ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
Τμήμα Πληροφορικής

ΕΠΛ 421 - Προγραμματισμός Συστημάτων

ΑΣΚΗΣΗ 4 – Ανάπτυξη Παράλληλου TLS Εξυπηρετητή Μηνυμάτων Ηλεκτρονικού Ταχυδρομείου (POP3S)

Διδάσκων: Δημήτρης Ζεϊναλιπούρ
Υπεύθυνος Εργαστηρίου: Παύλος Αντωνίου

Ημερομηνία Ανάθεσης: Πέμπτη 14/11/2024
Ημερομηνία Παράδοσης: Τετάρτη 04/12/2024 και ώρα 13:00 (20 μέρες)
(να υποβληθεί ηλεκτρονικά στο Moodle)

I. Στόχος Άσκησης

Στόχος αυτής της εργασίας είναι η εξοικείωση με προχωρημένες τεχνικές προγραμματισμού διεργασιών, δια-διεργασιακής επικοινωνίας (μέσω υποδοχών TCP/IP, Ουρών Μηνυμάτων ή/και Κοινόχρηστης Μνήμης) 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* και της *δομής* του.

II. Το πρωτόκολλο POP3

Αντικείμενο της άσκησης είναι να αναπτύξετε ένα παράλληλο εξυπηρετητή μηνυμάτων ηλεκτρονικού ταχυδρομείου (mail server) για το πρωτόκολλο POP3, ο οποίος θα υποστηρίζει ένα βασικό υποσύνολο των εντολών του πρωτοκόλλου. Η λειτουργία του πρωτοκόλλου POP3 περιγράφεται στο τεχνικό άρθρο **Request For Comments 1939: http://tools.ietf.org/html/rfc1939.**

Όταν χρησιμοποιούμε ένα πελάτη μηνυμάτων ηλεκτρονικού ταχυδρομείου (mail client), όπως ο Mozilla Thunderbird, Microsoft Outlook, κλπ, για να κατεβάσουμε τα email μας, δίνουμε ένα URL της μορφής pop3://<mail host>/. Αυτό που κάνει τότε ο πελάτης μηνυμάτων ηλεκτρονικού ταχυδρομείου είναι να συνδεθεί μέσω υποδοχών ροής (socket) στην προκαθορισμένη θύρα (110) για την POP3 υπηρεσία του μηχανήματος <mail host> και να υποβάλει ένα αίτημα, σε αυστηρά καθορισμένη μορφή που θα περιγραφεί στη συνέχεια, για να του κατεβάσει ή διαγράψει από τον mail server κάποια μηνύματα ηλεκτρονικού ταχυδρομείου (emails).

III. Το Ασφαλές Πρωτόκολλο POP3S

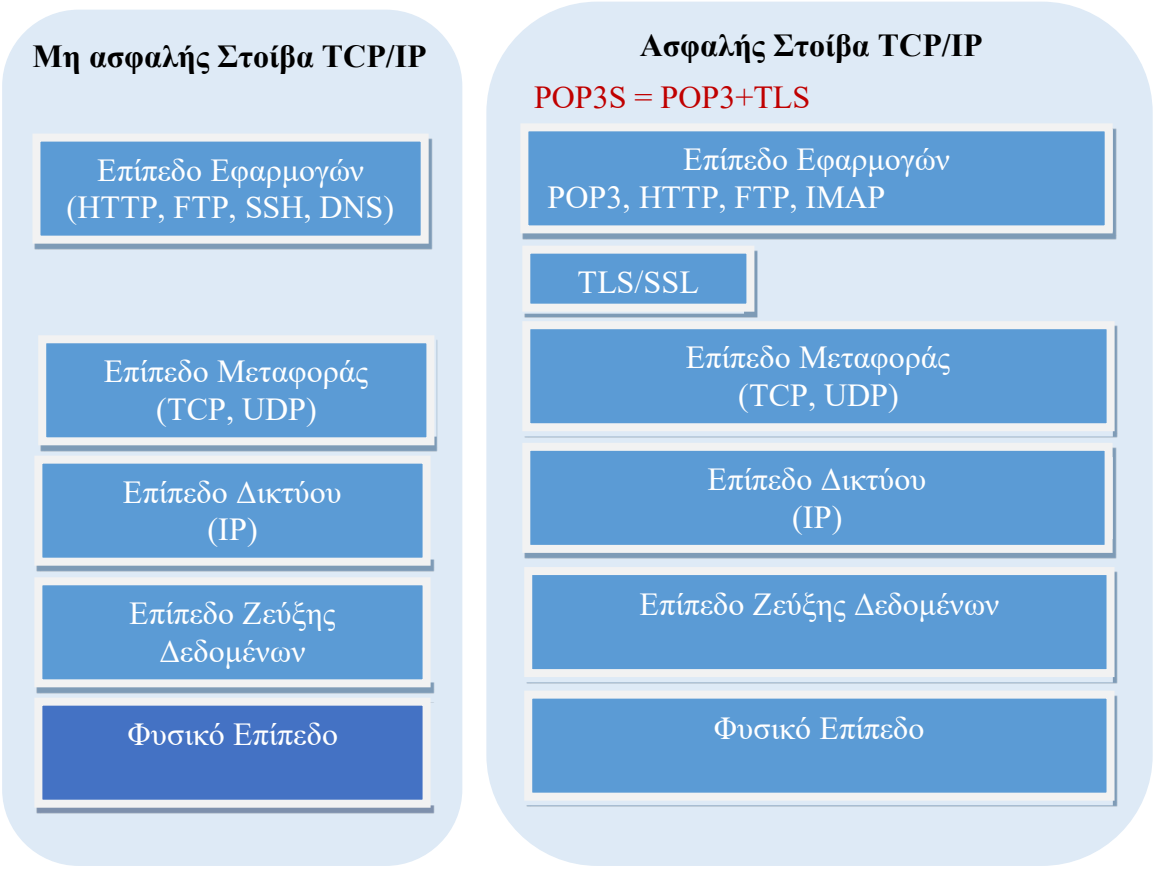
Το πρωτόκολλο POP3 δεν προδιαγράφει ασφαλή μετάδοση εντολών και δεδομένων και ως εκ τούτου η χρήση του τείνει να εξαλειφθεί. Υπάρχει διαθέσιμο το πρωτόκολλο που παρέχει ασφαλή επικοινωνία με τον mail server που ονομάζεται POP3S.

