ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
Τμήμα Πληροφορικής

ΕΠΛ 421 - Προγραμματισμός Συστημάτων

ΑΣΚΗΣΗ 4 – Ανάπτυξη Πολυνηματικού Κόμβου myOpenHAB

Διδάσκων: Δημήτρης Ζεϊναλιπούρ

Υπεύθυνος Εργαστηρίου: Παύλος Αντωνίου

Ημερομηνία Ανάθεσης: Παρασκευή 15/11/19
Ημερομηνία Παράδοσης: Παρασκευή 29/11/19 και ώρα 13:00 (14 μέρες)

(η λύση να υποβληθεί σε zip μέσω του Moodle)
http://www.cs.ucy.ac.cy/courses/EPL421

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.

Το θέμα της άσκησης είναι η υλοποίηση ενός προγράμματος που να υποστηρίζει ένα βασικό υποσύνολο των λειτουργιών του openHAB. Οι λειτουργίες του προγράμματος σας και το αναμενόμενο αποτέλεσμα περιγράφονται αναλυτικότερα στην συνέχεια.

II. Περιγραφή

Η λειτουργία της πλατφόρμας openHAB περιγράφηκε ενδελεχώς στην άσκηση 2. Στην παρούσα άσκηση θα αναπτύξετε ένα υποσύνολο των λειτουργιών της εν λόγω πλατφόρμας.
Ο κόμβος που θα αναπτύξετε θα είναι επί της ουσίας ένας «πολυνηματικός» εξυπηρετητής 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.[1]

II-A. Εισαγωγή στον Εξυπηρετητή Ιστού

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 (πελάτης) είναι να συνδεθεί μέσω υποδοχών ροής (sockets) στην προκαθορισμένη θύρα (80) για την HTTP υπηρεσία του μηχανήματος <host> (εξυπηρετητής) και να υποβάλει ένα αίτημα,

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

σε αυστηρά καθορισμένη μορφή που θα περιγραφεί στη συνέχεια, για να του αποσταλούν από το web server το αρχείο που βρίσκεται στη θέση /<path> μαζί με κάποιες συνοδευτικές μετα-πληροφορίες. Το /<path> είναι σχετικό σε σχέση με τον κατάλογο-ρίζα του ιεραρχικού συστήματος αρχείων που «σερβίρει» ο εξυπηρετητής. Στον εξυπηρετητή HTTP Apache, από προ-επιλογή, ο κατάλογος-ρίζα είναι ο /var/www/. Όταν ο φυλλομετρητής λάβει το αρχείο, το παρουσιάζει 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>.

Στην παρούσα άσκηση, ο δικός σας εξυπηρετητής δεν στέλνει αρχεία που βρίσκονται σε κάποιο path αλλά αποστέλνει δεδομένα σε μορφή json τα οποία θα είναι αποθηκευμένα σε κάποια δομή (στη RAM). Περισσότερες πληροφορίες θα δείτε στην ενότητα ΙΙ-Β.

Ο πελάτης και ο εξυπηρετητής HTTP επικοινωνούν μεταξύ τους ανταλλάζοντας 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] όπως για 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 π.χ. `GET`, `HEAD`, `POST`, `DELETE`, `PUT` κτλ) 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 εξυπηρετητής 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 εξυπηρετητής 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[2].

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

[2] 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 εξυπηρετητή 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.

- `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 εξυπηρετητή 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 εξυπηρετητής 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 από κάποιο configuration file) 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, 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 εξυπηρετητής 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).

