Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών
Διδάσκων: Ι. Γαροφαλάκης
Εργαστηριακή άσκηση 2014

<u>Στοιχεία μέλους ομάδας (αλφαβητικά)</u>

Ζαχαρόπουλος Ελευθέριος (Α.Μ.: 5512, e-mail: zacharopo@ceid.upatras.gr)

Ζαχαροπούλου Χρυσή (Α.Μ.: 5513, e-mail: zacharop@ceid.upatras.gr)

Μενύχτας Ευθύμιος (Α.Μ.: 5585, e-mail: menychta@ceid.upatras.gr)

Παππάς Αλέξανδρος (Α.Μ.: 5625, e-mail: papp@ceid.upatras.gr)

<u>ΑΝΑΦΟΡΑ</u>

Σκοπός της εργαστηριακής άσκησης ήταν η δημιουργία μίας γλώσσας με γραμματική σε μορφή BNF η οποία είναι ένα υποσύνολο του πρωτοκόλλου **http (rfc2616)** και πιο συγκεκριμένα ένα **μήνυμα αίτησης (request message)** με συγκεκριμένους περιορισμούς. Τέλος, έπρεπε να υλοποιηθούν δύο από τα τμήματα ενός μεταγλωττιστή, ο συντακτικός και λεκτικός αναλυτής με τη χρήση των εργαλείων Bison και Flex, βασιζόμενα πάνω σε αυτή την γλώσσα.

<u>ΣΗΜΕΙΩΣΗ:</u> Η εργαστηριακή άσκηση έγινε σε Ubuntu 14.04 LTS, οπότε μετά την είσοδο των εντολών του makefile στο terminal των Ubuntu, αρκεί να εισαχθεί η εντολή **./http** για να τρέξει το αρχείο.

1. Παρακάτω δίνεται η περιγραφή της γλώσσας σε BNF. Σημειώνεται πως στα πρωτόκολλα http, το message body διαχωρίζεται από τα headers με **μία** κενή γραμμή. Επίσης, δεν γίνεται αποδεκτή η ύπαρξη κενής γραμμής μεταξύ των header, μέσα στο message body όμως δεν υπάρχει πρόβλημα να υπάρχουν κενές γραμμές.

Γραμματική BNF

<Request> ::= <Request-Line> <EOL> <Headers> <Footer>

<Request-Line> ::= <Method> " " <Request-URI> " " <HTTP-Version>

<Method> ::= "GET" | "HEAD" | "POST"

<Request-URI> ::= <String>

<HTTP-Version> ::= "HTTP/1.1" | "HTTP/1.0"

<Headers> ::= <Headers> <Header> <EOL> | <Header> <EOL>

<Header> ::= <General-Header> | <Request-Header> | <Entity-Header>

<General-Header> ::= <Connection> | <Date> | <Transfer-Encoding>

<Connection> ::= "Connection: " <String>

<Date> ::= "Date: " <Date-Format>

<Transfer-Encoding> ::= "Transfer-Encoding: " ("chunked" | "gzip" | "deflate")

<Request-Header> ::= <Accept-Charset> | <Referer> | <User-Agent>

<Accept-Charset> ::= "Accept-Charset: " <String>

<Referer> ::= "Referer: " <String>

<User-Agent> ::= "User-Agent: " <String>

<Entity-Header> ::= <Content-Length> | <Expires>

<Content-Length> ::= "Content-Length: " <Unsigned-Integer>

<Unsigned-Integer> ::= <Unsigned-Integer> <Digit> | <Digit>

<Expires> ::= "Expires: " <Date-Format>

<Date-Format> ::= <Digit><Digit> "/" <Digit><Digit> "/"
<Digit><Digit><Digit><Digit> " " <Digit><Digit> ":" <Digit><Digit>

<Digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<Footer> ::= <EOF> | <ENDL> <EOF> | <EOL> <Message-Body>

<Message-Body> ::= <Message-Body> <String> <ENDL> <EOF> | <Message-Body> <String> <ENDL> | <String> <ENDL> <EOF> | <String> <ENDL>