Το POP3S είναι βασικά το POP3 με την προσθήκη ενός (υπό)στρώματος ασφάλειας. Η ασφάλεια των συνδέσεων επιτυγχάνεται στη βάση των πρωτοκόλλων SSL/TLS (Secure Sockets Layer/Transport Layer Security). Οι προεκτάσεις TLS πάνω από το πρωτόκολλο POP3 (RFC1939) περιγράφεται στο RFC2595 http://www.rfc-editor.org/rfc/rfc2595.txt.

Το SSL (Secure Sockets Layer) και TLS (Transport Layer Security) είναι κρυπτογραφικά πρωτόκολλα σχεδιασμένα για να διασφαλίζουν ασφαλείς επικοινωνίες σε δίκτυα, κυρίως στο διαδίκτυο. Το TLS είναι ουσιαστικά ο διάδοχος του SSL, με βελτιώσεις στην ασφάλεια και στην αποδοτικότητα. Στα πλαίσια της άσκησης θα επικεντρωθούμε μόνο σε TLS και θα σας δοθεί και υποδειγματικός κώδικας. Ακολουθεί η περιγραφή της διαδικασίας πιστοποίησης ταυτότητας και ασφαλούς σύνδεσης για **POP3 μέσω SSL/TLS** (POP3S):

1. Αρχικοποίηση της SSL/TLS Σύνδεσης και Ανταλλαγή Πιστοποιητικών

- Ο πελάτης (π.χ., ένα πρόγραμμα email) συνδέεται με τον διακομιστή στην **θύρα 995** (προεπιλεγμένη θύρα για POP3S).
- Με την έναρξη της σύνδεσης, ο διακομιστής ξεκινά τη **διαδικασία SSL/TLS handshake** στέλνοντας το πιστοποιητικό SSL/TLS του στον πελάτη.
- Ο πελάτης επαληθεύει το πιστοποιητικό του εξυπηρετητή για να επιβεβαιώσει την ταυτότητά του. Αυτό γίνεται συνήθως με έλεγχο του πιστοποιητικού έναντι έμπιστων Αρχών Πιστοποίησης (CAs) που υπάρχουν στον πελάτη.
- Εάν το πιστοποιητικό είναι έγκυρο, αξιόπιστο και δεν έχει λήξει, η διαδικασία συνεχίζεται. Διαφορετικά, ο πελάτης μπορεί να εμφανίσει προειδοποίηση ασφαλείας ή να τερματίσει τη σύνδεση, ανάλογα με τη διαμόρφωσή του.



2. Ανταλλαγή Κλειδιού Συνεδρίας
- Ο πελάτης και ο εξυπηρετητής συμφωνούν σε έναν αλγόριθμο κρυπτογράφησης και δημιουργούν ένα **κλειδί συνεδρίας** (session key) μέσω μιας διαδικασίας όπως η **ανταλλαγή κλειδιών Diffie-Hellman** ή **RSA**. Αυτό το κλειδί συνεδρίας χρησιμοποιείται για την κρυπτογράφηση των δεδομένων κατά τη διάρκεια της συνεδρίας.
- Με τη δημιουργία του κλειδιού συνεδρίας, διαμορφώνεται ένα ασφαλές, κρυπτογραφημένο κανάλι επικοινωνίας που αποτρέπει τρίτους από το να αναχαιτίσουν ή να διαβάσουν την επικοινωνία.

3. Πιστοποίηση Πελάτη (Προαιρετικό)
- Σε ορισμένες περιπτώσεις, ο εξυπηρετητής μπορεί να ζητήσει από τον πελάτη πιστοποίηση με πιστοποιητικό του πελάτη, αν και αυτό είναι πιο σπάνιο στο email. Συνήθως, μόνο ο εξυπηρετητής πιστοποιεί την ταυτότητά του στον πελάτη.

4. Έναρξη Αυθεντικοποίησης POP3 (Όνομα Χρήστη και Κωδικός Πρόσβασης)
- Αφού ολοκληρωθεί το SSL/TLS handshake και έχει δημιουργηθεί ένα ασφαλές κανάλι, ο πελάτης στέλνει το **όνομα χρήστη** και τον **κωδικό πρόσβασης** στον εξυπηρετητή χρησιμοποιώντας εντολές POP3 (USER και PASS) μέσω της κρυπτογραφημένης σύνδεσης.
- Εφόσον η κρυπτογράφηση SSL/TLS είναι ενεργή, το όνομα χρήστη και ο κωδικός πρόσβασης προστατεύονται και οποιαδήποτε ευαίσθητη πληροφορία που αποστέλλεται μεταξύ του πελάτη και του εξυπηρετητή παραμένει ασφαλής από υποκλοπές.

5. Επαλήθευση Διαπιστευτηρίων (όνομα χρήστη, κωδικός) από τον Εξυπηρετητή
- Ο εξυπηρετητής επαληθεύει τα διαπιστευτήρια που έλαβε από τον πελάτη.
- Αν τα διαπιστευτήρια είναι σωστά, ο εξυπηρετητής απαντά με επιβεβαίωση, δίνοντας πρόσβαση στον πελάτη για ανάκτηση των email.
- Αν τα διαπιστευτήρια είναι εσφαλμένα, ο εξυπηρετητής στέλνει ένα μήνυμα σφάλματος και ο πελάτης μπορεί είτε να ζητήσει επανεισαγωγή των διαπιστευτηρίων είτε να τερματίσει τη σύνδεση.

6. Επικοινωνία POP3 μέσω SSL/TLS
- Αφού ο πελάτης έχει επαληθευτεί, ξεκινά η επικοινωνία POP3 μέσω του ασφαλούς καναλιού SSL/TLS. Εντολές όπως LIST, RETR, και DELE αποστέλλονται κρυπτογραφημένες, εξασφαλίζοντας την εμπιστευτικότητα και την ακεραιότητα των δεδομένων των email.

Εσείς δεν θα υλοποιήσετε την λειτουργία του πρωτοκόλλου TLS (βήματα 1-3 πιο πάνω). Υπάρχει διαθέσιμο το δωρεάν εργαλείο OpenSSL το οποίο υλοποιεί το πρωτόκολλο TLS και θα χρησιμοποιηθεί για να δημιουργήσει τα ασφαλή κανάλια (συνδέσεις ελέγχου και συνδέσεις δεδομένων) πάνω από τα οποία θα υλοποιήσετε το πρωτόκολλο POP3. Για περισσότερες λεπτομέρειες διαβάστε την επόμενη ενότητα.

