείο με το ίδιο όνομα, είτε απλά προσθέτοντας το νέο αρχείο στη λίστα αρχείων αν πρόκειται για μοναδικό αρχείο. Τέλος, αποστέλλει ένα μήνυμα τύπου `SendFileResponse`:

```
MessageType: SendFileResponse\r\n
ResponseStatus: SendFileOK\r\n
Filename: <Filename>\r\n
File-md5sum: <md5 sum>\r\n
\r\n
```

Κατά τη διαγραφή ενός αρχείου από κάποιον client, αποστέλλεται στο server ο εξής τύπος μηνύματος:

```
MessageType: DeleteFile\r\n
Filename: <Filename>\r\n
File-md5sum: <md5 sum>\r\n
\r\n
```

και ο server αποκρίνεται με ένα μήνυμα τύπου `DeleteFileResponse` , όπως:

```
MessageType: DeleteFileResponse\r\n
ResponseStatus: DeleteFileOK\r\n
Filename: <Filename>\r\n
File-md5sum: <md5 sum>\r\n
\r\n
```

Ενδέχεται κάποιος client να ζητήσει να κάνει `SendFile` ή `DeleteFile` κάποιου αρχείου που εκείνη την στιγμή κάνει `GetFile` κάποιος client. Η αίτηση αυτή πρέπει να μπλοκαριστεί εως ότου τελειώσει η προηγούμενη διαδικασία. Το μπλοκάρισμα θα γίνει με thread locks.

Σε περίπτωση που υπήρξε κάποιο πρόβλημα με κάποια αίτηση το οποίο αφορά λανθασμένη σύνταξη ή ασυνέπεια δεδομένων, θα υπάρξει μία απάντηση με ένα μήνυμα της μορφής:

```
MessageType: ErrorMessage\r\n
ResponseStatus: BadRequest\r\n
\r\n
Bad Request\r\n
\r\n
```

Σε περίπτωση που ένας client στείλει `GetFile` στον server αλλά δεν υπάρχει το αρχείο αυτό:

```
MessageType: ErrorMessage\r\n
ResponseStatus: FileNotFound\r\n
\r\n
File not Found\r\n
\r\n
```

Επιπλέον, αν το αρχείο δεν είναι διαθέσιμο (π.χ. κάποιος client πραγματοποιεί `DeleteFile` από το server τη στιγμή που κάποιος κάνει `GetFile` του ίδιου αρχείου) τότε είτε ο server θα μπλοκάρει τη συγκεκριμένη ενέργεια όπως είπαμε πιο πάνω μέχρι να ολοκληρωθεί η προηγούμενη ενέργεια, είτε εναλλακτικά ο client θα λάβει απάντηση της μορφής:

```
MessageType: ErrorMessage\r\n
ResponseStatus: File not available\r\n
\r\n
```

```
File not available\r\n
\r\n
```

Όλα τα παραπάνω μηνύματα λάθους περιέχουν ένα σώμα το οποίο περιέχει μια λεκτική περιγραφή του μηνύματος, το οποίο τελειώνει με μία κενή γραμμή.

Επιπλέον Bonus:

Ένας client αντί για `GetFile` μπορεί να αιτείται `AskFile` από τον server. Ο server θα αναζητά στην λίστα με τους διαθέσιμους clients ποιος έχει το αρχείο και είναι διαθέσιμος ώστε να τον ανακατευθύνει τον αιτούντα ώστε να ζητήσει από τον πρώτο-διαθέσιμο στέλνοντας του την ip και την port του client αυτού. Σε περίπτωση που δεν υπάρχει τέτοιος client, τότε ο αιτούμενος client λαμβάνει την ip και το port του server.

Μόλις ο client λάβει το `AskFileResponse` κάνει αίτηση `GetFile` στην ip που αναφέρεται. Για τον λόγο αυτό οι clients θα πρέπει να υλοποιήσουν επιπλέον την διαδικασία upload ενός αρχείου ακούγοντας αιτήσεις όχι μόνο από τον server αλλά και από άλλους clients (multithreaded server).

Κατά τη σύνδεση του client με τον server, το μήνυμα JoinRequest διαφοροποιείται σε σχέση με πριν, καθότι ο client θα αποστείλει την port στην οποία θα δέχεται αιτήσεις.

```
MessageType: JoinRequest\r\n
Port: <PortNo>\r\n
\r\n
```

Με την προϋπόθεση ότι η αίτηση που έστειλε ο client ήταν επιτυχής, μετά την αποστολή της αίτησης `JoinResponse`, ο server κρατάει το κανάλι επικοινωνίας με τον client ανοικτό και στέλνει περιοδικά μηνύματα σηματοδοσίας στον client.