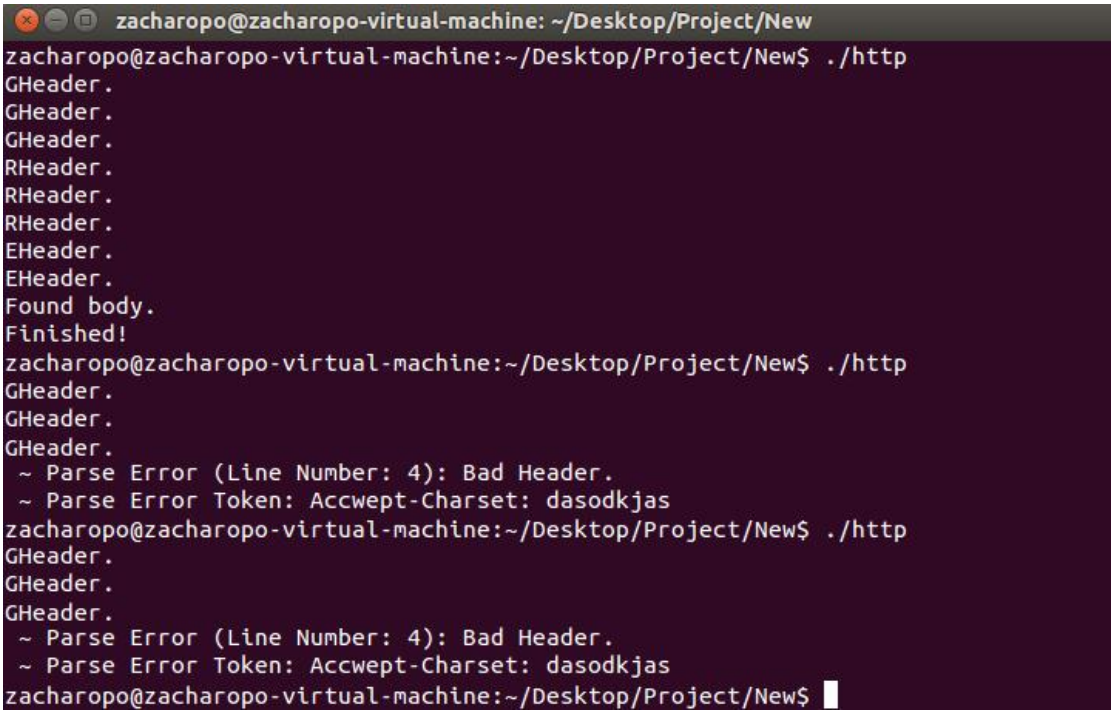
<ENDL> ::= <ENDL> <EOL> | <EOL>

<String> ::= <String> <Character> | <Character>

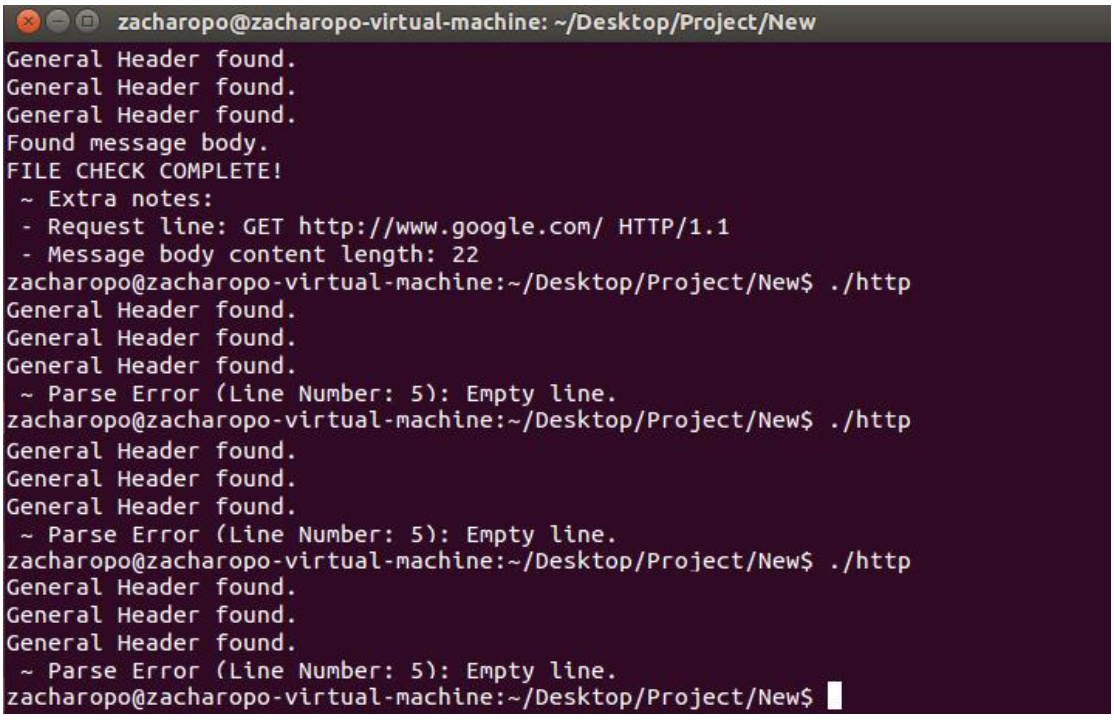
<Character> ::= (0 - 9) | (a - z) | (A - Z) | " " | "_" | "-" | "?" | "!" | "#" | "(" | ")" | "<" | ">" | "^" | "." | "," | ";" | ":" | "~" | "|" | "[" | "]" | "{" | "}" | "+" | "=" | "&" | "^" | "*" | "@" | "$" | "%" | "/" | "\" | "`" | "'" | """

