ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
K24: Προγραμματισμός Συστήματος
Εαρινό Εξάμηνο 2014
3η Προγραμματιστική Εργασία

Ημερομηνία Υποβολής: 13 Ιουλίου 2014

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) 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 το 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 με εκείνη του server.

Εξυπηρετητής (Server): Ο server, έστω *dataServer*, κατά την εκκίνηση του, θα περιμένει για εξωτερικές συνδέσεις σε μία προκαθορισμένη θύρα (port). Για κάθε νέα σύνδεση που δέχεται, ο server θα δημιουργεί ένα νέο thread, το οποίο θα είναι υπεύθυνο να διαβάσει από τον πελάτη τον κατάλογο προς αντιγραφή. Ο κατάλογος προς αντιγραφή θα διαβάζεται αναδρομικά μέχρι να βρεθούν όλα τα περιεχόμενα σε αυτόν αρχεία. Κάθε αρχείο της ιεραρχίας θα τοποθετείται σε μία ουρά εκτέλεσης. Η ουρά εκτέλεσης θεωρείται ότι έχει ένα μέγιστο μέγεθος. Σε περίπτωση που η ουρά εκτέλεσης είναι γεμάτη, το thread που εξυπηρετεί τον πελάτη θα πρέπει να περιμένει μέχρι κάποιο worker thread να αφαιρέσει κάποιο αρχείο από την ουρά εκτέλεσης.

Επιπλέον, ο server θα διαθέτει ένα thread pool σταθερού μεγέθους, το οποίο θα αναθέτει ένα αρχείο προς ανάγνωση σε ένα thread. Σε περίπτωση που η ουρά εκτέλεσης είναι κενή, τα worker threads θα πρέπει να περιμένουν. 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 ισούται με το προκαθορισμένο μέγεθος σελίδας του λειτουργικού συστήματος.

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): Ενας πελάτης, έστω *remoteClient*, συνδέεται με το server μέσω μίας προκαθορισμένης θύρας και προσδιορίζει το όνομα του προς αντιγραφή καταλόγου. Έπειτα, για κάθε αρχείο, διαβάζει το όνομα του και οτιδήποτε metadata στείλει ο server. Το όνομα του αρχείου είναι σε μορφή μονοπατιού, οπότε ο πελάτης πρέπει να δημιουργήσει τους κατάλληλους καταλόγους. Επιπλέον, για κάθε αρχείο, ο πελάτης πρέπει να διαβάσει τα δεδομένα που θα στείλει ο server και να τα γράφει στο τοπικό του αντίγραφο. Σε περίπτωση που ένα αρχείο υπάρχει ήδη στο τοπικό σύστημα αρχείων του πελάτη, το αρχείο αυτό διαγράφεται και δημιουργείται από την αρχή.

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>
```

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*: Το μέγεθος της ουράς εκτέλεσης.

Ο πελάτης θα καλείται από τη γραμμή εντολών ως εξής:

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

Όπου:

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.

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

Διαδικαστικά:

– Το πρόγραμμά σας θα πρέπει να τρέχει στα μηχανήματα Linux/Unix της σχολής.
– Υπεύθυνοι για την άσκηση αυτή (ερωτήσεις, αξιολόγηση, βαθμολόγηση, κτλ) είναι όλοι οι βοηθοί. Ονόματα και email θα βρείτε στην ιστοσελίδα του μαθήματος.
– Για επιπρόσθετες ανακοινώσεις, παρακολουθείτε το forum του μαθήματος στο piazza.com για να δείτε ερωτήσεις/απαντήσεις/διευκρινίσεις που δίνονται σχετικά με την άσκηση. Η παρακολούθηση του φόρουμ στο piazza.com είναι *υποχρεωτική*.
– Η προφορική εξέταση της άσκησης θα γίνει την Πέμπτη και Παρασκευή μετά την προθεσμία παράδοσης. Προθεσμία για δέσμευση ώρας (slot) για προφορική εξέταση είναι τα μεσάνυχτα Τετάρτης προς Πέμπτη πριν την εξέταση. Καθυστερημένη δέσμευση ώρας έχει σαν συνέπεια ποινή ή άρνηση εξέτασης.

Τι πρέπει να παραδοθεί:

1. Όλη η δουλειά σας σε ένα tar-file που να περιέχει όλα τα source files, header files, Makefile.
2. Μια σύντομη περιγραφή (1–2 σελίδες) για τις επιλογές που κάνατε στο σχεδιασμό της άσκησης, σε μορφή PDF.
3. Οποιαδήποτε πηγή πληροφορίας, συμπεριλαμβανομένου και κώδικα που μπορεί να βρήκατε στο Διαδίκτυο θα πρέπει να αναφερθεί και στον πηγαίο κώδικά σας αλλά και στην παραπάνω αναφορά.

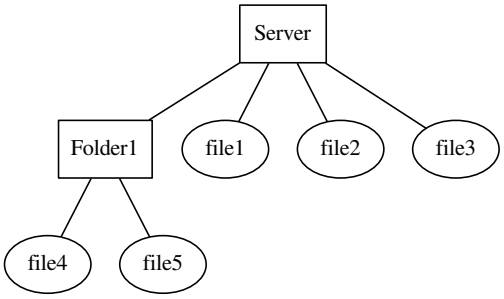
Τι θα βαθμολογηθεί:

1. Η συμμόρφωση του κώδικά σας με τις προδιαγραφές της άσκησης.
2. Η οργάνωση και η αναγνωσιμότητα (μαζί με την ύπαρξη σχολίων) του κώδικα.
3. Η χρήση Makefile και η κομματιαστή σύμβολο-μετάφραση (separate compilation).
4. Η αναφορά που θα γράψετε και θα υποβάλετε μαζί με τον πηγαίο κώδικα σε μορφή PDF.

Άλλες σημαντικές παρατηρήσεις:

1. Οι εργασίες είναι **ατομικές**.
2. Όποιος υποβάλλει / δείχνει κώδικα που δεν έχει γραφτεί από την ίδια/τον ίδιο **μηδενίζεται** στο μάθημα.
3. Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πως θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, *αντιγραφή κώδικα* (οποιασδήποτε 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 πρόγραμμά σας θα πρέπει να γραφτεί σε C (ή C++ χωρίς όμως STL extensions).

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 στη μεριά του πελάτη:

```
Server's parameters are:
        port: 12500
        thread_pool_size: 2
        queue_size: 2
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#33188#4096
Received: Server/Folder1/file4#33188#4096
Received: Server/file2#33188#4096
Received: Server/file3#33188#4096
Received: Server/Folder1/file5#33188#4096
```