Τα μηνύματα τύπου `HeartBeat` τα οποία στέλνονται στους client ανά τακτά χρονικά διαστήματα, με σκοπό να ενημερωθεί σχετικά εγκαίρως στην περίπτωση που κάποιος από αυτούς αποσυνδεθεί από τον server. Τα μηνύματα αυτά θα πρέπει να διαβάζονται από τους clients χωρίς αυτοί να χρειάζεται να κάνουν κάποια επιπλέον ενέργεια.

Παρακάτω φαίνεται ένα μήνυμα τύπου `HeartBeat` :

```
MessageType: HeartBeat\r\n
\r\n
```

Η διαφορά σε σχέση με πριν, έγκειται στη διαδικασία ανάκτησης ενός αρχείου. Πριν ο client ζητήσει ένα αρχείο μέσω ενός μηνύματος τύπου `GetFile` θα πρέπει να αποστείλει αρχικά ένα μήνυμα τύπου `AskFile` ως εξής:

```
MessageType: AskFile\r\n
Filename: <Filename>\r\n
\r\n
```

η απάντηση του μηνύματος τύπου `AskFile` που περιγράψαμε προηγουμένως, θα είναι ένα μήνυμα τύπου `AskFileResponse`:

```
MessageType: AskFileResponse\r\n
Filename: <Filename>\r\n
IP: <IP>\r\n
Port: <Port>\r\n
\r\n
```

Ας σημειωθεί ότι τα στοιχεία που αποστέλλει server δεν είναι απαραίτητα απόλυτα ακριβή, καθώς στο ενδιάμεσο των περιοδικών ενημερώσεων υπάρχει περίπτωση να έχουν αποσυνδεθεί ορισμένοι clients. Αυτό θα πρέπει να το λάβετε υπόψη σας όταν στέλνετε αιτήσεις για λήψη αρχείων από άλλους clients.

Τέλος, στη περίπτωση που δεν υπάρχει άλλος συνδεδεμένος client είτε κανείς δεν έχει διαθέσιμο το ζητούμενο αρχείο, ο server θα αποστείλει τα δικά του στοιχεία και θα αναλάβει ο ίδιος να απαντήσει σε μελλοντικό `GetFile` μήνυμα.

Κατευθύνσεις υλοποίησης:

Θα γράψετε κώδικα σε C/C++ (χωρίς STL) που θα υλοποιεί τη συμπεριφορά τόσο του server αλλά και τον κώδικα που θα υλοποιεί τη συμπεριφορά του client. Το πρόγραμμα του client θα δέχεται στη γραμμή εντολής 1 όρισμα που θα είναι το configuration αρχείο του, με τα ακόλουθα πεδία:

1. Το μονοπάτι (path), απόλυτο ή σχετικό προς το φάκελο στον οποίο θα μεταβεί η διεργασία και θα αναλάβει το συγχρονισμό του.
2. Η IP του host στον οποίο εκτελείται ο server.
3. Η port όπου ακούει για TCP συνδέσεις ο server.
4. Η port όπου ακούει για TCP συνδέσεις ο client.

Ένα παράδειγμα configuration αρχείου για λειτουργία του client είναι το ακόλουθο:

```
WorkingDirectory /home/bob/folder
ServerIP 192.168.1.100
ServerPort 3456
ClientPort 3478
```

Το αντίστοιχο παράδειγμα configuration αρχείου για λειτουργία του server που έχει λιγότερα πεδία, είναι το ακόλουθο:

```
WorkingDirectory /home/bob/folder
ServerIP 192.168.1.100
ServerPort 3456
```

Το πρώτο πράγμα που θα πρέπει να κάνει ένας client είναι να αποστείλει την κατάλληλη αίτηση ένταξης στον server . Ύστερα, το πρόγραμμα του client, αφού λάβει ενημέρωση της λίστας αρχείων συγχρονίζει τα αρχεία του με αυτά του server.

Για τη λειτουργία εξυπηρέτησης των client από τον server, θα πρέπει να υιοθετηθεί το μοντέλο του multi-threaded server: Ο server κάθε φορά που δέχεται μία σύνδεση από κάποιον απομακρυσμένο client, θα αναθέτει την εξυπηρέτηση του κομματιού το οποίο αιτείται ο δεύτερος σε ένα νέο thread, το οποίο μπορεί να δημιουργηθεί εκ νέου ως detached thread ή να επιλεγεί μέσα από μία ομάδα (thread pool) ανενεργών threads.