Η παραπάνω γραμματική δίνει την μορφή την οποία θα έχει ένα σωστό μήνυμα αίτησης βάση του πρωτοκόλλου που έχει ζητηθεί. Στον κανόνα <Character>, ο συμβολισμός (0 - 9) (αριθμός από 0 έως 9) χρησιμοποιήθηκε για εξοικονόμηση χώρου, όπως και οι (a - z), (A - Z). Επιλέχθηκε να γραφεί γλώσσα με αριστερή αναδρομή, πράγμα το οποίο παρουσίασε ορισμένα προβλήματα κατά την υλοποίηση των http.y και http.l αρχείων, τα οποία όμως λύθηκαν, αν και ανάγκασαν ορισμένους κανόνες να είναι μεγαλύτεροι σε σύγκριση με γραμματική που κάνει χρήση δεξιάς αναδρομής (κυρίως λόγω της ανάγκης ελέγχου για το <EOF>).

2. Μετά την υλοποίηση των http.y και http.l για έλεγχο σωστά γραμμένου μηνύματος αίτησης, έπρεπε να γίνει μορφοποίηση του αρχείου http.y έτσι ώστε στους κανόνες να συμπεριλαμβάνονται και οι περιπτώσεις των πιθανών σφαλμάτων, και όταν αυτές συναντώνται, να γίνεται κλήση της yyerror(char* s); συνάρτησης του yylex. Ύστερα, έγινε μορφοποίηση της yyerror(char* s); έτσι ώστε σε περίπτωση σφάλματος, να εμφανίζεται σωστό διαγνωστικό μήνυμα καθώς και η γραμμή στην οποία υπήρξε το σφάλμα. Παρακάτω παρατίθενται εικόνες όπου το πρόγραμμα συνάντησε σφάλματα (οι εικόνες είναι από νωρίτερες εκδόσεις, οπότε ορισμένα μηνύματα μπορεί να διαφέρουν με αυτά της τελικής έκδοσης που έχει σταλεί).



Εικόνα όπου το πρόγραμμα ανιχνεύει σφάλμα σε header και τερματίζει απρόοπτα.



Εικόνα όπου το πρόγραμμα ανιχνεύει κενή γραμμή ανάμεσα σε header και τερματίζει απρόοπτα.

3. Ύστερα από υλοποίηση και του χειρισμού σφαλμάτων και των κατάλληλων μηνυμάτων τους, έγινε τροποποίση των αρχείων http.l και http.y έτσι ώστε όταν η μέθοδος είναι η POST, να είναι υποχρεωτική η παρουσία του message-body και της επικεφαλίδας Content-Length με την σωστή τιμή που πρέπει να περιέχει. Αυτό υλοποιήθηκε ως εξής:

- Μέσα στο http.l αρχείο έγινε δημιουργία των εξής μεταβλητών:

```
char *request_line, *method_check = (char*) malloc(4), *con_length = (char*) malloc(64);

int content_header_check = 0;
```

- Κατά την ικανοποίηση του κανόνα για το REQUESTLINE, γίνονται οι εξής πράξεις εντός του http.l:

```
request_line = strdup(yytext); strncpy(method_check, request_line, 4);  return REQUESTLINE;
```

Έτσι, η μεταβλητή method_check περιέχει τα 4 πρώτα γράμματα του request line, πράγμα που μπορεί να τροφοδοτηθεί στο http.y και να χρησιμοποιηθεί για έλεγχο της μεθόδου.