IV. Περιγραφή Ενοτήτων Εργασίας

Αντικείμενο της άσκησης είναι να αναπτύξετε ένα ασφαλή «πολυδιεργασιακό» εξυπηρετητή ιστού, **POP3S server**. Πιο συγκεκριμένα θα υλοποιήσετε το πρωτόκολλο POP3 πάνω από το πρωτόκολλο TLS. Δεν θα υλοποιήσετε το TLS αλλά θα χρησιμοποιήσετε έτοιμες συναρτήσεις του εργαλείου OpenSSL που θα αναλάβουν τις διαδικασίες πιστοποίησης ταυτότητας και κρυπτογράφησης / αποκρυπτογράφησης. Ο εξυπηρετητής θα υποστηρίζει ένα βασικό υποσύνολο των εντολών του πρωτοκόλλου

POP3. Η λειτουργία του πρωτοκόλλου POP3 περιγράφεται στο τεχνικό άρθρο Request For Comments 1939. Προτού περιγράψουμε τις εντολές του POP3 που θα υλοποιήσετε, θα παρουσιάσουμε κάποια απλά παραδείγματα χρήσης του εργαλείου OpenSSL.

IV.Α) Δημιουργία Ασφαλούς Καναλιού Επικοινωνίας

Στις μηχανές του εργαστηρίου υπάρχει εγκατεστημένο το εργαλείο OpenSSL τόσο για χρήση από τη γραμμή εντολών (command line tool) όσο και σαν βιβλιοθήκη της γλώσσας C. Αν κάποιος χρησιμοποιεί δικό του UNIX/Linux λειτουργικό θα πρέπει να εγκαταστήσει τη βιβλιοθήκη αυτή. Για παράδειγμα σε λειτουργικό Ubuntu (Debian-based) η εντολή εγκατάστασης της βιβλιοθήκης είναι:

```
sudo apt update
sudo apt-get install openssl
sudo apt-get install libssl-dev
```

Ελέγξτε ότι ολοκληρώθηκε επιτυχώς η εγκατάσταση με την εντολή

```
openssl version
```

Το πιο κάτω πρόγραμμα (tls_server.c) δημιουργεί ένα απλό TLS server που ακούει για συνδέσεις στη θύρα 4433. Δείτε περισσότερα πιο κάτω.

```
/* ---------------------------------------------------------- *
 * file:        tls_server.c                                  *
 * purpose:     Example code for building a TLS server to     *
 *              accept connections at post 4433               *
 *                                                            *
 * gcc -o tls_server tls_server.c -lssl -lcrypto              *
 * ---------------------------------------------------------- */

#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <openssl/ssl.h>
#include <openssl/err.h>

int create_socket(int port)
{
    int s;
    struct sockaddr_in addr;

    /* set the type of connection to TCP/IP */
    addr.sin_family = AF_INET;
    /* set the server port number */
    addr.sin_port = htons(port);
    /* set our address to any interface */
    addr.sin_addr.s_addr = htonl(INADDR_ANY);

    s = socket(AF_INET, SOCK_STREAM, 0);
    if (s < 0) {
        perror("Unable to create socket");
        exit(EXIT_FAILURE);
    }

    /* bind serv information to s socket */
```

```
    if (bind(s, (struct sockaddr*)&addr, sizeof(addr)) < 0) {
        perror("Unable to bind");
        exit(EXIT_FAILURE);
    }

    /* start listening allowing a queue of up to 1 pending connection */
    if (listen(s, 1) < 0) {
        perror("Unable to listen");
        exit(EXIT_FAILURE);
    }

    return s;
}

void init_openssl()
{
    SSL_load_error_strings();
    OpenSSL_add_ssl_algorithms();
}
void cleanup_openssl()
{
    EVP_cleanup();
}

SSL_CTX *create_context()
{
    const SSL_METHOD *method;
    SSL_CTX *ctx;

    /* These are the general-purpose version-flexible SSL/TLS methods.
     * The actual protocol version used will be negotiated to the highest
     * version mutually supported by the client and the server. The
     * supported protocols are SSLv3, TLSv1, TLSv1.1, TLSv1.2 and TLSv1.3.
     */
    method = TLS_server_method(); // Βήματα 1-3 ενότητας III

    /* creates a new SSL_CTX object as framework to establish TLS/SSL or
     * DTLS enabled connections. It initializes the list of ciphers, the
     * session cache setting, the callbacks, the keys and certificates,
     * and the options to its default values
     */
    ctx = SSL_CTX_new(method);
    if (!ctx) {
        perror("Unable to create SSL context");
        ERR_print_errors_fp(stderr);
        exit(EXIT_FAILURE);
    }

    return ctx;
}

void configure_context(SSL_CTX *ctx)
{
    SSL_CTX_set_ecdh_auto(ctx, 1);

    /* Set the key and cert using dedicated pem files */
    if (SSL_CTX_use_certificate_file(ctx, "cert.pem", SSL_FILETYPE_PEM) <=
0) {
        ERR_print_errors_fp(stderr);
            exit(EXIT_FAILURE);
    }

    if (SSL_CTX_use_PrivateKey_file(ctx, "key.pem", SSL_FILETYPE_PEM) <= 0
) {
```

```
        ERR_print_errors_fp(stderr);
            exit(EXIT_FAILURE);
    }
}

int main(int argc, char **argv)
{
    int sock, bytes;
    char buf[1024];
    SSL_CTX *ctx;

    char enter[3] = { 0x0d, 0x0a, 0x00 };

        char output[1024];
        strcpy(output, "HTTP/1.1 200 OK");
        strcat(output, enter);
        strcat(output, "Content-Type: text/html");
        strcat(output, enter);
        strcat(output, "Content-Length: 47");
        strcat(output, enter);
        strcat(output, enter);
        strcat(output, "<html><body><h1>Hello World!</h1></body></html>");

    /* initialize OpenSSL */
    init_openssl();

    /* setting up algorithms needed by TLS */
    ctx = create_context();

    /* specify the certificate and private key to use */
    configure_context(ctx);

    sock = create_socket(4433);

    /* Handle connections */
    while(1) {
        printf("Accepting client....\n");
        struct sockaddr_in addr;
        uint len = sizeof(addr);
        SSL *ssl;
        //const char reply[] = "test\n";

            /* Server accepts a new connection on a socket.
         * Server extracts the first connection on the queue
         * of pending connections, create a new socket with the same
         * socket type protocol and address family as the specified
         * socket, and allocate a new file descriptor for that socket.
         */
        int client = accept(sock, (struct sockaddr*)&addr, &len);
        if (client < 0) {
            perror("Unable to accept");
            exit(EXIT_FAILURE);
        }

        /* creates a new SSL structure which is needed to hold the data
         * for a TLS/SSL connection
         */
        ssl = SSL_new(ctx);
        SSL_set_fd(ssl, client);

        /* wait for a TLS/SSL client to initiate a TLS/SSL handshake */
        if (SSL_accept(ssl) <= 0) {
            ERR_print_errors_fp(stderr);
        }
```

```
        /* if TLS/SSL handshake was successfully completed, a TLS/SSL
         * connection has been established
         */
        else {
            /* writes num bytes from the buffer reply into the
             * specified ssl connection
             */
            bytes = SSL_read(ssl, buf, sizeof(buf)); /* get request */
            if ( bytes > 0 )
            {
                buf[bytes] = 0;
                printf("Client msg: \"%s", buf);
                SSL_write(ssl, output, strlen(output));
            }
            else
                ERR_print_errors_fp(stderr);

        }

        /* sends a close notify over the socket */
        SSL_shutdown(ssl);
        /* free an allocated SSL structure */
        SSL_free(ssl);
        close(client);
    }

    close(sock);
    SSL_CTX_free(ctx);
    cleanup_openssl();
}
```

