ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

Τμήμα Πληροφορικής

ΕΠΛ 371 - Προγραμματισμός Συστημάτων

ΑΣΚΗΣΗ 4 – Ανάπτυξη Πολυνηματικού Εξυπηρετητή Ιστού για το Πρωτόκολλο HTTP/1.1

Διδάσκων: Δημήτρης Ζεϊναλιπούρ

Υπεύθυνος Εργαστηρίου: Παύλος Αντωνίου

Ημερομηνία Ανάθεσης: Τετάρτη 8/04/15
Ημερομηνία Παράδοσης: Τετάρτη 29/04/15 και ώρα 12:00 (21 μέρες)

(η λύση να υποβληθεί σε zip μέσω του Moodle και ο κώδικας να να παραδοθεί εκτυπωμένος στο εργαστήριο)

http://www.cs.ucy.ac.cy/courses/EPL371

I. Στόχος Άσκησης

Στόχος αυτής της εργασίας είναι η εξοικείωση με προχωρημένες τεχνικές προγραμματισμού διεργασιών, δια-διεργασιακής επικοινωνίας μέσω υποδοχών TCP/IP και πολυνηματικών εφαρμογών στη γλώσσα C. Ένας δεύτερος στόχος είναι να σας δοθεί η ευκαιρία να δουλέψετε ομαδικά για να υλοποιήσετε ένα ολοκληρωμένο σύστημα το οποίο θα κριθεί βάση της *ορθότητας*, *δομής* και *επίδοσής* του. Ένας τελευταίος στόχος είναι να σας δώσει την δυνατότητα να δουλέψετε με κάποια εργαλεία ανάπτυξης λογισμικού, όπως το Ολοκληρωμένο Περιβάλλον Ανάπτυξης Λογισμικού eClipse IDE for C/C++ και το σύστημα εκδόσεων SVN.

II. Περιγραφή

Αντικείμενο της άσκησης είναι να αναπτύξετε ένα «πολυνηματικό» εξυπηρετητή ιστού (web server) για το πρωτόκολλο HTTP, ο οποίος θα υποστηρίζει ένα βασικό υποσύνολο της έκδοσης 1.1 του πρωτοκόλλου. Η έκδοση 1.1 του πρωτοκόλλου HTTP περιγράφεται στο τεχνικό άρθρο Request For Comments 2616: http://www.rfc-editor.org/rfc/rfc2616.txt.[*]

[*] Στη συνέχεια, η έκφραση [RFC2616/§p] θα αναφέρεται στην παράγραφο p αυτού του εγγράφου.

Ευχαριστίες: Η εκφώνηση της άσκησης είναι, σε μεγάλο βαθμό, βασισμένη σε αντίστοιχη άσκηση που έχει προετοιμαστεί από τον Επίκουρο Καθ. Παναγιώτη Σταματόπουλο στο Καποδιστριακό Πανεπιστήμιο Αθηνών.

Όταν χρησιμοποιούμε ένα φυλλομετρητή σελίδων (browser), όπως ο Mozilla, ο Internet Explorer, κλπ, για να περιηγηθούμε στο WWW, δίνουμε ένα URL της μορφής http://<host>/<path>. Αυτό που κάνει τότε ο φυλλομετρητής είναι να συνδεθεί μέσω υποδοχών ροής στην προκαθορισμένη θύρα (80) για την HTTP υπηρεσία του μηχανήματος <host> και να υποβάλει ένα αίτημα, σε αυστηρά καθορισμένη μορφή που θα περιγραφεί στη συνέχεια, να του αποσταλούν από το web server κάποιες πληροφορίες για το αρχείο που βρίσκεται στη θέση /<path>, σε σχέση με τον κατάλογο-ρίζα του ιεραρχικού συστήματος αρχείων που «σερβίρει», καθώς και το ίδιο το αρχείο, για να το παρουσιάσει ο φυλλομετρητής στο χρήστη με τρόπο που κρίνει πρόσφορο. Αν ο web server δεν περιμένει αιτήσεις σύνδεσης από προγράμματα-πελάτες (clients) στην προκαθορισμένη θύρα 80, αλλά στην <port>, τότε το URL που έπρεπε να δώσουμε στο φυλλομετρητή θα ήταν http://<host>:<port>/<path>.
Ο πελάτης και ο web server επικοινωνούν μεταξύ τους ανταλλάσοντας μηνύματα με την εξής βασική μορφή [RFC2616/§4.1]:

- Αρχικά υπάρχει μια ακριβώς γραμμή, η οποία ορίζει το είδος της λειτουργίας που σχετίζεται με το αποστελλόμενο μήνυμα. Πρόκειται για μια *γραμμή αίτησης*, αν το μήνυμα αποστέλλεται από τον πελάτη προς το web server, ή μια *γραμμή κατάστασης*, αν το μήνυμα είναι απάντηση του web server προς τον πελάτη.