- Κατά την ικανοποίηση του κανόνα για το CONTENTLENGTH, γίνονται οι εξής πράξεις εντός του http.l:

```
strcpy(con_length, yytext+16); content_length = strtol(con_length, NULL, 10); content_header_check = 1; return CONTENTLENGTH;
```

Με αυτόν τον τρόπο, εφόσον το content header είναι της μορφής Content-Length:" "[0-9]+, αγνοούνται οι 16 πρώτοι χαρακτήρες του που αποτελούν γράμματα, και αποθηκεύονται οι υπόλοιποι που δηλώνουν το μέγεθος του message body αφού πρώτα μετατραπούν από αριθμοί-χαρακτήρες σε integer βάσης του 10. Επίσης, γίνεται αλλαγή της μεταβλητής content_header_check από 0 σε 1 η οποία δείχνει εάν βρέθηκε ή όχι content header. Ύστερα αυτά τροφοδοτούνται στο http.y για ελέγχους.

- Κατά την ικανοποίηση του κανόνα BODYSTRING, γίνονται οι εξής πράξεις εντός του http.l:

```
body_length += strlen(yytext); return BODYSTRING;
```

Έτσι, κάθε φορά που συναντάται αλφαριθμητικό, αυξάνεται η μεταβλητή body_length η οποία περιέχει το μέγεθος του message body. Αυτή η μεταβλητή τροφοδοτείται στο http.y για έλεγχο με το δοσμένο από τον χρήστη μέγεθος μέσω της μεταβλητής content_length. Στην περίπτωση, εντός του http.y, που κάποιο header ενεργοποιήσει το σφάλμα "Bad Header.", που σημαίνει πως το header διαβάστηκε ως BODYSTRING, υπάρχει η εντολή body_length = 0; για την αποφυγή λάθους αποτελέσματος κατά το διάβασμα του message body.

- Εντός του http.y, προστέθηκαν οι περιπτώσεις όπου η μέθοδος POST πρέπει να πετάξει μήνυμα σφάλματος, και με χρήση των μεταβλητών που αναφέρθηκαν παραπάνω, δημιουργήθηκαν έλεγχοι για την κάθε περίπτωση. Επίσης έγινε περαιτέρω επέκταση της yyerror(char* s) ώστε να καλύπτονται τα νέα είδη σφαλμάτων. Οι έλεγχοι αυτοί μπορούν να φανούν στο αρχείο http.y το οποίο βρίσκεται στο τέλος της αναφοράς. Παρακάτω παρατίθενται εικόνες όπου το πρόγραμμα συνάντησε σφάλματα μεθόδου POST (οι εικόνες είναι από νωρίτερες εκδόσεις, οπότε ορισμένα μηνύματα μπορεί να διαφέρουν με αυτά της τελικής έκδοσης που έχει σταλεί).

```
zacharopo@zacharopo-virtual-machine:~/Desktop/Project/New$ ./http
General Header found.
General Header found.
General Header found.
Request Header found.
Request Header found.
Request Header found.
Entity Header found.
Entity Header found.
Found message body while method was POST.
Bad content length.
 ~ ERROR during POST method. Content length given (20) does not match actual bod
y length.
zacharopo@zacharopo-virtual-machine:~/Desktop/Project/New$
```

Εικόνα όπου το πρόγραμμα αναγνωρίζει πως το δοσμένο εντός του αρχείου μέγεθος του message body κατά την μέθοδο POST είναι λάθος και τερματίζει απρόοπτα.

```
zacharopo@zacharopo-virtual-machine:~/Desktop/Project/New$ ./http
General Header found.
General Header found.
General Header found.
Request Header found.
Request Header found.
Request Header found.
Entity Header found.
Entity Header found.
No body found.
 ~ ERROR during POST method. No message body found.
zacharopo@zacharopo-virtual-machine:~/Desktop/Project/New$
```

Εικόνα όπου το πρόγραμμα αναγνωρίζει πως δεν υπάρχει message body κατά την μέθοδο POST και τερματίζει απρόοπτα.

```
zacharopo@zacharopo-virtual-machine:~/Desktop/Project/New$ ./http
General Header found.
General Header found.
General Header found.
Request Header found.
Request Header found.
Request Header found.
No Content Header found.
 ~ ERROR during POST method. No content header found.
zacharopo@zacharopo-virtual-machine:~/Desktop/Project/New$
```

Εικόνα όπου το πρόγραμμα αναγνωρίζει πως δεν υπάρχει content header κατά την μέθοδο POST και τερματίζει απρόοπτα.