Όπως θα προσέξετε στη συνάρτηση configure_context απαιτείται η χρήση αρχείου που να περιέχει το ψηφιακό πιστοποιητικό (cert.pem) εξυπηρετητή και ενός άλλου αρχείου (key.pem) που να περιέχει το ιδιωτικό κλειδί του εξυπηρετητή. Μπορούμε να δημιουργήσουμε και τα 2 αυτά αρχεία μέσω του εργαλείου openssl από τη γραμμή εντολών μέσω της εντολής:

```
openssl req -newkey rsa:2048 -new -nodes -x509 -days 3650
-keyout key.pem -out cert.pem
```

η οποία δημιουργεί το ζεύγος δημόσιου και ιδιωτικού κλειδιού με βάση τον αλγόριθμο RSA, και αποθηκεύει το δημόσιο κλειδί μέσα στο πιστοποιητικό τύπου x509 (cert.pem) το οποίο ισχύει 3650 μέρες (10 χρόνια) προτού λήξει, και το ιδιωτικό κλειδί μέσα στο αρχείο key.pem. Και τα 2 αυτά κλειδιά έχουν μέγεθος 2048 bits. Κατά τη διάρκεια της δημιουργίας του κλειδιού, το εργαλείο ζητά να δοθούν κάποιες πληροφορίες για τον ιδιοκτήτη των κλειδιών (π.χ. χώρα, επαρχία, πόλη, όνομα οργανισμού, όνομα κόμβου/hostname, email). Αν θέλετε, μπορείτε να τα αφήσετε όλα κενά (πατώντας διαδοχικά enter για να προχωρήσει η διαδικασία).

Όταν ο πιο πάνω εξυπηρετητής δεχθεί μια αίτηση για σύνδεση και γίνει αποδεκτή (από τη συνάρτηση accept), αρχίζει η διαδικασία πιστοποίησης ταυτότητας μέσω ανταλλαγής πιστοποιητικών. Αν η πιστοποίηση ταυτότητας είναι επιτυχής, ο εξυπηρετητής στέλνει πίσω τη συμβολοσειρά test και τερματίζει τη σύνδεση.

Στη συνέχεια μπορείτε να μεταγλωττίσετε και να τρέξετε το πρόγραμμα με τις εντολές:

```
gcc -o tls_server tls_server.c -lssl –lcrypto ./tls_server
```

IV.Β) Πελάτες για Δοκιμή του Εξυπηρετητή σας

Για να δείτε ότι τρέχει το πρόγραμμα, χρειάζεστε ένα πρόγραμμα πελάτη (client). Αυτό μπορεί να γίνει με 3 τρόπους.

1. Ο πρώτος τρόπος (εδώ θα πρέπει να μιλήσετε εσείς το πρωτόκολλο) είναι με το εργαλείο openssl από τη γραμμή εντολών:

   - `openssl s_client -connect server-name:4433`

   προσέξετε να δώσετε το σωστό όνομα μηχανής πάνω στο οποίο τρέχει ο εξυπηρετητής και τη σωστή θύρα, αν ο εξυπηρετητής είναι στην τοπική σας μηχανή, θα πρέπει να δώσετε σαν server-name το localhost ή την IP διεύθυνση 127.0.0.1). Μετά την εκτέλεση της εντολής θα δείτε στην οθόνη τη διαδικασία ανταλλαγής πιστοποιητικών (TSL handshake and negotiation phase) και στο τέλος μπορεί κανείς διαδραστικά (μέσω ψευδο-κελύφους) να μιλήσει απευθείας στην γλώσσα του RFC1939. **Αυτό είναι χρονοβόρα διαδικασία και δεν συνίσταται.**

2. Ο δεύτερος τρόπος είναι γράφοντας ένα μικρό πρόγραμμα σε γλώσσα C (δείτε πρόγραμμα tls_client.c στο **as4-supplementary.zip**). **Αυτό είναι επίσης χρονοβόρα διαδικασία και δεν συνίσταται.**

3. **Ο τρίτος τρόπος (συνιστώμενος) είναι μέσω του CURL.**

   - `curl --ssl-reqd pop3s://your_pop3_server_address -u your_username:your_password -X "LIST"`

4. Αφού έχετε κάνει ένα από τα πιο πάνω, ο τέταρτος τρόπος είναι μέσω Mail client (π.χ. Mozilla Thunderbird, ή MS Outlook) το οποίο μπορείτε να δοκιμάσετε. Σε περίπτωση που ο server σας δουλεύει (στοιχειωδώς) με έστω ένα εμπορικό πελάτη θα πάρετε επιπλέον μονάδες.

Επίσης θα πρέπει να δημιουργήσετε ένα κατάλογο με όνομα mailbox/ μέσα στον ίδιο κατάλογο που είναι τα .c και .h αρχεία σας ο οποίος θα περιέχει τα email των χρηστών (ένα κατάλογο ανά χρήστη) που μπορεί να «σερβίρει» ο mail server (το όνομα του καταλόγου αυτού μπορείτε να το βρίσκετε ο κώδικάς σας μέσα στο config file – δες ενότητα IV.Δ). Μέσα στον κατάλογο mailbox θα βρίσκονται οι κατάλογοι των χρηστών. Π.χ. όταν ο χρήστης με username pop3user1 συνδεθεί στον POP3S server θα βρίσκει τα email του μέσα στον κατάλογο mailbox/pop3user1

IV.Γ) Περιγραφή Ζητούμενων Εντολών Πρωτοκόλλου POP3 (RFC1939)

Για τις ανάγκες της άσκησης, πρέπει να μπορείτε να χειριστείτε τις εντολές **USER** και **PASS** [RFC1939/§7], καθώς και τις εντολές **STAT**, **LIST**, **RETR** και **DELE** [RFC1939/§5]. Αν δοθεί κάποια διαφορετική εντολή από τον πελάτη, τότε ο εξυπηρετητής πρέπει να απαντά με κάποιο αρνητικό δείκτη κατάστασης όπως θα

περιγραφεί πιο κάτω (εκτός αν θέλετε να προσθέσετε κώδικα για να δέχεται ο server σας τις επιπλέον εντολές όπως περιγράφονται στο RFC1939).