- Ακολουθούν ένα πλήθος από *γραμμές επικεφαλίδας*.

- Μετά υπάρχει μια ακριβώς κενή γραμμή, η οποία διαχωρίζει τις γραμμές επικεφαλίδας από το σώμα του μηνύματος.

- Τέλος, ακολουθούν τα bytes του σώματος του μηνύματος, αν υπάρχουν.

Οι γραμμές που αναφέρονται παραπάνω χωρίζονται μεταξύ τους από την ακολουθία "\r\n", δηλαδή από δυο χαρακτήρες, τον "\r" (Carriage Return) με ASCII κωδικό 13 και τον "\n" (Line feed) με ASCII κωδικό 10.

Κάθε γραμμή επικεφαλίδας αποτελείται από το όνομα της επικεφαλίδας, το χαρακτήρα ":" και την τιμή της επικεφαλίδας [RFC2616/§4.2]. Ακολουθεί παράδειγμα:

```
Content-Type: text/html
```

Αν το αποστελλόμενο μήνυμα είναι αίτηση του πελάτη προς το web server, η αρχική γραμμή του (*γραμμή αίτησης*) αποτελείται από μια λέξη η οποία καθορίζει την επιθυμητή λειτουργία (ή μέθοδο) που αιτείται ο πελάτης, ένα κενό χαρακτήρα, τη διαδρομή του αρχείου πάνω στο οποίο θα εφαρμοσθεί η μέθοδος αυτή, ένα ακόμα κενό χαρακτήρα, τη συμβολοσειρά "HTTP/" και τέλος την έκδοση του HTTP πρωτοκόλλου (για την άσκηση πάντα "1.1") [RFC2616/§5.1]. Ακολουθεί παράδειγμα:

```
GET /mydirectory/myfile.html HTTP/1.1
```

Αν το αποστελλόμενο μήνυμα είναι απάντηση από το web server, η αρχική γραμμή του (*γραμμή κατάστασης*) αποτελείται από τη συμβολοσειρά "HTTP/" και την έκδοση του HTTP πρωτοκόλλου (για την άσκηση πάντα "1.1"), ένα κενό χαρακτήρα, τον κωδικό κατάληξης της αίτησης και τέλος μια συμβολοσειρά που είναι η λεκτική περιγραφή του κωδικού κατάληξης [RFC2616/§6.1]. Ακολουθούν παραδείγματα:

```
HTTP/1.1 200 OK
HTTP/1.1 404 Not Found
HTTP/1.1 501 Not Implemented
```

Για τις ανάγκες της άσκησης, πρέπει να μπορείτε να χειριστείτε αιτήσεις των μεθόδων **GET** [RFC2616/§9.3], **HEAD** [RFC2616/§9.4] και **DELETE** [RFC2616/§9.7]. Αν σε μια αίτηση σας ζητηθεί οποιαδήποτε άλλη μέθοδος που δεν είναι υλοποιημένη, πρέπει να απαντήσετε σε αυτή με ένα μήνυμα απάντησης που θα έχει κωδικό κατάληξης της αίτησης το 501 και περιγραφή "Not Implemented" [RFC2616/§10.5.2]. Ακολουθεί παράδειγμα:

```
HTTP/1.1 501 Not Implemented
Server: my_webserver
Connection: close
Content-Type: text/plain
Content-Length: 24

Method not implemented!
```

Η μέθοδος GET [RFC2616/§9.3] που θα υλοποιήσετε θα πρέπει να δοκιμάσει την ανάκτηση του αρχείου που της προσδιορίζει η γραμμή αίτησης μέσα από τον κατάλογο περιεχομένου του web server σας. Αν τα καταφέρει, θα πρέπει να στείλει ένα μήνυμα απάντησης με κωδικό κατάληξης της αίτησης το 200 και περιγραφή "OK" [RFC2616/§10.2.1]. Στο σώμα του μηνύματος πρέπει να βρίσκεται το περιεχόμενο του αρχείου που σας ζητήθηκε. Αν αποτύχει στην ανάκτηση του αρχείου, θα πρέπει να στείλει ένα μήνυμα απάντησης με κωδικό κατάληξης της αίτησης 404 και περιγραφή "Not Found" [RFC2616/§10.4.5].