4. Ύστερα από μορφοποίηση του κώδικα να ανιχνεύει λάθη σχετικά με την μορφή της εισόδου, αλλά και λάθη σχετικά με την μέθοδο POST, το μόνο που απομένει είναι η τροποποίηση του http.y έτσι ώστε όταν συναντάται λάθος να μην τερματίζει απρόοπτα το πρόγραμμα. Αυτό επιτυγχάνεται με την χρήση των εντολών yyerrok; και yyclearin; μετά από κάθε κλήση της yyerror(char* s); στις περιπτώσεις για σφάλμα του κάθε κανόνα. Η εντολή yyerrok; αποτελεί macro του Bison το οποίο υποδεικνύει πως η ανάκτηση από σφάλμα ήταν επιτυχής. Η εντολή yyclearin; αποτελεί macro του Bison το οποίο "πετάει" το look ahead symbol και επιτρέπει να πάει το πρόγραμμα παρακάτω. Παρακάτω παρατίθενται εικόνες όπου το πρόγραμμα συνάντησε σφάλματα αλλά συνέχισε την εκτέλεση του δίχως απρόοπτο τερματισμό (οι εικόνες είναι από νωρίτερες εκδόσεις, οπότε ορισμένα μηνύματα μπορεί να διαφέρουν με αυτά της τελικής έκδοσης που έχει σταλεί).

```
zacharopo@zacharopo-virtual-machine:~/Desktop/Project/New$ ./http
 ~ Parse Error (Line Number: 0): Bad Request line.
 ~ Parse Error Token: GEEET http://www.google.com/ HTTP/1.1
General Header found.
General Header found.
General Header found.
 ~ Parse Error (Line Number: 4): Bad Header.
 ~ Parse Error Token: Accwept-Charset: dasodkjas
Request Header found.
Request Header found.
Entity Header found.
Entity Header found.
No message body found.
Finished!
zacharopo@zacharopo-virtual-machine:~/Desktop/Project/New$
```

Σφάλματα σε request line και header.

```
 ~ Parse Error (Line Number: 2): Empty line.
 ~ Parse Error (Line Number: 3): Empty line.
 ~ Parse Error (Line Number: 4): Empty line.
 ~ Parse Error (Line Number: 5): Bad Header.
 ~ Parse Error Token: Connsection: 54mbps
 ~ Parse Error (Line Number: 6): Empty line.
 ~ Parse Error (Line Number: 7): Empty line.
General Header found.
 ~ Parse Error (Line Number: 9): Empty line.
 ~ Parse Error (Line Number: 10): Empty line.
General Header found.
 ~ Parse Error (Line Number: 12): Bad Header.
 ~ Parse Error Token: Refaerer: Hideo Kojima
 ~ Parse Error (Line Number: 13): Empty line.
 ~ Parse Error (Line Number: 14): Empty line.
Request Header found.
Request Header found.
 ~ Parse Error (Line Number: 17): Bad Header.
 ~ Parse Error Token: Cowntent-Length: 54321
 ~ Parse Error (Line Number: 18): Empty line.
 ~ Parse Error (Line Number: 19): Empty line.
Entity Header found.
Found message body.
Finished!
```

Σφάλματα κενών γραμμών και header.

```
zacharopo@zacharopo-virtual-machine:~/Desktop/Project/New$ ./http
General Header found.
General Header found.
General Header found.
Request Header found.
Request Header found.
Request Header found.
Entity Header found.
Entity Header found.
No body found.
 ~ ERROR during POST method. No message body found.
Bad content length.
 ~ ERROR during POST method. Content length given (15) does not match actual bod
y length.
Finished!
Request line: POST http://www.google.com/ HTTP/1.1
Content length: 0
zacharopo@zacharopo-virtual-machine:~/Desktop/Project/New$
```