Αρχικά, ο ασφαλής εξυπηρετητή POP3S ξεκινά δημιουργώντας ένα TCP socket στη θύρα 995. Όταν πελάτης POP3 επιθυμεί να κάνει χρήση της υπηρεσίας POP3S, δημιουργεί μια σύνδεση TCP με τον εξυπηρετητή POP3S. Όταν θα εγκαθιδρυθεί η σύνδεση, ο εξυπηρετητής POP3S στέλνει ένα χαιρετισμό (ο οποίος θα περιγραφεί στη συνέχεια). Στη συνέχεια, ο πελάτης και ο εξυπηρετητής POP3S ανταλλάζουν εντολές και απαντήσεις (αντίστοιχα) μέχρι να κλείσει ή να διακοπεί η σύνδεση.

Οι εντολές POP3S αποτελούνται από μια case-insensitive λέξη-κλειδί (command), πιθανώς ακολουθούμενη από ένα ή περισσότερα ορίσματα (arguments). Όλες οι εντολές τερματίζονται 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. Μπορείτε να το δείτε και γραμμένο σαν CRLF. Οι λέξεις-κλειδιά και τα ορίσματα αποτελούνται από εκτυπώσιμους ASCII χαρακτήρες και διαχωρίζονται μεταξύ τους από ένα μόνο SPACE χαρακτήρα. Οι λέξεις-κλειδιά αποτελούνται από τρεις ή τέσσερις χαρακτήρες. Κάθε όρισμα μπορεί να είναι μέχρι 40 χαρακτήρες.

Οι απαντήσεις στο πρωτόκολλο POP3S αποτελούνται από μια ένδειξη της τρέχουσας κατάστασης (που δίδεται πιο κάτω) και μια λέξη-κλειδί η οποία ακολουθείται ενδεχομένως από συμπληρωματικές πληροφορίες. Όλες οι απαντήσεις τερματίζονται από το CRLF. Οι απαντήσεις μπορεί να έχουν μέγεθος το πολύ μέχρι 512 χαρακτήρες, συμπεριλαμβανομένου του τερματισμού CRLF. Στην παρούσα φάση υπάρχουν δύο δείκτες κατάστασης (status indicators): θετικό ("+OK") και αρνητικό ("-ERR"). Ο εξυπηρετητής ΠΡΕΠΕΙ να στέλνει το "+OK" και "-ERR" με κεφαλαία.

Οι απαντήσεις σε ορισμένες εντολές μπορεί να περιέχουν πολλαπλές γραμμές. Σε αυτές τις περιπτώσεις, οι οποίες αναφέρονται σαφώς πιο κάτω, μετά την αποστολή της πρώτης γραμμής της απάντησης και του CRLF, κάθε επιπρόσθετη γραμμή τελειώνει με το CRLF. Όταν έχουν σταλεί όλες οι γραμμές της απάντησης, αποστέλλεται η τελική γραμμή που αποτελείται από μια τελεία "." (οκτάδα τερματισμού/termination octet) και το CRLF. Εάν οποιαδήποτε γραμμή της απάντησης ξεκινά με το termination octet, η γραμμή γίνεται "byte-stuffed», δηλαδή ο εξυπηρετητής βάζει μπροστά από τη γραμμή ακόμα ένα termination octet (τελεία). Όταν εξετάζεται μια απάντηση πολλαπλών γραμμών, ο πελάτης ελέγχει αν η γραμμή ξεκινά με το termination octet (τελεία). Αν ναι, και αυτή ακολουθείται από άλλους χαρακτήρες εκτός από το CRLF, τότε αφαιρείται το termination octet. Αν η γραμμή ξεκινά με το termination octet και ακολουθεί το CRLF τότε θεωρείται ότι είναι το τέλος της απάντησης του εξυπηρετητή. Η γραμμή που περιέχει το ".CRLF" δε θεωρείται μέρος της απάντησης του εξυπηρετητή.

Μια σύνοδος (session) POP3S κατά τη διάρκεια της ζωής περνά από ένα σύνολο καταστάσεων. Μόλις ανοίξει η σύνδεση TCP και ο εξυπηρετητής στείλει τον αρχικό χαιρετισμό, η σύνοδος εισέρχεται στην κατάσταση AUTHORIZATION. Σε αυτή την κατάσταση, ο πελάτης οφείλει να προσδιορίσει την ταυτότητα του (κωδικό/συνθηματικό) στον εξυπηρετητή. Μόλις ο πελάτης προσδιορίσει επιτυχώς τη ταυτότητά του, ο εξυπηρετητής κλειδώνει τους πόρους που σχετίζονται με το mailbox του πελάτη (στην περίπτωσή μας ένας κατάλογος που θα θεωρείτε το INBOX του πελάτη), και η σύνοδος εισέρχεται στην κατάσταση TRANSACTION. Σε αυτή την κατάσταση, ο πελάτης μπορεί να στείλει εντολές στον εξυπηρετητή. Όταν ο πελάτης

στείλει την εντολή "QUIT", η σύνοδος μπαίνει στην κατάσταση UPDATE. Στην κατάσταση αυτή, ο εξυπηρετητής απελευθερώνει τους πόρους που κλειδώθηκαν κατά τη διάρκεια της κατάστασης TRANSACTION και κλείνει τη σύνδεση TCP.
Ένας εξυπηρετητής POP3 ΠΡΕΠΕΙ να απαντά σε εντολές που στέλνονται από τον πελάτη οι οποίες είτε δεν αναγνωρίζονται, ή δεν είναι υλοποιημένες, ή είναι λάθος συντακτικά. Σε οποιαδήποτε από τις περιπτώσεις αυτές, ο εξυπηρετητής πρέπει να απαντά με τον αρνητικό δείκτη κατάστασης "-ERR". Ένας εξυπηρετητής POP3 ΠΡΕΠΕΙ να ανταποκρίνεται σε μια εντολή που στάληκε σε φάση που η σύνοδος είναι σε λάθος κατάσταση απαντώντας με τον αρνητικό δείκτη κατάστασης "-ERR".
Ο εξυπηρετητής που θα υλοποιήσετε θα πρέπει να έχει χρονοδιακόπτη αυτόματης διακοπής σύνδεσης εφόσον ο πελάτης παραμείνει αδρανής για κάποιο διάστημα (π.χ., 1 λεπτό). Αν μέσα στο χρόνο αυτό ο πελάτης αποστείλει κάποια εντολή τότε ο χρονοδιακόπτης θα μηδενίζεται. Όταν ο χρονοδιακόπτης λήξει, η σύνοδος ΔΕΝ θα εισέρχεται σε κατάσταση UPDATE και ο εξυπηρετητής θα πρέπει να τερματίζει τη σύνδεση TCP χωρίς να αποστέλλει οποιαδήποτε απάντηση προς τον πελάτη. Δείτε το Παράρτημα για τη λειτουργία αυτή.

