ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

Τμήμα Πληροφορικής

ΕΠΛ 421 – Προγραμματισμός Συστημάτων

ΑΣΚΗΣΗ 4 – Ανάπτυξη Ασφαλούς Παράλληλου Εξυπηρετητή Αρχείων FTP

Διδάσκων: Δημήτρης Ζεϊναλιπούρ
Υπεύθυνος Εργαστηρίου: Παύλος Αντωνίου

Ημερομηνία Ανάθεσης: Παρασκευή 01/04/2022
Ημερομηνία Παράδοσης: Παρασκευή 15/04/2022

(να υποβληθεί ηλεκτρονικά στο Moodle)

Ι. 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* και της *δομής* του.

ΙΙ. Αποστολή / Ανάκτηση αρχείων μέσω πρωτοκόλλου FTP μέσα από ασφαλές κανάλι επικοινωνίας

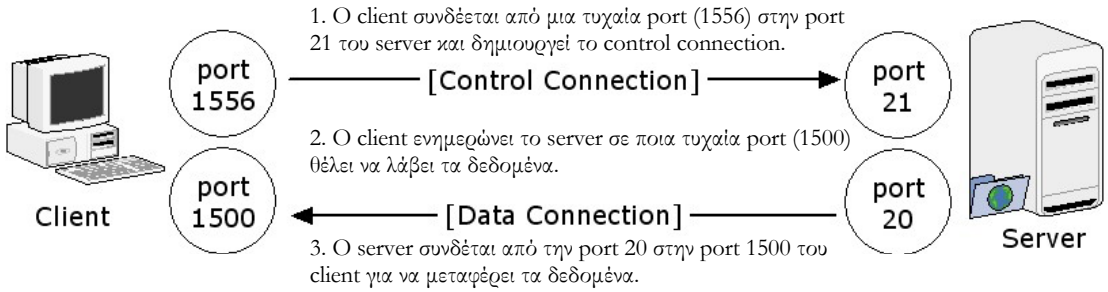
1. Τρόπος Λειτουργίας Πρωτοκόλλου FTP

Το πρωτόκολλο FTP λειτουργεί με βάση το μοντέλο πελάτη-εξυπηρετητή (client-server) επιτρέποντας την αποστολή/ανάκτηση αρχείων μεταξύ των 2 οντοτήτων.
Αρχικά ο FTP server ανοίγει την θύρα (port) 21 περιμένοντας έναν FTP client να συνδεθεί. Στη συνέχεια ο client ξεκινά μια νέα σύνδεση από μια τυχαία θύρα προς την θύρα 21 του server. Μόλις γίνει η σύνδεση παραμένει ανοιχτή για όλη τη διάρκεια της συνόδου FTP. Η συγκεκριμένη σύνδεση ονομάζεται σύνδεση ελέγχου (**control connection**) και μέσα από το κανάλι αυτό στέλνονται όλες οι εντολές από τον client και οι απαντήσεις από το server, αλλά όχι δεδομένα (αρχεία). Έπεται η δημιουργία της σύνδεσης δεδομένων (**data connection**), της σύνδεσης με την οποία μεταφέρονται τα δεδομένα. Υπάρχουν δύο τρόποι για να δημιουργηθεί η σύνδεση δεδομένων μέσω της σύνδεσης ελέγχου, (α) με χρήση της ενεργητικής λειτουργίας (active mode) στην οποία τα δεδομένα σπρώχνονται (push) από τον server στον client ή (β) με χρήση της παθητικής λειτουργίας (passive mode) στην οποία τα δεδομένα αντλούνται (pull) από τον client.