II-B. Υποστηριζόμενες Λειτουργίες τύπου openHAB

Ο πολυνηματικός εξυπηρετητής αυτός θα υποστηρίζει **1 thing**, το openweathermap μαζί με το binding του καθώς και τα πιο κάτω **18 items**:

- Station Id (station_id)
- Observation Time (current_time)
- Outdoor Temperature (current_temp)
- Barometric Pressure (current_pressure)
- Atmospheric Humidity (current_humidity)
- Wind Speed (current_speed)
- Cloudiness (current_cloudiness)
- Rain (current_rain)
- Sunrise (current_sunrise)
- Sunset (current_sunset)

Current Weather

- 3 Hours Forecast Time (forecast3_time)
- 3 Hours Forecasted Temperature (forecast3_temp)
- 6 Hours Forecast Time (forecast6_time)
- 6 Hours Forecasted Temperature (forecast6_temp)
- 9 Hours Forecast Time (forecast9_time)
- 9 Hours Forecasted Temperature (forecast9_temp)
- 12 Hours Forecast Time (forecast12_time)
- 12 Hours Forecasted Temperature (forecast12_temp)

Forecast

Στην παρένθεση φαίνεται το id κάθε item το οποίο θα χρησιμοποιηθεί για να ανακτήσουμε πληροφορίες για το κάθε item.

Τα πιο πάνω items απορρέουν από τα πιο κάτω APIs:

- Current weather API:
  - Περιγραφή: https://openweathermap.org/current
  - URL:
    http://api.openweathermap.org/data/2.5/weather?q=LOCATION&units=metric&APPID=API_KEY
- 5 days/3 hour forecast API
  - Περιγραφή: https://openweathermap.org/forecast5
  - URL:
    http://api.openweathermap.org/data/2.5/forecast?q=LOCATION&units=metric&APPID=API_KEY

Οι τιμές του LOCATION και API_KEY θα βρίσκονται στο config.txt π.χ.

```
# API KEY for openweathermap
API_KEY=b6907d289e10d714a6e88b30761fae22

# Location for openweathermap
LOCATION=Nicosia,cy
```

του κόμβου το οποίο θα περιγραφεί πιο κάτω. Όταν ο κόμβος σας ξεκινά, θα αρχικοποιεί ένα νήμα το οποίο θα ενώνεται στη πλατφόρμα Open Weather Map και θα φέρνει δεδομένα από τα πιο πάνω APIs και θα τα αποθηκεύει σε κάποια δομή. Για παράδειγμα, για να λάβουμε τα τρέχοντα καιρικά δεδομένα για τη Λευκωσία θα πρέπει να στείλουμε μια HTTP GET αίτηση όπως φαίνεται πιο κάτω:

```
GET /data/2.5/weather?q=Nicosia,cy&APPID=API_KEY HTTP/1.1
Host: api.openweathermap.org
User-Agent: myOpenHAB
Accept: application/json
Connection: close
```

Η πιο πάνω αίτηση (θυμηθείτε από την άσκηση 2, ότι η αίτηση τερματίζεται με μια κενή γραμμή) θα σταλεί στη θύρα 80 του εξυπηρετητή ιστού της πλατφόρμας OpenWeatherMap ο οποίος είναι ο `api.openweathermap.org`. Για την επεξεργασία του αρχείου json που θα ληφθεί μπορείτε να χρησιμοποιήσετε κάποιες βιβλιοθήκες επεξεργασίας json strings όπως είναι για παράδειγμα η βιβλιοθήκη Jansson[3]. Η διαδικασία αυτή θα επαναλαμβάνεται περιοδικά κάθε κάποιο προκαθορισμένο χρονικό διάστημα το οποίο θα ορίζεται επίσης στο config.txt:

```
# time duration between successive connections to openweathermap
# (in seconds)
DURATION=600
```

Οι REST αιτήσεις[4] που θα δέχεται ο εξυπηρετητής σας θα είναι οι ίδιες με αυτές που δέχεται το openHAB Εγκαταστήστε το REST API όπως φαίνεται εδώ για να δείτε τη μορφή των json strings που ανταλλάζονται.