Κατάσταση AUTHORIZATION
Μόλις εγκαθιδρυθεί μια TCP σύνδεση από έναν πελάτη, ο εξυπηρετητής POP3 στέλνει ένα χαιρετισμό της μορφής:

S: + OK POP3 server ready

Στη φάση αυτή η σύνοδος POP3 εισέρχεται στην κατάσταση AUTHORIZATION. Ο πελάτης πρέπει να προσδιορίσει την ταυτότητά του στον εξυπηρετητή POP3. Στην παρούσα άσκηση αυτό θα γίνεται με χρήση του συνδυασμού εντολών USER και PASS. Μόλις ο εξυπηρετητής POP3 προσδιορίσει την ταυτότητα του πελάτη αποκτά αποκλειστική πρόσβαση στο mailbox του πελάτη και απαντά με τον θετικό δείκτη κατάστασης "+OK". Στα πλαίσια της άσκησης αυτής θεωρείστε ότι το mailbox κάθε πελάτη θα είναι ένας κατάλογος που (με τον κωδικό του πελάτη) που θα περιέχει μέσα τα μηνύματα ηλεκτρονικού ταχυδρομείου του πελάτη σε μορφή αρχείων text.
Στη φάση αυτή η σύνοδος μπαίνει στην κατάσταση TRANSACTION. Εάν για κάποιο λόγο το mailbox δεν μπορεί να ανοίξει ο εξυπηρετητής POP3 θα ανταποκρίνεται με αρνητική δείκτη κατάστασης και θα κλείνει τη σύνδεση. Οι δύο εντολές USER και PASS συντάσσονται όπως φαίνεται πιο κάτω:

USER name

Όρισμα: τον κωδικό (username) που να προσδιορίζει τον κατάλογο μέσα στο mailbox του (υποχρεωτικό)
Περιορισμοί: μπορεί να δίνεται από τον πελάτη μόνο στην κατάσταση AUTHORIZATION μετά τον χαιρετισμό ή μετά από μια ανεπιτυχή εντολή USER ή PASS.
Πιθανές απαντήσεις:
+OK *name* is a valid mailbox
-ERR never heard of mailbox *name*
Παραδείγματα:
C: USER frated
S: -ERR never heard of mailbox frated
...
C: USER mrose

S: +OK mrose is a valid mailbox

PASS string

Όρισμα: το συνθηματικό (password) του χρήστη (υποχρεωτικό)
Περιορισμοί: μπορεί να δίνεται από τον πελάτη μόνο στην κατάσταση AUTHORIZATION μετά την εντολή USER.
Πιθανές απαντήσεις:
+OK maildrop locked and ready
-ERR invalid password
Παραδείγματα:
C: USER mrose
S: +OK mrose is a valid mailbox
C: PASS secret
S: -ERR invalid password
...
C: USER mrose
S: +OK mrose is a valid mailbox
C: PASS secret
S: +OK maildrop locked and ready

Όταν ο εξυπηρετητής ανοίξει το mailbox του πελάτη θα αναθέτει ένα μοναδικό αριθμό σε κάθε μήνυμα που είναι αποθηκευμένο και θα σημειώνει το μέγεθος του κάθε μηνύματος σε οκτάδες (bytes). Το πρώτο μήνυμα μέσα στο mailbox θα έχει τον αριθμό 1, το δεύτερο το 2 και ούτω καθ'εξής. Σε όλες τις εντολές και απαντήσεις του πρωτοκόλλου POP3, οι αριθμοί των μηνυμάτων και το μέγεθός τους εκφράζονται με βάση το 10 (δηλαδή, ακολουθούν το δεκαδικό σύστημα).
Πιο κάτω φαίνεται η εντολή "QUIT" όταν χρησιμοποιείται στην κατάσταση AUTHORIZATION:

QUIT

Ορίσματα: Κανένα
Περιορισμοί: Κανένας
Πιθανές απαντήσεις:
+OK
Παραδείγματα:
C: QUIT
S: +OK *user* POP3 server signing off

Μετά την εκτέλεση της εντολής QUIT ο εξυπηρετητής απελευθερώνει τη πρόσβαση πάνω στο mailbox του πελάτη και κλείνει τη σύνδεση TCP.

Κατάσταση TRANSACTION

Μόλις ο πελάτης προσδιορίσει με επιτυχία τη ταυτότητα του στον εξυπηρετητή και ο εξυπηρετητής έχει αποκτήσει πρόσβαση πάνω στο κατάλληλο κατάλογο μέσα στο mailbox, η σύνοδος μπαίνει στην κατάσταση TRANSACTION. Στη φάση αυτή ο πελάτης μπορεί δίνει διαδοχικά εντολές. Μετά από κάθε εντολή, ο εξυπηρετητής αποκρίνεται με μια απάντηση. Στο τέλος ο πελάτης αποστέλλει την εντολή QUIT και η σύνοδος εισέρχεται στην κατάσταση UPDATE.
Πιο κάτω παρατίθενται οι εντολές που έχουν ισχύ στην κατάσταση TRANSACTION:

STAT

Ορίσματα: Κανένα

Περιορισμοί: Μπορεί να δοθεί μόνο στην κατάσταση TRANSACTION

Συζήτηση: Ο εξυπηρετητής στέλνει θετικό δείκτη κατάστασης +OK μαζί με κάποιες πληροφορίες για το mailbox του χρήστη (για τον δικό του κατάλογο μέσα στον κατάλογο mailbox). Πιο συγκεκριμένα το +OK ακολουθείται από ένα ακριβώς κενό, ακολουθεί ο αριθμός των μηνυμάτων μέσα στο mailbox του χρήστη, ακόμα ένα κενό και το μέγεθος του mailbox του χρήστη σε bytes. Τα μηνύματα που χαρακτηρίζονται ως ***deleted*** (βλέπε παρακάτω εντολή DELE) δεν προσμετρώνται.