Κάθε client αλλά και ο server διατηρεί μία λίστα για το σύνολο των αρχείων του φακέλου. Για λόγους

απλότητας, θα θεωρήσουμε ότι δεν υπάρχουν υποφάκελοι στον αρχικό φάκελο. Κοινώς, ο φάκελος συγχρονισμού θα περιλαμβάνει μόνο αρχεία. Η ελάχιστη πληροφορία που θα διατηρείται στη λίστα θα είναι το όνομα και το md5 check sum για κάθε αρχείο.

Η ορθή λήψη ενός αρχείου που ζητήθηκε θα πρέπει να επιβεβαιώνεται με τη βοήθεια του αλγόριθμου κατακερματισμού MD5. Θα σας δοθεί μία υλοποίηση όπου θα πρέπει να καλέσετε την κατάλληλη συνάρτηση υπολογισμού MD5 sums προκειμένου να επιβεβαιώσετε την ορθή λήψη ενός αρχείου. Εάν ένα αρχείο ληφθεί corrupted ο client θα επαναλαμβάνει την όλη διαδικασία από την αρχή.

Συνολικά, τα threads που θα χρειαστεί να διατηρήσετε δίνονται παρακάτω:

1. Server:
    - Το main thread, το οποίο λειτουργεί ως manager για τον multi-threaded server.
    - Multi-threaded server(detached threads ή pool of threads) που λειτουργεί ως εξυπηρετητής για τα αρχεία που δέχεται αιτήσεις.
    - (Bonus) 1 thread το οποίο thread παράγει μηνύματα heart beat με μόνο στόχο την υλοποίηση ενός μηχανισμού εντοπισμού των clients. Το thread πρέπει να ενημερώνει τη λίστα που κρατάει τοπικά. Αποτυχία λήψης του heart beat υπονοεί ότι ο εν λόγω (clent) είναι εκτός λειτουργίας και ακολούθως ο server θα ενημερώσει κατάλληλα τη λίστα με τους εν ενεργεία clients.
2. Client:
    - Το main thread, το οποίο λειτουργεί ως manager για τον multi-threaded client.
    - Ένα thread το οποίο θα ελέγχει περιοδικά τα περιεχόμενα του υπό συγχρονισμό φακέλου και συγκρίνοντας με τη λίστα που έχει κατασκευάσει προηγουμένως ελέγχει για τυχών αλλαγές και πραγματοποιεί τις απαραίτητες ενέργειες ώστε να συμφωνεί με τα περιεχόμενα του server.
    - (Bonus) Multi-threaded server(detached threads ή pool of threads) που λειτουργεί ως εξυπηρετητής για τα αρχεία που δέχεται αιτήσεις.
    - (Bonus) 1 thread το οποίο thread καταναλώνει μηνύματα heart beat χωρίς να αντιδρά σε αυτά.

Όπως περιγράφηκε και παραπάνω, η εφαρμογή σας θα πρέπει να αγνοεί τυχόν corrupted files και να επαναλαμβάνει τις αιτήσεις μέχρι να τα λάβει επιτυχώς. Αφού τα λάβει επιτυχώς, πρέπει να υπολογίσει και το md5 sum του αρχείου, το οποίο αναμένεται να συμφωνεί με το md5 sum που έχει λάβει από την προηγούμενη επικοινωνία με τον server. Αν κάτι τέτοιο όμως δεν ισχύει, η εφαρμογή θα πρέπει να σβήνει το αρχείο από το δίσκο και να ενημερώνει τον χρήστη με κατάλληλο μήνυμα λάθους στο standard output.

Η εφαρμογή σας θα πρέπει να τυπώνει κάποιες πληροφορίες στο standard output και σε δύο αρχεία με ονόματα clientLog.txt και serverLog.txt. Τα δύο αυτά αρχεία θα είναι log files στα οποία θα αποθηκεύονται λεπτομερείς πληροφορίες σχετικές με τη διαμεταγωγή των δεδομένων.

Τι πρέπει να Παραδοθεί:

1. Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματος σας (1-2 σελίδες ASCII κειμένου είναι αρκετές).
2. Ένα tar file με όλη σας τη δουλειά σε έναν κατάλογο που πιθανώς να φέρει το όνομά σας και θα περιέχει όλη σας τη δουλειά.