Α) Ενεργητική λειτουργία

Στην ενεργητική λειτουργία (active mode) ο FTP client διαλέγει μια τυχαία θύρα στην οποία δέχεται τα δεδομένα της σύνδεσης. Ο client στέλνει τον αριθμό της θύρας, στην

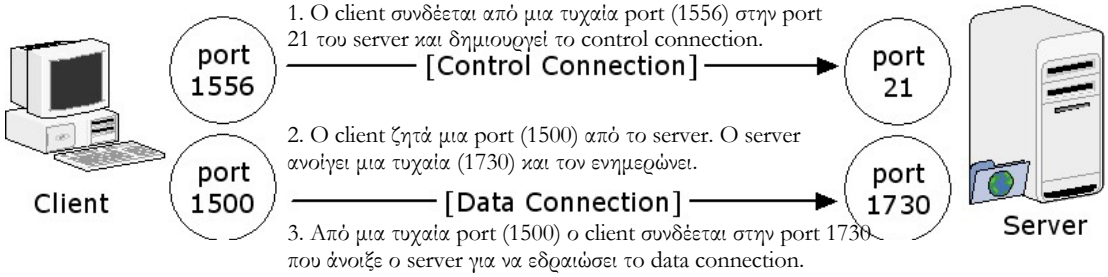
οποία επιθυμεί να "ακούει" (listen) για εισερχόμενες συνδέσεις. Ο FTP server δημιουργεί μια σύνδεση από την θύρα 20 στην ανοιχτή θύρα του client για τη μεταφορά των δεδομένων. Οποιαδήποτε πληροφορία ζητήσει ο client, ανταλλάσσεται με βάση αυτή τη σύνδεση, που βασίζεται στο TCP. Όταν η μεταφορά ολοκληρωθεί ο server κλείνει τη σύνδεση αποστέλλοντας ένα πακέτο FIN, όπως σε κάθε σύνδεση βασισμένη στο TCP. Κάθε φορά που ο client ζητάει δεδομένα, δημιουργείται κατά παρόμοιο τρόπο μια σύνδεση δεδομένων και η διαδικασία επαναλαμβάνεται.



Εικόνα 1: Διαγραμματική και διαλογική αναπαράσταση της ενεργητικής λειτουργίας FTP.

Β) Παθητική λειτουργία

Στην παθητική λειτουργία (passive mode) ο client ζητά από τον server να διαλέξει μια τυχαία θύρα, στην οποία θα "ακούει" (listen) για την σύνδεση δεδομένων (data connection). Ο server ενημερώνει τον client για την θύρα την οποία έχει διαλέξει και ο client συνδέεται σε αυτή για τη μεταφορά των δεδομένων. Η μεταφορά ολοκληρώνεται όπως και στην ενεργητική λειτουργία (active mode), αφού η σύνδεση δεδομένων βασίζεται στο TCP.



Εικόνα 2: Διαγραμματική και διαλογική αναπαράσταση της παθητικής λειτουργίας FTP.

Σύμφωνα με την παράγραφό 3.4 του RFC959 το οποίο περιγράφει τη λειτουργία του πρωτοκόλλου FTP, κάθε μεταφορά αρχείου πρέπει να τερματίζεται με το end-of-file (EOF) το οποίο μπορεί να δηλώνεται ρητώς ή να υπονοείται με το κλείσιμο της σύνδεσης δεδομένων.

2. Εντολές FTP

Οι εντολές που περιγράφονται πιο κάτω είναι ένα υποσύνολο των εντολών που υποστηρίζει το πρωτόκολλο FTP, το οποίο δίνουμε απλά για κατανόηση του πρωτοκόλλου. **Στην ενότητα ΙΙΙ θα εξειδικεύσουμε τις εντολές τις οποίες θα πρέπει να υλοποιήσετε στα πλαίσια της παρούσας άσκησης.**

Εντολές ελέγχου πρόσβασης

USER <username> καθορισμός κωδικού που είναι καταχωρημένος στο server
PASS <password> καθορισμός συνθηματικού που είναι καταχωρημένος στο server
CWD <pathname> αλλαγή καταλόγου **(όπως cd)**