GET /items	Επιστρέφει μήνυμα 200 OK που στο body περιέχει μια λίστα με όλα τα διαθέσιμα items. Η μορφή του json που στέλνει ο server φαίνεται εδώ - http://10.16.30.25:8080/doc/index.html#/items/getitems άρα μόνο με VPN (και ένα πλήρες παράδειγμα της εντολής φαίνεται εδώ. - http://10.16.30.25:8080/rest/items . Αρχικά ο εξυπηρετητής δεν θα έχει κανένα διαθέσιμο item και το endpoint αυτό θα επιστρέφει άδεια λίστα []. Για το όνομα (name) και το σύνδεσμο (link) προς τις πληροφορίες του κάθε item να χρησιμοποιήσετε το id που αναφέραμε πιο πάνω (στην παρένθεση δίπλα από κάθε item). Για παράδειγμα, το name για την τρέχουσα εξωτερική θερμοκρασία θα είναι current_temp και το link για τις πληροφορίες του θα είναι http://SERVER-IP-ADDRESS/items/current_temp.
PUT /items	Προσθέτει μια λίστα με items (έτσι ώστε να γίνουν διαθέσιμα) ή ανανεώνει παραμέτρους ήδη διαθέσιμων items. Για κάθε item που γίνεται διαθέσιμο θα μπορεί ο χρήστης να ανακτήσει πληροφορίες. Επιστρέφει 200 OK.

3 Μπορείτε να κατεβάσετε και να εγκαταστήσετε τη βιβλιοθήκη αυτή σε υπολογιστή στον οποίο έχει δικαιώματα διαχειριστή όπως για παράδειγμα οι εικονικές μηχανές που σας δώσαμε και να δείτε στη συνέχεια πως μπορεί να χρησιμοποιηθεί από το documentation. Για να ξεκινήσετε, μπορείτε να συμπεριλάβετε τη γραμμή αυτή στην αρχή του προγράμματός σας `#include <jansson.h>` και να μεταγλωττίσετε βάζοντας στη γραμμή εντολών (του gcc) το `-ljansson`

4 The REST API documentation is available on your openHAB installation as a custom user interface. You can add it by opening the PaperUI interace, clicking "Add-ons", selecting "Misc". You will see an addon called "REST Documentation" that can be easily installed. On the welcome screen of openHAB, you will now see a new interface called "REST API" where you can easily see the documentation of the REST API and test it! All available commands will be displayed in the REST API documentation as a response to the button "Try me out" (may require some user input for mandatory fields).

GET /items/{itemname}	Αν το item (που προσδιορίζεται με το itemname) υπάρχει, ο server επιστρέφει μήνυμα 200 OK που στο body περιέχει τις πληροφορίες του item. Δείτε παράδειγμα εδώ. http://10.16.30.25:8080/rest/items/WeatherAndForecast_Current_OutdoorTemperature
PUT /items/{itemname}	Αν το item (που προσδιορίζεται με το itemname) υπάρχει, ο server το ενημερώνει με τις νέες πληροφορίες και επιστρέφει μήνυμα 200 OK που στο body δεν έχει κάτι. Αν το item δεν υπάρχει, ο server επιστρέφει μήνυμα 404 Not Found.
DELETE /items/{itemname}	Αν το item (που προσδιορίζεται με το itemname) υπάρχει, ο server το διαγράφει και επιστρέφει μήνυμα 200 OK που στο body δεν έχει κάτι. Αν το item δεν υπάρχει, ο server επιστρέφει μήνυμα 404 Not Found.

III. Ανάπτυξη Λογισμικού

Ομάδες

Η άσκηση αυτή θα υλοποιηθεί σε ομάδες των 3 ατόμων (βάσει των ομάδων της παρουσίασης) τα οποία αναμένεται να συμβάλουν ισομερώς σε χρόνο και ουσιαστική δουλειά.