Τη μέθοδο HEAD πρέπει να τη χειρίζεστε με εντελώς παρόμοιο τρόπο με αυτό για την GET, εκτός από το ότι στην απάντηση από το web server δε θα περιλαμβάνεται το περιεχόμενο του αρχείου που ζητήθηκε [RFC2616/§9.4].

Η εντολή DELETE πρέπει να τη χειρίζεστε με εντελώς παρόμοιο τρόπο με αυτό για την HEAD, με τη διαφορά ότι ο web server θα σβήνει το περιεχόμενο του αρχείου που ζητήθηκε και να επιστρέφει HTTP/1.1 200 OK ή HTTP/1.1 404 Not Found [RFC2616/§9.7].

Κατά την εξυπηρέτηση των αιτήσεων που έρχονται, ο web server σας θα πρέπει να αντιλαμβάνεται και να ερμηνεύει σωστά την επικεφαλίδα Connection της αίτησης, εφ' όσον υπάρχει, με τον εξής τρόπο: Αν έχει την τιμή close, μετά την εξυπηρέτηση της αίτησης, ο web server θα πρέπει να τερματίσει τη σύνδεση με τον πελάτη. Στην απάντησή του σε αυτήν την περίπτωση θα πρέπει να συμπεριλάβει τη γραμμή επικεφαλίδας "Connection: close". Διαφορετικά, αν δηλαδή η επικεφαλίδα στην αίτηση του πελάτη δεν έχει την τιμή close ή αν απουσιάζει, τότε ο web server θα

πρέπει να περιμένει για νέα μηνύματα αιτήσεων μετά από την εξυπηρέτηση της τρέχουσας αίτησης. Στην απάντησή του, σε αυτή την περίπτωση, θα πρέπει να συμπεριλάβει τη γραμμή επικεφαλίδας "Connection: keep-alive" [RFC2616/§14.10], [RFC2616/§8.1]. Οποιαδήποτε άλλη επικεφαλίδα της αίτησης μπορείτε να την αγνοείτε[†].

Στα μηνύματα απάντησης που στέλνει ο web server σας θα πρέπει να περιλαμβάνει οπωσδήποτε τις ακόλουθες επικεφαλίδες:

- Connection: Έχει τις τιμές close ή keep-alive, ανάλογα με το αν πρόκειται να κλείσει τη σύνδεση μετά από αυτή την απάντηση, ή αν θα περιμένει για νέες αιτήσεις [RFC2616/§14.10], [RFC2616/§8.1].
- Server: Εδώ σαν τιμή βάζετε το όνομα του web server σας. [RFC2616/§14.38].
- Content-Length: Έχει σαν τιμή το μέγεθος σε bytes του σώματος του μηνύματος. Πρακτικά δηλαδή είναι το μέγεθος του αρχείου που στέλνετε [RFC2616/§14.13].
- Content-Type: Έχει σαν τιμή τον τύπο του αρχείου που επιστρέφετε, κατά MIME (*Multipurpose Internet Mail Extensions*). Για τις ανάγκες της άσκησης, μπορείτε να διαλέγετε έναν από τους τύπους text/plain (.txt, .sed, .awk, .c, .h), text/html (.html, .htm), image/jpeg (.jpeg, .jpg), image/gif (.gif) και application/pdf (.pdf), ανάλογα με την κατάληξη του αρχείου που στέλνετε. Αν η κατάληξη δεν είναι κάποια απ' αυτές, μπορείτε να στείλετε τον τύπο application/octet-stream [RFC2616/§14.17].

Ο web server που θα υλοποιήσετε πρέπει να είναι σε θέση να εξυπηρετεί «ταυτόχρονα» πολλές αιτήσεις από πελάτες. Δεν πρέπει, δηλαδή, να τελειώσει πρώτα με την εξυπηρέτηση μιας αίτησης και μετά να δέχεται νέες. Για να το πετύχετε αυτό, θα πρέπει να εκμεταλλευτείτε τη δυνατότητα ύπαρξης πολλών *νημάτων* μέσα στη διεργασία του web server. Μια ιδέα θα ήταν, όταν παίρνει μια αίτηση από πελάτη, να δημιουργεί ένα νήμα για να την εξυπηρετήσει, ενώ το αρχικό νήμα να περιμένει νέες αιτήσεις. Η εξυπηρέτηση των αιτήσεων αυτών θα γίνεται από νέα νήματα που θα δημιουργεί το αρχικό. Φυσικά, όταν ένα νήμα τελειώνει την αποστολή του, θα πρέπει να τερματίζει. **Η προσέγγιση αυτή δεν είναι πολύ καλή**, γιατί δεν είναι ιδιαίτερα ελεγχόμενη η δημιουργία και καταστροφή νημάτων στην εφαρμογή, κάτι που μπορεί να αποβεί εξαιρετικά προβληματικό σε κάποιες περιπτώσεις.