CDUP μετάβαση στο «γονικό» κατάλογο **(όπως cd ..)**
QUIT έξοδος

Εντολές προσδιορισμού παραμέτρων
PORT h1,h2,h3,h4,p1,p2 Δήλωση ενεργητικής λειτουργίας: κοινοποίηση τοπικής θύρας δεδομένων.
PASV Δήλωση παθητικής λειτουργίας: αίτηση για αποστολή θύρας αφούγκρασης (listening) του server για παραλαβή δεδομένων
TYPE καθορισμός τύπου δεδομένων: ascii (για κείμενο), image (για εικόνες και binary αρχεία), default τιμή: ascii

Εντολές υπηρεσίας
RETR <pathname> ανάκτηση (retrieve) αρχείου. Η εντολή αυτή μπορεί να εκτελεστεί αν υπάρχει ανοικτή σύνδεση δεδομένων (δηλαδή μετά από την εντολή PORT ή PASV). **(όπως cp remote_file local_file)**
STOR <pathname> αποστολή (store) αρχείου. Η εντολή αυτή μπορεί να εκτελεστεί αν υπάρχει ανοικτή σύνδεση δεδομένων (δηλάδη μετά από την εντολή PORT ή PASV). **(όπως cp local_file remote_file)**
APPE <pathname> αποστολή αρχείου και προσάρτηση σε υφιστάμενο αρχείο που βρίσκεται στο server. Η εντολή αυτή μπορεί να εκτελεστεί αν υπάρχει ανοικτή σύνδεση δεδομένων (δηλαδή μετά από την εντολή PORT ή PASV) **(όπως cat local_file >> remote_file)**
ABOR διακοπή (abort) προηγούμενης εντολής υπηρεσίας
PWD εμφάνιση τρέχοντος καταλόγου **(pwd)**
LIST μεταφορά της λίστας των αρχείων **(ls -la)** Η εντολή αυτή μπορεί να εκτελεστεί αν υπάρχει ανοικτή σύνδεση δεδομένων (δηλαδή μετά από την εντολή PORT ή PASV). Η εντολή αυτή επιστρέφει παρόμοια αποτελέσματα με την εντολή ls –la. Οι καταλόγοι επισημαίνονται με “d”.
DELE <pathname> διαγραφή αρχείου **(rm)**
MKD <pathname> δημιουργία καταλόγου **(mkdir)**
RMD <pathname> διαγραφή καταλόγου **(rmdir)**
SIZE <pathname> μέγεθος αρχείου (οκτάδες, 8-bit byte) σαν δεκαδικός αριθμός
STAT <pathname> εάν δοθεί όρισμα ένα directory τότε είναι το ίδιο με την εντολή list με τη διαφορά ότι η απάντηση από το server έρχεται μέσα από τη σύνδεση ελέγχου (χωρίς να πρέπει να ανοίξει σύνδεση δεδομένων)
Περισσότερες πληροφορίες μπορείτε να βρείτε στο RFC959 (http://www.ietf.org/rfc/rfc0959.txt) ή στον πιο κάτω σύνδεσμο: http://www.nsftools.com/tips/RawFTP.htm

