Σκοπός

Ο στόχος αυτής της εργασίας είναι να εξοικειωθείτε με τον προγραμματισμό συστήματος σε Unix περιβάλλον και συγκεκριμένα, με τη δημιουργία νημάτων και τη δικτυακή επικοινωνία.

Θα υλοποιήσετε ένα πρόγραμμα που έχει ως σκοπό την αντιγραφή όλων των περιεχομένων ενός καταλόγου (directory) αναδρομικά από έναν εξυπηρετητή (server), στο τοπικό σύστημα αρχείων (file system) του πελάτη (client).

Ο server θα πρέπει να είναι σε θέση να διαχειρίζεται αιτήματα από διαφορετικούς πελάτες ανά πάσα χρονική στιγμή και να επεξεργάζεται κάθε αίτημα παράλληλα, σπάζοντάς το σε ανεξάρτητες πράξεις αντιγραφής αρχείων. Αντίστοιχα, ο πελάτης θα πρέπει να επεξεργάζεται τα δεδομένα που στέλνονται από τον server και τελικά, να δημιουργεί ένα τοπικό αντίγραφο του καταλόγου που αιτήθηκε από το server. Ο κατάλογος αυτός εσωτερικά θα πρέπει να περιέχει ακριβώς την ίδια δομή και αρχεία με εκείνη του server.

Οι βασικές οντότητες του σχεδιασμού είναι ο εξυπηρετητής (server) και ο πελάτης (client). Πολλοί clients μπορούν να συνδεθούν ταυτόχρονα σε έναν server. Η επικοινωνία μεταξύ τους γίνεται με sockets.

Εξυπηρετητής (**Server):** Ο server, θα ονομάζεται `dataServer`. Κατά την εκκίνηση του, θα περιμένει για συνδέσεις από τους clients σε μία προκαθορισμένη θύρα (port) που θα του παρέχεται ως όρισμα. Ο εξυπηρετητής δημιουργεί δύο ειδών threads. Τα threads επικοινωνίας (communication threads) και τα theads εργασίας (worker threads).

Ο server θα δημιουργεί ένα νέο communication thread για κάθε νέα σύνδεση που δέχεται, το οποίο θα είναι υπεύθυνο να διαβάσει από τον πελάτη το όνομα του καταλόγου προς αντιγραφή. Ο κατάλογος προς αντιγραφή θα διαβάζεται από το τοπικό file system του server αναδρομικά μέχρι να εξαντληθουν όλα τα περιεχόμενα σε αυτόν αρχεία. Κάθε αρχείο της ιεραρχίας θα τοποθετείται σε μία ουρά εκτέλεσης που θα είναι κοινή για όλα τα threads. Η ουρά εκτέλεσης έχει έναν αριθμό θέσεων ο οποίος δίνεται ως ορισμα κατά την εκτέλεση. Στην περίπτωση που η ουρά εκτέλεσης είναι γεμάτη, το communication thread που εξυπηρετεί τον πελάτη θα πρέπει να περιμένει μέχρι να δημιουργηθεί ελεύθερος χώρος (δηλαδή κάποιο worker thread - το οποίο θα περιγράψουμε πιο κάτω - να αφαιρέσει κάποιο αρχείο από την ουρά εκτέλεσης.) Στην ουρά εκτέλεσης ΔΕΝ περιέχονται τα δεδομένα των αρχείων.

Ο server θα διαθέτει ένα thread pool με worker threads. Το μέγεθος του thread pool ορίζεται με όρισμα κατά την εκτέλεση. Το thread pool θα αναθέτει σε ένα worker thread, ένα αρχείο προς ανάγνωση. Σε περίπτωση που η ουρά εκτέλεσης είναι κενή, τα worker threads θα πρέπει να περιμένουν μέχρι να υπάρξει κάποια εγγραφή στην ουρά. Κάθε worker thread είναι υπεύθυνο να διαβάσει τα περιεχόμενα του αρχείου και να τα στείλει στον κατάλληλο πελάτη, μέσω του socket που επιστρέφει η accept κλήση για την επικοινωνία με τον πελάτη. Το αρχείο, θα διαβάζεται και θα αποστέλλεται στον πελάτη ανά μπλοκ. Το μέγεθος του μπλοκ δίνεται σε όρισμα από το command line, σε bytes. Τέλος, ο server θα πρέπει να εξασφαλίζει κατά τη διάρκεια εκτέλεσης του, ότι στο socket κάθε πελάτη γράφει δεδομένα μόνο ένα worker thread τη φορά, μέχρι να γράψει ολόκληρο το αρχείο που του αντιστοιχεί, παρόλο που τα writes στο socket γίνονται ένα block τη φορά.