Μια άλλη, καλύτερη, ιδέα είναι το αρχικό νήμα να δημιουργήσει ένα thread-pool, δηλαδή να δημιουργήσει εξ αρχής ένα σταθερό αριθμό νημάτων-εργατών (που το πλήθος τους να δίνεται) και όταν υπάρχει αίτηση για εξυπηρέτηση να την αναθέτει σε κάποιο από τα νήματα αυτά που δεν έχει δουλειά. Τα νήματα, αφού εξυπηρετήσουν ένα πελάτη, δεν τερματίζουν, αλλά μεταβαίνουν σε κατάσταση αναμονής. Φυσικά, αν δεν υπάρχει διαθέσιμο νήμα, το αρχικό θα πρέπει να περιμένει μέχρι να υπάρξει,

[†] Σημειώνεται ότι στην έκδοση 1.1 του πρωτοκόλλου HTTP, στις αιτήσεις των πελατών είναι υποχρεωτική η επικεφαλίδα HOST, στη μορφή Host: <host>:<port>, όπου <host> είναι το μηχάνημα του web server και <port> η θύρα στην οποία περιμένει αιτήσεις σύνδεσης, ή απλώς Host: <host>, όπου η θύρα είναι η προκαθορισμένη για το HTTP πρωτόκολλο, η 80 [RFC2616/§14.23]. Παρ' όλα αυτά, στην δική σας υλοποίηση πρέπει να αγνοήσετε αυτό τον περιορισμό γιατί αυτό θα κάνει ευκολότερη την αποσφαλμάτωση της εφαρμογής σας με telnet.

χωρίς να δέχεται νέες αιτήσεις. Συγκεκριμένα εάν υπάρξουν περισσότερες αιτήσεις από το μέγιστο αριθμό νημάτων στο pool τότε το σύστημα απορρίπτει την αίτηση κλείνοντας το socket (χωρίς να επιστρέφει οποιανδήποτε απάντηση στον πελάτη). Άλλες παραμέτρους που χρειάζεται να πάρει ο web server σας, εκτός από το πλήθος των νημάτων, είναι ο αριθμός θύρας στον οποίο θα αναμένει αιτήσεις, ο κατάλογος-ρίζα του ιεραρχικού συστήματος αρχείων που «σερβίρει» και άλλες παραμέτρους που τυχόν χρησιμοποιήσετε. Οι παράμετροι μπορεί είτε να δίδονται ως ορίσματα στο πρόγραμμά σας ή καλύτερα να βρίσκονται σε κάποιο αρχείο config.txt, το οποίο θα έχει τη δομή

```
# Webserver Configuration File

# The Number of Threads in the Threadpool
THREADS=40

# The Port number of the web server
PORT=30000
…
```

Για να πειραματίζεστε με το πρόγραμμα που θα γράψετε, κατά τη φάση της ανάπτυξής του, μπορείτε να του στέλνετε αιτήσεις, είτε χρησιμοποιώντας ένα από τους υπάρχοντες φυλλομετρητές, είτε χειρονακτικά μέσω της εντολής telnet ("man telnet" για να δείτε πώς).

Για να δοκιμάσετε τον web server σας μπορείτε να δημιουργήσετε ένα δικό σας κατάλογο και να τοποθετήσετε εκεί τα αρχεία της αρεσκείας σας ή να αξιοποιήσετε ένα έτοιμο ιστοχώρο (δείτε `as4-supplementary.zip` στη σελίδα το μαθήματος)

III. Παράδειγμα Εκτέλεσης

Ακολουθούν παραδείγματα αιτήσεων από ένα πελάτη (αριστερή στήλη) και οι αντίστοιχες απαντήσεις από ένα web server (δεξιά στήλη).

```
GET /sample.html HTTP/1.1
User-Agent: My_web_browser
Host: b103ws1.in.cs.ucy.ac.cy:30000
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Server: My_test_server
Content-Length: 211
Connection: keep-alive
Content-Type: text/html

<html>
<head>
    <title>It worked!!!</title>
</head>
<body>
    <h1>Yes, It worked!!!</h1>
    Click at the image to see a
     sample text!<br>
     <a href="sample.txt">
         <img src="sample.gif">
     </a>
</body>
</html>
```

```
GET /sample.txt HTTP/1.1
User-Agent: My_web_browser
Host: b103ws1.in.cs.ucy.ac.cy:30000
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Server: My_test_server
Content-Length: 24
Connection: keep-alive
Content-Type: text/plain

Yes, this works also!!!
```

```
GET /dir/non-existent.gif HTTP/1.1
User-Agent: My_web_browser
Host: b103ws1.in.cs.ucy.ac.cy:30000
Connection: keep-alive
```

```
HTTP/1.1 404 Not Found
Server: My_test_server
Content-Length: 20
Connection: keep-alive
Content-Type: text/plain

Document not found!
```

```
OPTIONS * HTTP/1.1
User-Agent: My_web_browser
Host: b103ws1.in.cs.ucy.ac.cy:30000
Connection: close
```

```
HTTP/1.1 501 Not Implemented
Server: My_test_server
Content-Length: 24
Connection: close
Content-Type: text/plain

Method not implemented!
```