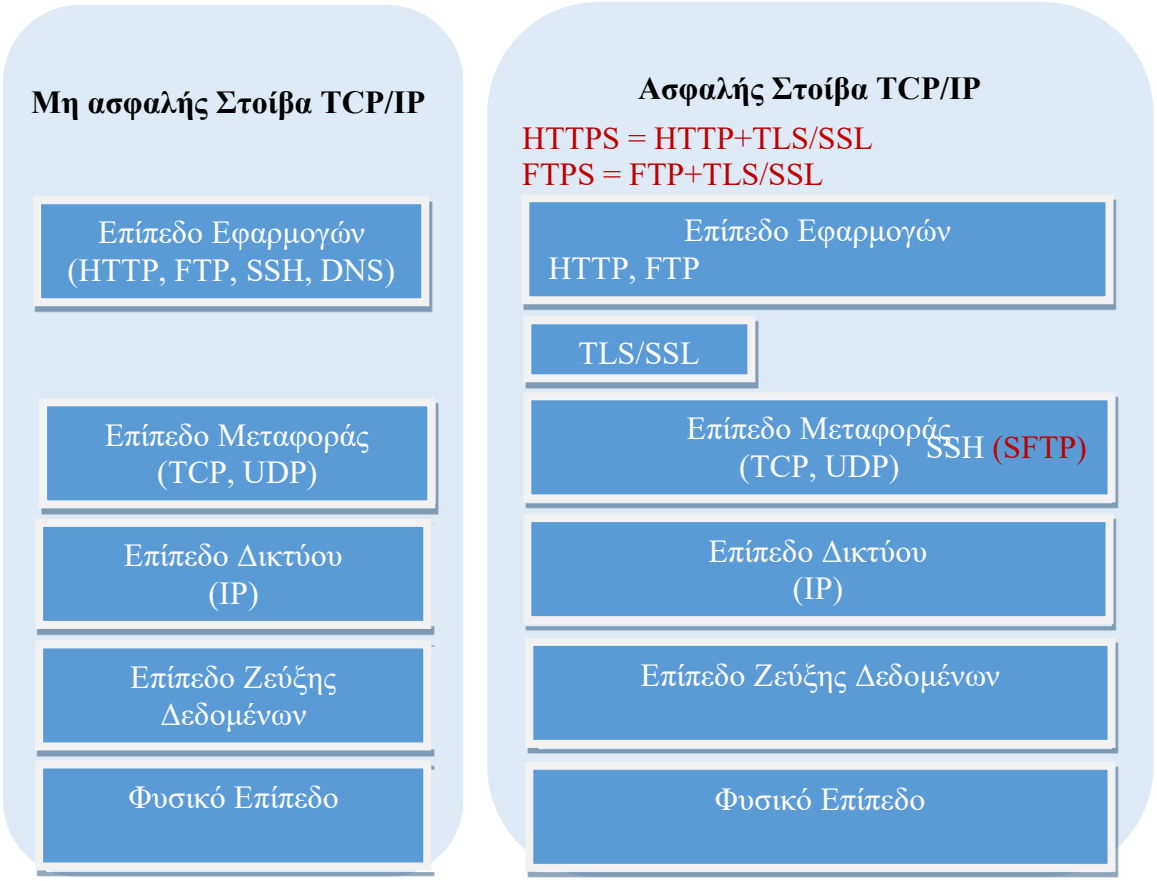
3. Αποστολή μηνυμάτων FTP μέσα από ασφαλές κανάλι επικοινωνίας

Το πρωτόκολλο FTP δεν προδιαγράφει ασφαλή μετάδοση εντολών και δεδομένων και ως εκ τούτου η χρήση του τείνει να εξαλειφθεί. Υπάρχουν διαθέσιμα 2 πρωτόκολλα που παρέχουν ασφαλή μεταφορά αρχείων: (α) FTPS (FTP over SSL) και (β) SFTP (SSH FTP).
Το FTPS είναι βασικά το FTP με την προσθήκη ενός (υπό)στρώματος ασφάλειας. Πιο συγκεκριμένα, το FTPS περιέχει σύνδεση ελέγχου και σύνδεση δεδομένων (όπως και το FTP) αλλά η ειδοποιός διαφορά είναι ότι παρέχει ασφάλεια (κρυπτογράφηση) πληροφοριών είτε και στις 2 συνδέσεις ή μόνο στη σύνδεση δεδομένων. Η ασφάλεια των συνδέσεων επιτυγχάνεται στη βάση των πρωτοκόλλων SSL/TLS (Secure Sockets Layer/Transport Layer Security).