Πελάτης (**Client):** Στο σύστημα μπορεί να είναι ενεργοί περισσότεροι από ένας πελάτες ταυτόχρονα. Κάθε πελάτης**,** που θα ονομάζεται `remoteClient`, συνδέεται με το server σε θύρα που ορίζεται ως όρισμα, και προσδιορίζει (σε όρισμα) το όνομα του καταλόγου που θέλει να αντιγράψει. Έπειτα, για κάθε αρχείο που περιέχεται στον κατάλογο, του επιστρέφεται από τον server α) όνομα του αρχείου β) οτιδήποτε metadata του αρχείου και γ) τα περιεχόμενα του αρχείου τα οποία θα χρησιμοποιήσει για να γράψει το τοπικό αρχείο. Το όνομα του αρχείου είναι σε μορφή μονοπατιού, οπότε ο πελάτης πρέπει να δημιουργήσει τους κατάλληλους καταλόγους και υποκαταλόγους.. Σε περίπτωση που ένα αρχείο υπάρχει ήδη στο τοπικό σύστημα αρχείων του πελάτη, το αρχείο αυτό θα πρέπει να

διαγράφεται και δημιουργείται από την αρχή (όχι να γραφτεί από πάνω)

Συμβάσεις:

- Θεωρούμε πως ο πελάτης γνωρίζει την ιεραρχία του συστήματος αρχείων του server και μπορεί να επιλέξει οποιοδήποτε κατάλογο για αντιγραφή.
- Θεωρούμε πως το σύστημα αρχείων του server δεν περιέχει άδειους καταλόγους.

Command Line Arguments: Ο server θα καλείται από τη γραμμή εντολών ως εξής:

```
./dataServer -p <port> -s <thread_pool_size> -q <queue_size> -b <block_size>
```

Όπου:

1. *port*: Η θύρα στην οποία ο server θα ακούει για εξωτερικές συνδέσεις.
2. *thread pool size*: Ο αριθμός των worker threads στο thread pool.
3. *queue size*: Ο αριθμός θέσεων στην ουρά εκτέλεσης.
4. *block size*: Το μέγεθος των μπλοκ των αρχείων σε bytes που θα στέλνουν οι worker threads.

Ο πελάτης θα καλείται από τη γραμμή εντολών ως εξής:

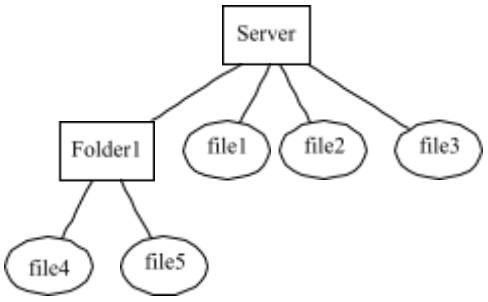
```
./remoteClient -i <server_ip> -p <server_port> -d <directory>
```

΄Οπου:

1. *server ip*: Η διεύθυνση **IP** που χρησιμοποιεί ο server.
2. *server port*: Η θύρα στην οποία ακούει ο server για εξωτερικές συνδέσεις.
3. *directory*: Ο κατάλογος προς αντιγραφή (ένα σχετικό μονοπάτι).

Τα ορίσματα της γραμμής εντολών είναι υποχρεωτικά και για τα δύο εκτελέσιμά προγράμματα. Επιπλέον, τα ζευγάρια ορισμάτων μπορεί να δίνονται σε διαφορετική σειρά από αυτή της εκφώνησης.

Παράδειγμα Εκτέλεσης: Στην παράγραφο αυτή περιγράφουμε ένα ενδεικτικό παράδειγμα εκτέλεσης. Ο server έχει την παρακάτω ιεραρχία:



Αν εκτελέσουμε το αίτημα ενός πελάτη που ζητά να αντιγράψει όλη την ιεραρχία παρατηρούμε ότι διαφορετικά worker threads διαβάζουν και αποστέλλουν τα δεδομένα των αρχείων στον πελάτη:

```
Server’s parameters are:
port: 12500
```

```
thread_pool_size:  2
queue_size: 2
Block_size: 512
Server was successfully initialized...
Listening for connections to port 12500
Accepted connection from localhost

[Thread: 140069174429440]: About to scan directory Server
[Thread: 140069174429440]: Adding file Server/file1 to the queue…
[Thread: 140069174429440]: Adding file Server/Folder1/file5 to the queue...
[Thread: 140069174429440]: Adding file Server/Folder1/file4 to the queue...
[Thread: 140069193361152]: Received task: <Server/file1, 4>
[Thread: 140069193361152]: About to read file Server/file1
[Thread: 140069184968448]: Received task: <Server/Folder1/file5, 4>
[Thread: 140069184968448]: About to read file Server/Folder1/file5
[Thread: 140069174429440]: Adding file Server/file2 to the queue...
[Thread: 140069174429440]: Adding file Server/file3 to the queue...
[Thread: 140069193361152]: Received task: <Server/Folder1/file4, 4>
[Thread: 140069193361152]: About to read file Server/Folder1/file4

[Thread: 140069193361152]: Received task: <Server/file2, 4>
[Thread: 140069193361152]: About to read file Server/file2
[Thread: 140069193361152]: Received task: <Server/file3, 4>
[Thread: 140069193361152]: About to read file Server/file3
```

```
Client’s parameters are:
serverIP: 127.0.0.1
port: 12500
directory: Server
Connecting to 127.0.0.1 on port 12500
Received: Server/file1
Received: Server/Folder1/file4
Received: Server/file2
Received: Server/file3
Received: Server/Folder1/file5
```

Διαδικαστικά

- Το πρόγραμμά σας θα πρέπει να τρέχει στα μηχανήματα Linux/Unix της σχολής. Κώδικας που δε μεταγλωττίζεται εκεί, θεωρείται ότι δεν μεταγλωττίζεται.
- Για επιπρόσθετες ανακοινώσεις, παρακολουθείτε το forum του μαθήματος στο piazza.com. Η πλήρης διεύθυνση είναι https://piazza.com/uoa.gr/spring2022/k24/home.
Η παρακολούθηση του φόρουμ στο Piazza είναι υποχρεωτική.
- Το πρόγραμμά σας θα πρέπει να γραφεί σε C (ή C++). 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.

- Η υποβολή θα γίνει μέσω eclass.
- Ο κώδικάς σας θα πρέπει να κάνει compile στα εκτελέσιμα `dataServer/remoteClient` όπως **ακριβώς** ορίζει η άσκηση και να έχει τις ίδιες παραμέτρους όπως ακριβώς ορίζει η άσκηση.

Τι πρέπει να παραδοθεί

- Όλη η δουλειά σας (πηγαίος κώδικας, Makefile και README) σε ένα tar.gz file με ονομασία OnomaEponymoProject2.tar.gz. Προσοχή να υποβάλετε μόνο κώδικα, Makefile, README και όχι τα binaries. Η άσκησή σας θα γίνει compile από την αρχή πριν βαθμολογηθεί.
- Όποιες σχεδιαστικές επιλογές κάνετε, θα πρέπει να τις περιγράψετε σε ένα README (απλό text αρχείο) που θα υποβάλετε μαζί με τον κώδικά σας. Το README χρειάζεται να περιέχει μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στον 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.
- Ο κώδικας που θα υποβάλετε θα πρέπει να είναι δικός σας. Απαγορεύεται η χρήση κώδικα που δεν έχει γραφεί από εσάς.
- Καλό θα είναι να έχετε ένα backup .tar της άσκησής σας όπως ακριβώς αυτή υποβλήθηκε σε κάποιο εύκολα προσπελάσιμο μηχάνημα (server του τμήματος, github, cloud).
- 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/tar.gz ή tar/tar.gz που έχουν λάθος και δε γίνονται extract δε βαθμολογούνται**.

Τι θα βαθμολογηθεί

- Η συμμόρφωση του κώδικά σας με τις προδιαγραφές της άσκησης.
- Η οργάνωση και η αναγνωσιμότητα (μαζί με την ύπαρξη σχολίων) του κώδικα.
- Η χρήση Makefile και το separate compilation.

Άλλες σημαντικές παρατηρήσεις

- Οι εργασίες είναι ατομικές.
- Όποιος υποβάλλει / δείχνει 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. Τονίζουμε πως θα πρέπει να λάβετε τα κατάλληλα μέτρα ώστε να είναι προστατευμένος ο κώδικάς σας και να μην αποθηκεύεται κάπου που να έχει πρόσβαση άλλος χρήστης (π.χ., η δικαιολογία «Το είχα βάλει σε ένα github repo και μάλλον μου το πήρε από εκεί», δεν είναι δεκτή.)
- 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.
- Το πρόγραμμά σας θα πρέπει να γραφτεί σε C ή C++. Μπορείτε να χρησιμοποιήσετε μόνο

εντολές οι οποίες είναι διαθέσιμες στα μηχανήματα Linux του τμήματος.