Πιθανές απαντήσεις:

+OK nn mm

Παραδείγματα:

C: STAT
S: + OK 2 320

LIST [msg]

Ορίσματα: ένας αριθμός μηνύματος (προαιρετικά), ο οποίος εάν δίνεται, δεν μπορεί να αναφέρεται σε ένα μήνυμα που χαρακτηρίζεται ως deleted.

Περιορισμοί: Μπορεί να δοθεί μόνο στην κατάσταση TRANSACTION

Συζήτηση: Εάν δοθεί ο αριθμός μηνύματος, ο εξυπηρετητής στέλνει μια γραμμή που περιέχει το δείκτη θετικής κατάστασης +OK ακολουθούμενο από ένα κενό, τον αριθμό του μηνύματος, ακόμα ένα κενό και το μέγεθος του μηνύματος σε bytes. Η γραμμή αυτή ονομάζεται scan listing για το μήνυμα αυτό. Αν δεν δίνεται κανένα όρισμα, ο εξυπηρετητής στέλνει μια θετική απόκριση πολλαπλών γραμμών. Μετά το αρχικό +OK, για κάθε μήνυμα μέσα στο mailbox του χρήστη, στέλνεται ξεχωριστή γραμμή που περιέχει αριθμό του μηνύματος, ένα κενό και το μέγεθος του μηνύματος σε bytes. Αν δεν υπάρχουν μηνύματα μέσα στο mailbox του χρήστη, ο εξυπηρετητής απαντά με το +OK μόνο. Τα μηνύματα που χαρακτηρίζονται ως *deleted* δεν προσμετρώνται.

Πιθανές απαντήσεις:

+OK x messages (x octets)
-ERR no such message, only x messages in mailbox

Παραδείγματα:

C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
...
C: LIST 2
S: +OK 2 200
...
C: LIST 3
S: -ERR no such message, only 2 messages in mailbox

RETR msg

Ορίσματα: ένας αριθμός μηνύματος (υποχρεωτικό) που δεν μπορεί να αναφέρεται σε μήνυμα που χαρακτηρίζεται ως deleted.
Περιορισμοί: Μπορεί να δοθεί μόνο στην κατάσταση TRANSACTION
Συζήτηση: Εάν ο εξυπηρετητής απαντήσει θετικά, τότε επιστρέφεται απάντηση πολλαπλών γραμμών. Μετά το αρχικό +OK, στέλνεται το μήνυμα που αντιστοιχεί στον δοσμένο αριθμό μηνύματος, προσέχοντας να κάνετε byte-stuff το χαρακτήρα τερματισμού.

Πιθανές απαντήσεις:
+OK x octets
-ERR no such message
Παραδείγματα:
C: RETR 1
S: +OK 120 octets
S: <Ο εξυπηρετητής στέλνει ολόκληρο το μήνυμα εδώ>
S: .

DELE msg

Ορίσματα: ένας αριθμός μηνύματος (υποχρεωτικό) που δεν μπορεί να αναφέρεται σε μήνυμα που χαρακτηρίζεται ως deleted.
Περιορισμοί: Μπορεί να δοθεί μόνο στην κατάσταση TRANSACTION
Συζήτηση: Ο εξυπηρετητής σημειώνει το μήνυμα σαν ***deleted***. Κάθε μελλοντική αναφορά (κάποιας POP3 εντολής) στον αριθμό που σχετίζεται με κάποιο μήνυμα που χαρακτηρίζεται ως deleted θα επιστρέφει λάθος. Στην πραγματικότητα ο εξυπηρετητής δεν διαγράφει το μήνυμα μέχρι η σύνοδος να φτάσει στην κατάσταση UPDATE.

Πιθανές απαντήσεις:
+OK message x deleted
-ERR no such message

Παραδείγματα:
C: DELE 1
S: + OK message 1 deleted
...
C: DELE 2
S: -ERR no such message

Κατάσταση UPDATE

Όταν ο πελάτης στείλει την εντολή QUIT στην κατάσταση TRANSACTION, ο εξυπηρετητής εισέρχεται στην κατάσταση UPDATE. (Σημειώστε ότι αν ο πελάτης στείλει την εντολή QUIT από την κατάσταση AUTHORIZATION, η POP3 σύνοδος τερματίζει αλλά δεν μπαίνει στην κατάσταση UPDATE).
Εάν μια σύνοδος τερματιστεί για κάποιο άλλο λόγο (χωρίς να στείλει ο πελάτης την εντολή QUIT) η σύνοδος δεν μπαίνει στην κατάσταση UPDATE και ΔΕΝ πρέπει να διαγραφεί κανένα μήνυμα από το mailbox του χρήστη.

QUIT

Ορίσματα: Κανένα
Περιορισμοί: Κανένας
Συζήτηση: Ο εξυπηρετητής διαγράφει από το mailbox όλα τα μηνύματα που έχουν χαρακτηριστεί ως ***deleted*** και στέλνει ανάλογη απάντηση χρησιμοποιώντας τους δείκτες κατάστασης +OK και -ERR. Σε καμία περίπτωση δεν μπορεί ο εξυπηρετητής να διαγράψει μηνύματα που δεν έχουν χαρακτηριστεί ως deleted.
Ανεξαρτήτως η διαγραφή αρχείων υπήρξε επιτυχής ή όχι, ο εξυπηρετητής απελευθερώνει τη πρόσβαση πάνω στο mailbox του χρήστη και κλείνει τη σύνδεση TCP.

Πιθανές απαντήσεις:

+ OK
-ERR some deleted messages not removed

Παραδείγματα:

C: QUIT
S: + OK *user* POP3 server signing off (mailbox empty)
...
C: QUIT
S: + OK *user* POP3 server signing off (2 messages left)

IV.Δ) Παράλληλος (πολύ-διεργασιακός) εξυπηρετητής

Ο POP3S 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 διεργασιών ή νημάτων μέσα στη διεργασία του POP3S server. Μια ιδέα θα ήταν, όταν παίρνει μια αίτηση από πελάτη, να δημιουργεί μια διεργασία για να την εξυπηρετήσει (όπως στο μάθημα), ενώ η αρχική διεργασία να περιμένει νέες αιτήσεις. Φυσικά, όταν μια διεργασία 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 ή διεργασίων στην εφαρμογή, κάτι που μπορεί να αποβεί εξαιρετικά προβληματικό σε κάποιες περιπτώσεις.