Το πρωτόκολλο SSL υλοποιήθηκε αρχικά από την Netscape εμφανίστηκε στη σκηνή το 1995 σαν SSL 2.0 ή SSLv2 (το SSL 1.0 δεν εμφανίστηκε ποτέ δημοσίως). Η έκδοση SSL 2.0 αντικαταστάθηκε νωρίς από την έκδοση SSL 3.0 (ή SSLv3) το 1996 μετά από ένα αριθμό τρωτών σημείων (vulnerabilities) που εμφανίστηκαν. Το TLS εμφανίστηκε το 1999 σαν μια αναβαθμισμένη, πιο ασφαλής έκδοση του SSL, στη βάση του πρωτοκόλλου SSL 3.0. Η τρέχουσα έκδοση του TLS είναι η v1.2, ενώ βρίσκεται σε draft μορφή η έκδοση v1.3. Η έκδοση SSL 2.0 δεν είναι διαλειτουργική με την έκδοση SSL 3.0 και η έκδοση SSL 3.0 δεν είναι διαλειτουργική με την έκδοση 1 του TLS. Τόσο το SSL 2.0 όσο και το 3.0 έχουν χαρακτηριστεί ως πεπαλαιωμένα (deprecated) από το IETF (το 2011 και το 2015, αντίστοιχα). Στην παρούσα άσκηση ο FTP server που θα υλοποιήσετε θα υποστηρίζει τα πρωτόκολλα SSL 2.0 και SSL 3.0 (δείτε συνάρτηση SSLv23_server_method στον κώδικα του server που θα παρουσιάσουμε πιο κάτω).

Το SFTP είναι ένα εντελώς διαφορετικό πρωτόκολλο το οποίο βασίζεται στο πρωτόκολλο SSH (Secure SHell) παρά στο πρωτόκολλο FTP. Το SFTP χρησιμοποιεί μόνο μια σύνδεση μέσα από την οποία όλες οι πληροφορίες είναι κρυπτογραφημένες.

Τα πρωτόκολλα SSH και SSL έχουν πολλές ομοιότητες. Δημιουργούν και τα δύο μια ασφαλή σήραγγα (secure tunnel) μεταφοράς δεδομένων η οποία παρέχει κρυπτογράφηση δεδομένων (data encryption), πιστοποίηση ταυτότητας του server (server authentication), πιστοποίηση ταυτότητας του client (client authentication), και μηχανισμούς ακεραιότητας (μη αλλοίωσης) δεδομένων (data integrity). Μια από τις πιο βασικές διαφορές τους είναι ότι το SSL εμπλέκει τη χρήση ψηφιακών πιστοποιητικών (certificates) τύπου X.509 για την πιστοποίηση ταυτότητας του client και του server ενώ το SSH χρησιμοποιεί ένα άλλο εσωτερικό μηχανισμό για την πιστοποίηση ταυτότητας. Ένα πιστοποιητικό X.509 περιέχει το δημόσιο κλειδί (public key) μιας οντότητας και την ταυτότητα της οντότητας (π.χ. όνομα κόμβου/hostname, όνομα οργανισμού, όνομα ιδιώτη) και είναι υπογεγραμμένο από κάποια αρχή πιστοποίησης (Certificate Authority, CA) ή αυτό-υπογεγραμμένο (self-signed) με το ιδιωτικό κλειδί (private key) της ίδιας της οντότητας. Και επειδή το SSL χρησιμοποιεί