Σφάλματα μεθόδου POST.

```
zacharopo@zacharopo-virtual-machine:~/Desktop/Project/New$ ./http
General Header found.
General Header found.
General Header found.
Request Header found.
Request Header found.
Request Header found.
 ~ Parse Error (Line Number: 8): Bad Header.
 ~ Parse Error Token: Cowntent-Length: 20
Entity Header found.
No body found.
 ~ ERROR during POST method. No message body found.
No Content Header found.
 ~ ERROR during POST method. No content header found.
Finished!
Request line: POST http://www.google.com/ HTTP/1.1
Content length: 0
zacharopo@zacharopo-virtual-machine:~/Desktop/Project/New$
```

Σφάλματα header και μεθόδου POST.

```
zacharopo@zacharopo-virtual-machine:~/Desktop/Project/New$ ./http
General Header found.
General Header found.
General Header found.
Request Header found.
Request Header found.
Request Header found.
 ~ Parse Error (Line Number: 8): Bad Header.
 ~ Parse Error Token: Cowntent-Length: 20
Entity Header found.
Found message body while method was POST.
No Content Header found.
 ~ ERROR during POST method. No content header found.
Finished!
Request line: POST http://www.google.com/ HTTP/1.1
Content length: 16
zacharopo@zacharopo-virtual-machine:~/Desktop/Project/New$
```

Σφάλματα header και μεθόδου POST.

Σφάλματα κενών γραμμών και header.

ΣΗΜΕΙΩΣΗ: Υπάρχει η περίπτωση λάθους όπου υπάρχει μια κενή γραμμή μεταξύ δύο header, και το header μετά την κενή γραμμή έχει συντακτικό λάθος, σε αυτή την περίπτωση, το πρόγραμμα αναγνωρίζει το συντακτικά λάθος header ως body, και εάν υπάρχουν σωστά header μετά από αυτό, το πρόγραμμα βρίσκει σφάλμα. Έχει γίνει handle αυτού του σφάλματος με εμφάνιση σωστού μηνύματος, ήταν αδύνατο όμως να μορφοποιηθεί ώστε το πρόγραμμα να συνεχίζει μετά από τέτοιο σφάλμα γιατί τότε θα υπήρχε conflict στον κανόνα <Request> ::= <Request-Line> <EOL> <Headers> <Footer> της γραμματικής.

Για την διευκόλυνση εξέτασης ορθής λειτουργίας του προγράμματος (λιγότερη πληκτρολόγηση), στην main του http.y έχει γραφεί κώδικας, ο οποίος εμφανίζει ένα μενού στον χρήστη και του δίνεται η επιλογή να επιλέξει τι είδους αρχείο επιθυμεί να ελεγχθεί, απλά πληκτρολογώντας τον αντίστοιχο αριθμό και πατώντας Enter.

Παρακάτω παρατίθεται ο κώδικας των αρχείων http.l και http.y και στο τέλος της αναφοράς υπάρχει η βιβλιογραφία.

Κώδικες λεκτικού και συντακτικού αναλυτή

http.l

```
%{
#include <cstdio>
#include <iostream>
#include <string>
#include <fstream>
#include "http.tab.h"
using namespace std;
#define YY_DECL extern "C" int yylex(void)
int line_number = 0;
int body_length = 0;
int content_length = 0;
int content_header_check = 0;
char *request_line, *method_check = (char*) malloc(4), *con_length = (char*) malloc(64);
%}

%%

(GET|HEAD|POST)" ".+" "HTTP\/1\.(1|0) { request_line = strdup(yytext);
strncpy(method_check, request_line, 4); return REQUESTLINE; }

Connection:" ".+ { return CONNECTION; }

Date:" "[0-9][0-9]\/[0-9][0-9]\/[0-9][0-9][0-9][0-9]" "[0-9][0-9]:[0-9][0-9] { return DATE; }

Transfer-Encoding:" "(gzip|chunked|deflate) { return TRANSFERENCODING; }

Accept-Charset:" ".+ { return ACCEPTCHARSET; }

Referer:" ".+ { return REFERER; }

User-Agent:" ".+ { return USERAGENT; }

Content-Length:" "[0-9]+ { strcpy(con_length, yytext+16); content_length =
strtol(con_length, NULL, 10);  content_header_check = 1; return CONTENTLENGTH; }

Expires:" "[0-9][0-9]\/[0-9][0-9]\/[0-9][0-9][0-9][0-9]" "[0-9][0-9]:[0-9][0-9]  { return
EXPIRES; }

\n { ++line_number; return ENDL; }

.+ { body_length += strlen(yytext); return BODYSTRING; }

<<EOF>> { return END_OF_FILE; }

%%
```