Ανάπτυξη Κώδικα με IDE και Σύστημα Ελέγχου Εκδόσεων (Versioning System)

Συνίσταται όπως αναπτύξετε τον κώδικα σας με χρήση κάποιου IDE εργαλείου π.χ. **Eclipse IDE for C/C++ Developers (http://www.eclipse.org/downloads/) ή Clion IDE από Jetbrains.**

Συνίσταται επίσης η χρήση Συστήματος Έλεγχου Εκδόσεων ((Re)Version Control System) **Subversion (SVN) ή Git** και αυτό **θα ληφθεί υπόψιν θετικά κατά την αξιολόγηση**. Η χρήση τέτοιου εργαλείου συνίσταται κυρίως για να γίνει καλύτερος συντονισμός μεταξύ των μελών της ομάδας σας (ο κάθε φοιτητής θα ανεβάζει το δικό του κομμάτι κώδικα στο σύστημα) αλλά και για να μπορείτε να παρακολουθείτε τη πρόοδο ανάπτυξης του λογισμικού σας. Τα συστήματα ελέγχου εκδόσεων χρησιμοποιούνται για τη συντήρηση ιστορικών εκδόσεων αρχείων όπως για παράδειγμα πηγαίου κώδικα, ιστοσελίδων αλλά και άλλων εγγράφων και τέτοια συστήματα θεωρούνται απαραίτητα για την ανάπτυξη λογισμικού σε ομάδες (για ένα πλήρη κατάλογο τέτοιων συστημάτων: http://en.wikipedia.org/wiki/List_of_revision_control_software). Όποια ομάδα ενδιαφέρεται να χρησιμοποιήσει το SVN να επικοινωνήσει στο paul.antoniou@cs.ucy.ac.cy για να δημιουργήσουμε repository πάνω στον SVN server του τμήματος πληροφορικής. Όποια ομάδα επιθυμεί να χρησιμοποιήσει το git μπορεί να δημιουργήσει -δωρεάν- repository πάνω στο https://github.com/.

SVN

Σχετικά με το SVN, σημειώστε ότι το **Eclipse** διαθέτει το πρόσθετο εργαλείο (plugin) με όνομα **Subclipse,** το οποίο σας επιτρέπει να διαχειριστείτε τον SVN κατάλογο της ομάδας σας μέσα από το ίδιο το Eclipse. Για να εγκαταστήσετε το **Subclipse** στο **Eclipse** ακολουθείστε τον πιο κάτω σύνδεσμο: https://github.com/subclipse/subclipse/wiki.
Ένας άλλος βολικός τρόπος διαχείρισης του SVN καταλόγου σας είναι μέσω κάποιου εξειδικευμένου προγράμματος ο οποίος σας επιτρέπει να επιτύχετε αυτή την αλληλεπίδραση από τον Window Manager του λειτουργικού σας συστήματος. Παραδείγματα τέτοιων εργαλείων είναι τα ακόλουθα:

- **Linux:** kdesvn (KDE-integrated SVN client) (https://kdesvn.alwins-world.de/)
- **Windows:** TortoiseSVN (https://tortoisesvn.net/)
- **Cross-Platform (QT-based):** eSVN (http://sourceforge.net/projects/esvn)
- **MacOSX:** SCPlugin (http://scplugin.tigris.org/)

Ένας τελευταίος τρόπος διαχείρισης του SVN καταλόγου της ομάδας σας είναι μέσω της εντολής κελύφους svn (man svn για περισσότερες πληροφορίες).

Git
Σχετικά με το Git, το Eclipse διαθέτει επίσης Plugin με το όνομα EGit. Για να εγκαταστήσετε το **Egit** στο **Eclipse** ακολουθείστε τον πιο κάτω σύνδεσμο: http://www.eclipse.org/egit/.
Υπάρχει επίσης και Git client για χρήση του git μέσω του terminal.

IV. Αξιολόγηση

A) Τι πρέπει να παραδώσετε στο moodle;

- Ενα αρχείο **web.tar.gz** το οποίο θα περιέχει:
  1. Τον πηγαίο κώδικα (όλα τα .c/.h αρχεία) 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 (~2-3 σελίδες).

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. Θα ληφθεί ιδιαιτέρως υπόψη η δυνατότητα του εξυπηρετητή σας να επικοινωνεί σωστά με εμπορικούς φυλλομετρητές όπως για παράδειγμα Google Chrome, Mozilla Firefox, Safari και Microsoft Edge. 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 !