ψηφιακά πιστοποιητικά, απαιτεί συνεπώς την παρουσία υποδομής δημόσιου κλειδιού (Public Key Infrastructure, PKI) και τη συμμετοχή μιας αρχής πιστοποίησης (CA). Επίσης μια άλλη μεγάλη διαφορά είναι ότι το SSH έχει ενσωματωμένη περισσότερη λειτουργικότητα. Για παράδειγμα, από μόνο του, το πρωτόκολλο SSH μπορεί να επιτρέψει στους χρήστες να συνδεθούν σε ένα server και να εκτελούν εντολές κελύφους εξ αποστάσεως (βλέπε εργαλείο απομακρυσμένης πρόσβασης Putty που χρησιμοποιείται και για σύνδεση στις μηχανές εργαστηρίου). Δημιουργεί επί της ουσίας ένα ασφαλές κέλυφος (shell) για εκτέλεση εντολών σε απομακρυσμένο server. Το SSL δεν έχει αυτή τη δυνατότητα. Λειτουργεί μόνο σε συνδυασμό με άλλα πρωτόκολλα (π.χ. HTTP, FTP) για να παρέχει ασφαλές κανάλι επικοινωνίας στα πρωτόκολλα αυτά (και να γίνουν HTTPS και FTPS).
Το SFTP είναι μέρος του πρωτοκόλλου SSH και δεν προδιαγράφεται από ξεχωριστό RFC και ούτε του έχει ανατεθεί ξεχωριστό port. Υπάρχει διαθέσιμο το δωρεάν εργαλείο OpenSSH το οποίο υλοποιεί το πρωτόκολλο SSH. Μέρος του εργαλείου OpenSSH είναι το πρόγραμμα-πελάτης sftp και το πρόγραμμα-εξυπηρετητής sftp-server. Η λειτουργία του sftp προδιαγράφεται σε ένα Internet Draft* του οποίου η έκδοση 3 υλοποιείται στο εργαλείο OpenSSH (σύμφωνα με την πηγή αυτή). Το sftp μπορεί να καλεστεί από τη γραμμή εντολών (κέλυφος) και επιτρέπει ασφαλή μεταφορά (λήψη και αποστολή) δεδομένων από και προς κάποια μηχανή που έχει ενεργοποιημένο το sftp-server. Το πρόγραμμα sftp-server δεν καλείται άμεσα αλλά εμμέσως μέσω του προγράμματος sshd το οποίο ενεργοποιεί τον SSH server. Με απλά λόγια, δεν υπάρχει ανεξάρτητος SFTP server αλλά υλοποιείται σαν επιπλέον συστατικό του πρωτοκόλλου SSH και ενεργοποιείται μέσω του SSH server. Ο SSH server ακούει στο port 22 οπότε οποιαδήποτε πληροφορία ανταλλάζεται μεταξύ του sftp (client) και του sftp-server περνά από τη θύρα 22 του SSH server.

Το SFTP παρέχει δύο μεθόδους για την πιστοποίηση ταυτότητας των συμβαλλομένων μερών (client, server). Η πρώτη μέθοδος ομοιάζει με το FTP, όπου ο χρήστης (sftp client) πιστοποιεί την ταυτότητά του απλά μέσω username και password. Ωστόσο, στο SFTP αυτές οι 2 σημαντικές πληροφορίες του χρήστη είναι κρυπτογραφημένες. Η δεύτερη μέθοδος πιστοποίησης ταυτότητας είναι μέσω των SSH κλειδιών (keys). Πριν την εγκατάσταση σύνδεσης, τόσο ο client όσο και ο server πρέπει να δημιουργήσουν από ένα ζεύγος κλειδιών, το δημόσιο (public) και το ιδιωτικό (private) κλειδί. Τα δημόσια κλειδιά ανταλλάζονται κατά τη διάρκεια της έναρξης της επικοινωνίας (handshake phase, φάση χειραψίας) μεταξύ client και server και μέσω αυτών των κλειδιών γίνεται τόσο η πιστοποίηση ταυτότητας όσο και η κρυπτογράφηση και αποκρυπτογράφηση των δεδομένων. Μετά την εγκατάσταση του ασφαλούς καναλιού επικοινωνίας, αποστέλνονται αιτήσεις (εντολές) από το client στο server οι οποίες έχουν διαφορετική μορφή από αυτές που περιγράψαμε πιο πάνω (ενότητα II.2) για το πρωτόκολλο FTP. Κάθε αίτηση (και απόκριση) έχει ένα ξεχωριστό τύπο που περιγράφει το είδος της αίτησης όπως για παράδειγμα, SSH_FXP_INIT, SSH_FXP_OPEN, SSH_FXP_CLOSE, SSH_FXP_READ, SSH_FXP_WRITE SSH_FXP_STAT, SSH_FXP_LSTAT, SSH_FXP_FSTAT, SSH_FXP_OPENDIR, SSH_FXP_READDIR, SSH_FXP_MKDIR, SSH_FXP_RMDIR, SSH_FXP_DATA, SSH_FXP_RENAME οι οποίες περιγράφονται στο προαναφερθέν Internet Draft (ενότητα 3). Η αίτηση με τύπο SSH_FXP_INT αρχικοποιεί τη σύνδεση. Τα αρχεία σε ένα κατάλογο μπορούν να παρουσιαστούν (list) χρησιμοποιώντας τις αιτήσεις SSH_FXP_OPENDIR και SSH_FXP_READDIR. Η αίτηση SSH_FXP_OPENDIR ανοίγει ένα κατάλογο για διάβασμα. Κάθε αίτηση SSH_FXP_READDIR επιστρέφει ένα ή περισσότερα ονόματα αρχείων μαζί με τα χαρακτηριστικά τους (attributes). Τα