Μια άλλη, καλύτερη, ιδέα είναι το αρχική διεργασία να δημιουργήσει ένα **process-pool,** δηλαδή να δημιουργήσει εξ αρχής ένα σταθερό αριθμό **διεργασιών εργατών** (που το πλήθος τους να δίνεται) και όταν υπάρχει αίτηση για εξυπηρέτηση να την αναθέτει σε κάποια διεργασία που δεν έχει δουλειά. Οι διεργασίες, αφού εξυπηρετήσουν ένα πελάτη, δεν τερματίζουν, αλλά μεταβαίνουν σε κατάσταση αναμονής. Φυσικά, αν δεν υπάρχει διαθέσιμη διεργασία, το αρχικό θα πρέπει να περιμένει μέχρι να υπάρξει, χωρίς να δέχεται νέες αιτήσεις. Συγκεκριμένα εάν υπάρξουν περισσότερες αιτήσεις από το μέγιστο αριθμό νημάτων ή διεργασιών 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 POP3S server σας, εκτός από το πλήθος των νημάτων ή διεργασιών, 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:

```
# POP3S server Configuration File

# The Number of Processes in the pool
PROCESSES=40

# The Port number of the POP3S server
PORT=995

# The MAILBOX folder of the POP3S server
HOME=./mailbox

# The logifile
LOGS=logs.txt
…
```

Για να δοκιμάσετε τον mail server σας μπορείτε να δημιουργήσετε ένα δικό σας κατάλογο και να τοποθετήσετε εκεί τα αρχεία της αρεσκείας σας ή να αξιοποιήσετε υποσύνολο των αρχείων του **ENRON dataset**, διαθέσιμα μέσω του συνοδευτικού αρχείου **as4-supplementary.zip.**

IV.E) Δημιουργία LOG αρχείου μέσω Message Queue

Ο POP3S server θα κρατά βασικές πληροφορίες των πελατών (client) που ενώνονται στο σύστημα. Μια συγκεκριμένη διεργασία (που θα είναι ζωντανή καθόλη τη διάρκεια του προγράμματος) θα έχει την υπευθυνότητά του **logger** δηλαδή να γράφει μέσα στο log file. Ο logger θα ακούει σε ένα **message queue** για να μαθαίνει ποια εντολή και από ποια διεύθυνση IP λήφθηκε. Κάθε διεργασία εργάτης που εξυπηρετεί ένα πελάτη, όταν δέχεται μια εντολή θα γράφει σε αυτό το message queue το IP του πελάτη και την εντολή για να το αξιοποιήσει ο logger.

Σε ένα αρχείο **logs.txt** θα καταγράφεται η κάθε POP3S εντολή που φτάνει στο server μαζί με την ώρα άφιξης (σε μορφή UNIX timestamp) και τη διεύθυνση IP του πελάτη όπως φαίνεται στο πιο κάτω παράδειγμα:

```
1731570447 194.10.15.3 USER
1731570451 194.10.15.3 PASS
1731570457 194.10.15.3 STAT
1731570502 194.10.15.3 LIST
1731570677 194.10.15.3 RETR
```

V. Παράδειγμα συνόδου

```
S: <wait for connection on TCP port 995>
C: <open connection>
S:    +OK POP3 server ready
C:    USER mrose
S:    +OK mrose is a valid mailbox
```

```
C:    PASS secret
S:    +OK mrose's maildrop has 2 messages (320 octets)
C:    STAT
S:    +OK 2 320
C:    LIST
S:    +OK 2 messages (320 octets)
S:    1 120
S:    2 200
S:    .
C:    RETR 1
S:    +OK 120 octets
S:    <the POP3 server sends message 1>
S:    .
C:    DELE 1
S:    +OK message 1 deleted
C:    RETR 2
S:    +OK 200 octets
S:    <the POP3 server sends message 2>
S:    .
C:    DELE 2
S:    +OK message 2 deleted
C:    QUIT
S:    +OK dewey POP3 server signing off (maildrop empty)
C:  <close connection>
S:  <wait for next connection>
```

VII. Ανάπτυξη Λογισμικού

Η άσκηση αυτή θα υλοποιηθεί σε ομάδες όπως έχουν αναρτηθεί στο Moodle για την παρουσίαση, των οποίων τα άτομα αναμένεται να συμβάλουν ισομερώς σε χρόνο και ουσιαστική δουλειά.

VIII. Αξιολόγηση

Α) Τι πρέπει να παραδώσετε;

- **Στο Moodle:** Ενα αρχείο **pop3s.tar.gz** το οποίο θα περιέχει:
  1. Τον πηγαίο κώδικα (αρχεία .c/.h) μαζί με το σχετικό Makefile,
  2. Ένα README.txt αρχείο οποίο θα δίδει οδηγίες χρήσης του συστήματός σας (περίπου 1 σελίδα) και
  3. Architecture (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).

Β) Κριτήρια Αξιολόγησης.

1. **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 (unit tests) 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. **Ορθότητα Λειτουργίας:** Το σύστημα θα πρέπει να διεκπεραιώνει ορθά τις λειτουργίες του συστήματος όπως αυτές περιγράφονται σε αυτή την εκφώνηση και το RFC1939. 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 (RFC1939) καθώς επίσης να ανακαλύψετε νέες συναρτήσεις πέρα από αυτές που διδαχθήκατε ήδη στις διαλέξεις.

Καλή Επιτυχία !

Παράρτημα (Προχωρημένο)

Για να υλοποιήσετε το timeout με το οποίο να περιμένει μια διεργασία τον πελάτη να στείλει δεδομένα και αν λήξει ο χρόνος (χωρίς να στείλει ο πελάτης) να κλείνει η σύνδεση, μπορείτε να βρείτε χρήσιμη την συνάρτηση select() – non blocking I/O. Η συνάρτηση αυτή περιμένει (blocking function) μέχρι να συμβεί ένα από τα πιο κάτω 3:
α) ο file descriptor (client socket) λαμβάνει δεδομένα
β) ληφθεί ένα σήμα που διακόπτει τη λειτουργία της συνάρτησης
γ) λήγει το timeout
Διαβάστε το manual (https://man7.org/linux/man-pages/man2/select.2.html) και ιδιαιτέρως το κομμάτι με τα file descriptor sets.

Δείτε πιο κάτω ένα κομμάτι κώδικα που μπορεί να βοηθήσει:

```
struct timeval timeoutValue;
timeoutValue.tv_sec = timeoutInSeconds; //
timeoutValue.tv_usec = 0;

// file descriptor set
fd_set client_set;
// clears (removes all file descriptors from) set
FD_ZERO(&client_set);
// adds the file descriptor (clientSocket) to set
FD_SET(clientSocket, &client_set);
result = select(clientSocket + 1, &client_set, NULL, NULL,
&timeoutValue);
```