Άλλες Σημαντικές Παρατηρήσεις:

1. Οι εργασίες είναι είτε **ατομικές** ή μπορούν να γραφτούν από ομάδες των **δυο ατόμων**.
2. Όποιος υποβάλλει/δείχνει κώδικα που δεν έχει γραφτεί από την ίδια/ίδιο **μηδενίζεται** στο μάθημα.
3. Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πώς θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιαδήποτε μορφής) είναι κάτι που **δεν επιτρέπεται** και δεν πρέπει να γίνει. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικα *απλά παίρνει μηδέν* στο μάθημα. Αυτό ισχύει για **όλους όσους εμπλέκονται** ανεξάρτητα από το ποιος έδωσε/πήρε κλπ.
4. Το πρόγραμμα σας θα πρέπει να τρέχει σε Ubuntu-Linux ή Solaris αλλιώς **δεν θα βαθμολογηθεί**.
5. Σε καμιά περίπτωση τα MS-Windows **δεν είναι επιλογή** πλατφόρμας για την παρουσίαση αυτής της άσκησης.

Appendix: Σύνοψη του πρωτοκόλου επικοινωνίας:

Όπως αναφέρθηκε, όλα τα μηνύματα διαθέτουν επικεφαλίδα της οποίας οι γραμμές διαχωρίζονται με την ακολουθία `\r\n` και η οποία τελειώνει με μία κενή γραμμή (η οποία αποτελείται μόνο από την ακολουθία `\r\n`). Ορισμένα μηνύματα απάντησης έχουν και σώμα, το οποίο τελειώνει με μια κενή γραμμή, εκτός από την περίπτωση του μηνύματος και `SendFile` , στα οποία το μέγεθος του σώματος ορίζεται στο πεδίο επικεφαλίδας `FileSizeInBytes` .

Παρακάτω παρουσιάζεται η γενική μορφή όλων των μηνυμάτων:

1. `JoinRequest`

```
MessageType: JoinRequest\r\n
\r\n
```

2. `JoinResponse`

```
MessageType: JoinResponse\r\n
\r\n
```

3. `AskFileList`

```
MessageType: AskFileList\r\n
\r\n
```

4. `FileList`

```
MessageType: FileList\r\n
\r\n
<FileName1 File-md5sum-1>\r\n
<FileName2 File-md5sum-2>\r\n
...
<FileNameN File-md5sum-N>\r\n
\r\n
```

5. `GetFile`

```
MessageType: GetFile\r\n
Filename: <Filename>\r\n
File-md5sum: <md5 sum>\r\n
\r\n
```

6. `GetFileResponse`

```
MessageType: GetFileResponse\r\n
ResponseStatus: FileFound\r\n
Filename: <Filename>\r\n
File-md5sum: <md5 sum>\r\n
FileSizeInBytes: <File size>\r\n
\r\n
<File data>
```

7. `SendFile`

```
MessageType: SendFile\r\n
Filename: <Filename>\r\n
File-md5sum: <md5 sum>\r\n
FileSizeInBytes: <File size>\r\n
\r\n
<File Data>
```

8. `SendFileResponse`

```
MessageType: SendFileResponse\r\n
ResponseStatus: SendFileOK\r\n
Filename: <Filename>\r\n
File-md5sum: <md5 sum>\r\n
\r\n
```

9. DeleteFile

```
MessageType: DeleteFile\r\n
Filename: <Filename>\r\n
File-md5sum: <md5 sum>\r\n
\r\n
```

10. DeleteFileResponse

```
MessageType: DeleteFileResponse\r\n
ResponseStatus: DeleteFileOK\r\n
Filename: <Filename>\r\n
File-md5sum: <md5 sum>\r\n
\r\n
```

11. ErrorMessage

```
MessageType: ErrorMessage\r\n
ResponseStatus: BadRequest\r\n
\r\n
Bad Request\r\n
\r\n
```

12. ErrorMessage

```
MessageType: ErrorMessage\r\n
ResponseStatus: FileNotFound\r\n
\r\n
File not Found\r\n
\r\n
```

13. ErrorMessage

```
MessageType: ErrorMessage\r\n
ResponseStatus: File not available\r\n
\r\n
File not available\r\n
\r\n
```

14. HeartBeat

```
MessageType: HeartBeat\r\n
\r\n
```

15. AskFile

```
MessageType: AskFile\r\n
Filename: <Filename>\r\n
\r\n
```

16. AskFileResponse

```
MessageType: AskFileResponse\r\n
Filename: <Filename>\r\n
IP: <IP>\r\n
Port: <Port>\r\n
\r\n
```