http.y

```
%{
#include <cstdio>
#include <iostream>
using namespace std;

#include <string.h>
#include "http.tab.h"
extern "C" int yylex(void);
extern "C++" int yyparse(void);
extern "C" FILE *yyin;
extern "C" char *yytext;
extern int line_number, body_length, content_length, content_header_check;
extern char *method_check, *request_line;
int req_error = 0, empty_lines = 0, unhandled_error = 1, i;
void yyerror(const char *s);

%}

%token ENDL
%token REQUESTLINE
%token CONNECTION DATE TRANSFERENCODING
%token ACCEPTCHARSET REFERER USERAGENT
%token CONTENTLENGTH EXPIRES
%token BODYSTRING
%token END_OF_FILE

%%

Request: Request_Line ENDL Headers Footer { cout << "FILE CHECK COMPLETE!\n"; };

Request_Line: REQUESTLINE | BODYSTRING { yyerror("Bad Request line."); yyerrok;
yyclearin; body_length = 0; req_error = 1; };

Headers: Headers Header ENDL | Header ENDL;

Header: ENDLS General_Header { yyerror("Empty line."); cout << "General Header
found.\n"; yyerrok; yyclearin; } | ENDLS Request_Header { yyerror("Empty line."); cout <<
"Request Header found.\n"; yyerrok; yyclearin; } | ENDLS Entity_Header { yyerror("Empty
line."); cout << "Entity Header found.\n"; yyerrok; yyclearin; } | General_Header { cout <<
"General Header found.\n"; } | Request_Header { cout << "Request Header found.\n"; } |
Entity_Header { cout << "Entity Header found.\n"; } | ENDLS error { yyerror("Empty line.");
yyerror("Bad Header."); yyerrok; yyclearin; body_length = 0; } | error { yyerror("Bad
Header."); yyerrok; yyclearin; body_length = 0;};

General_Header: CONNECTION | DATE | TRANSFERENCODING;

Request_Header: ACCEPTCHARSET | REFERER | USERAGENT;

Entity_Header: CONTENTLENGTH | EXPIRES;
```

```
Footer: END_OF_FILE { if ( strcmp(method_check, "POST") == 0 ) { yyerror("No body found."); if (content_header_check == 0) { yyerror("No Content Header found."); yyerrok; yyclearin; } else { if (body_length == content_length) cout << "Content length given matches.\n"; else { yyerror("Bad content length."); yyerrok; yyclearin; } } } else cout << "No message body found.\n"; unhandled_error = 0; } | ENDL Message_Body | ENDLS END_OF_FILE { if ( strcmp(method_check, "POST") == 0 ) { yyerror("No body found."); if (content_header_check == 0) { yyerror("No Content Header found."); yyerrok; yyclearin; } else { if (body_length == content_length) cout << "Content length given matches.\n"; else { yyerror("Bad content length."); yyerrok; yyclearin; } } } else cout << "No message body found.\n"; unhandled_error = 0; };

Message_Body: Message_Body BODYSTRING ENDLS END_OF_FILE { if ( strcmp(method_check, "POST") == 0 ) { cout << "Found message body while method was POST.\n"; if (content_header_check == 0) { yyerror("No Content Header found."); yyerrok; yyclearin; } else { if (body_length == content_length) cout << "Content length given matches.\n"; else { yyerror("Bad content length."); yyerrok; yyclearin; } } } else cout << "Found message body.\n"; unhandled_error = 0; } | BODYSTRING  ENDLS END_OF_FILE { if ( strcmp(method_check, "POST") == 0 ) { cout << "Found message body while method was POST.\n"; if (content_header_check == 0) { yyerror("No Content Header found."); yyerrok; yyclearin; } else { if (body_length == content_length) cout << "Content length given matches.\n"; else { yyerror("Bad content length."); yyerrok; yyclearin; } } } else cout << "Found message body.\n"; unhandled_error = 0; } | Message_Body BODYSTRING ENDLS | BODYSTRING ENDLS;

ENDLS: ENDLS ENDL { ++empty_lines; } | ENDL { ++empty_lines; }; /* Όποτε συναντάται \n, αυξάνει ο μετρητής empty_lines, ο οποίος χρησιμοποιείται στον αλγόριθμο της συνάρτησης yyerror("Empty line."); */