IV. Ανάπτυξη Λογισμικού

Η άσκηση αυτή θα υλοποιηθεί σε ομάδες όπως θα ανακοινωθούν στο Moodle, τα οποία αναμένεται να συμβάλουν ισομερώς σε χρόνο και ουσιαστική δουλειά.

Συντήρηση Εκδόσεων με το SVN

Για να γίνει καλύτερος συντονισμός μεταξύ των μελών της ομάδας σας αλλά και για να μπορέσουμε να παρακολουθήσουμε τη πρόοδο ανάπτυξης του λογισμικού σας (δηλ., πως ακριβώς έχει συμβάλει κάποιο άτομο στη διεκπεραίωση της εργασίας), **πρέπει** να γίνει χρήση του Συστήματος Έλεγχου Εκδόσεων ((Re)Version Control System) **Subversion (ή SVN)**. Παρακαλώ ανατρέξετε στο ΕΠΛ132 για περισσότερες πληροφορίες σχετικά με το SVN.

V. Αξιολόγηση
A) Τι πρέπει να παραδώσετε;

- **Στο Moodle:** Ενα αρχείο **web.tar.gz** το οποίο θα περιέχει:
  1. Τον πηγαίο κώδικα μαζί με το σχετικό Makefile,
  2. Ένα README.txt αρχείο οποίο θα δίδει οδηγίες χρήσης του συστήματός σας (περίπου 1 σελίδα) και
  3. Architecture (ή ODT, DOC ή PDF), το οποίο θα περιγράφει την αρχιτεκτονική του συστήματος, τις βασικές επιλογές στο σχεδιασμό αυτής της αρχιτεκτονικής, περιγραφή της επιπλέον λειτουργίας που αποφασίσατε να υλοποιήσετε, διάφορες δυσκολίες που αντιμετωπίσατε (~1-2 σελίδες).
- **Στο Εργαστήριο: Να παραδοθούν εκτυπωμένα τα 1-3.**

B) Κριτήρια Αξιολόγησης.

1. **30% - Δομή Συστήματος:** Το σύστημα πρέπει να χρησιμοποιεί τεχνικές δομημένου προγραμματισμού με τη χρήση συναρτήσεων, αρχείων επικεφαλίδας (.h), πολλαπλών αρχείων για καλύτερη δομή του πηγαίου κώδικα, Makefile, αρχείων ελέγχου (test files) τα οποία θα ελέγχουν την ορθότητα των συστατικών (modules) του συστήματος σας ανεξάρτητα από την υπόλοιπη λειτουργία του συστήματος, διαχείριση λαθών συστήματος με την perror, έλεγχος ταυτοχρονίας νημάτων με χρήση σηματοφόρων, κτλ.

2. **70% - Ορθότητα Λειτουργίας:** Το σύστημα θα πρέπει να διεκπεραιώνει ορθά τις λειτουργίες του συστήματος όπως αυτές περιγράφονται σε αυτή την εκφώνηση και το RFC2616. Θέλουμε να γίνει ξεκάθαρο ότι η εκφώνηση της άσκησης δεν σας δεσμεύει για τις δυνατότητες που θα έχει ο εξυπηρετητής που θα υλοποιήσετε. **Η εκφώνηση απλά θέτει ένα ελάχιστο όριο δυνατοτήτων που θα πρέπει να υλοποιήσετε.** Αυτό είναι σκόπιμο για να σας αφήσει αρκετή ελευθερία στη λήψη πρωτοβουλιών και στην εκδήλωση δημιουργικότητας από την πλευρά σας. Μέσα από αυτή την άσκηση θέλουμε να σας δοθεί η δυνατότητα να επεξεργαστείτε από μόνοι σας ένα τεχνικό έγγραφο (RFC2616) καθώς επίσης να ανακαλύψετε νέες συναρτήσεις πέρα από αυτές που διδαχθήκατε ήδη στις διαλέξεις.

Καλή Επιτυχία !