%%

int user_input;
FILE *myfile;

main()
{
        cout << endl << endl << "Welcome! For your convenience, 11 different input files were made" << endl << "to test the protocol's functionality." << endl << "Below you will find each of the input file's number with" << endl <<
"their corresponding test:" << endl << endl << "Enter 1 for a functioning GET request file." << endl <<  "Enter 2 for a functioning POST request file." << endl << "Enter 3 for a file with a bad request line." << endl << "Enter 4 for a GET request file with empty lines." << endl << "Enter 5 for a GET request file with bad headers." << endl << "Enter 6 for a GET request file with mixed errors." << endl <<  "Enter 7 for a POST request file with bad content length." << endl <<  "Enter 8 for a POST request file with no content header." << endl <<  "Enter 9 for a POST request file with no message body." << endl << "Enter 10 for a POST request file with mixed errors." << endl <<  "Enter 11 for a request file with an unhandled error." << endl << endl << "User input:" <<endl;

        cin >> user_input;
```

```
switch (user_input)
{
	case 1:
	myfile = fopen("get", "r");
	break;
	case 2:
	myfile = fopen("post", "r");
	break;
	case 3:
	myfile = fopen("badreq", "r");
	break;
	case 4:
	myfile = fopen("get1", "r");
	break;
	case 5:
	myfile = fopen("get2", "r");
	break;
	case 6:
	myfile = fopen("get3", "r");
	break;
	case 7:
	myfile = fopen("post1", "r");
	break;
	case 8:
	myfile = fopen("post2", "r");
	break;
	case 9:
	myfile = fopen("post3", "r");
	break;
	case 10:
	myfile = fopen("post4", "r");
	break;
	case 11:
	myfile = fopen("un_err", "r");
	break;
	default:
	cout << "No valid choice entered." << endl;
	break;
}

if (!myfile)
{
	cout << "File not found!\n";
	return -1;
}

yyin = myfile;
```

```
	do
	{
		yyparse();
		if (unhandled_error == 1)
		cout << " ~ UNHANDLED ERROR: Header(s) were found after body data.
Terminating." << endl;
		else
		{
		if (req_error == 0)
		cout << " ~ Extra notes:" << endl << " - Request line: " << request_line <<
endl << " - Message body content length: " << body_length << endl;
		else
		cout << " ~ Extra notes:" << endl << " - Message body content length: " <<
body_length << endl;
		}
	} while (!feof(yyin));
}

void yyerror(const char *s)
{
	if (s == "Empty line.")
	{
	for (i = empty_lines; i > 0; i--)
	cout << " ~ Parse Error (Line Number: " << (line_number - i) + 1 << "): " << s <<
endl;
	empty_lines = 0;
	}
      else if (s == "Bad Request line.")
	{
	cout << " ~ Parse Error: " << s << endl;
	cout << " ~ Parse Error Token: " << yytext << endl;
	}
	else if (s == "Bad Header.")
	{
	cout << " ~ Parse Error (Line Number: " << line_number + 1 << "): " << s << endl;
	cout << " ~ Parse Error Token: " << yytext << endl;
	}
	else if (s == "No Content Header found.")
	{
	cout << s << endl << " ~ ERROR during POST method. No content header found. "
<< endl;
	}
	else if (s == "Bad content length.")
	{
	cout << s << endl << " ~ ERROR during POST method. Content length given (" <<
content_length << ") does not match actual body length. " << endl;
      }
      else if (s == "No body found.")
	{
	cout << s << endl << " ~ ERROR during POST method. No message body found. "
<< endl;   }
}
```

Βιβλιογραφία

Online:

Σχετικά με Flex:
http://150.140.9.29/arxes_glwsswn/Flex_lecture_2011.pdf

Σχετικά με Bison:
http://150.140.9.29/arxes_glwsswn/Bison_lecture_2011.pdf

Συνεργασία Flex και Bison:
http://aquamentus.com/flex_bison.html