* Έγγραφο από την Internet Engineering Task Force (IETF) που περιέχει προκαταρκτικές τεχνικές προδιαγραφές, αποτελέσματα έρευνας σχετικής με τη δικτύωση ή άλλες τεχνικές πληροφορίες.

χαρακτηριστικά των αρχείων μπορούν να επιστραφούν και με τις αιτήσεις SSH_FXP_STAT (follows symbolic links), SSH_FXP_LSTAT (does not follow symbolic links), SSH_FXP_FSTAT (status information for an open file). Όπως βλέπετε τα ονόματα κάποιων αιτήσεων μοιάζουν με αντίστοιχα systems calls της γλώσσας C.

Το FTPS προδιαγράφεται στο RFC 2228 (FTP Security Extensions) και πιο εκτεταμένα στο RFC 4217 (Securing FTP with TLS). Στα 2 αυτά RFCs προδιαγράφεται η λειτουργία Explicit FTPS σύμφωνα με την οποία ο FTP client δημιουργεί μια μη ασφαλή (non secure) σύνδεση ελέγχου (συνήθως στη θύρα 21, όπως ακριβώς κάνει και το FTP) και μετά εκκινεί τη διαδικασία αναβάθμισης της μη ασφαλούς σύνδεσης ελέγχου σε ασφαλή σύνδεση ελέγχου μέσω επιπλέον εντολών όπως AUTH, PROT, CCC κτλ. Εκτός από τη λειτουργία Explicit FTPS υπάρχει και η λειτουργία Implicit FTPS η οποία δεν προδιαγράφεται σε κάποιο RFC. Σύμφωνα με τη λειτουργία αυτή, η εγκατάσταση ασφαλούς σύνδεσης ελέγχου γίνεται από την αρχή. Δηλαδή όποια σύνδεση γίνεται από τον FTPS client στο FTPS server (είτε σύνδεση ελέγχου ή σύνδεση δεδομένων) πρέπει να γίνεται εξαρχής πάνω από τα πρωτόκολλα SSL/TLS. Με απλά λόγια στην Implicit FTPS λειτουργία ο server δεν δέχεται μη ασφαλείς συνδέσεις. Στην άσκηση αυτή θα υλοποιήσουμε τη λειτουργία Implicit FTPS.

Το (Explicit & Implicit) FTPS πιστοποιεί την ταυτότητα ενός χρήστη μέσω username και password ή μέσω πιστοποιητικού ή και με τα δύο. Αυτή η διαδικασία διεκπεραιώνεται μέσω των πρωτοκόλλων TLS/SSL. Όταν επιχειρείται ασφαλής σύνδεση σε ένα FTPS server, ο FTPS client θα ελέγξει πρώτα αν το πιστοποιητικό του server είναι αξιόπιστο (trusted). Το πιστοποιητικό θεωρείται αξιόπιστο εάν το πιστοποιητικό υπογράφηκε από μια γνωστή αρχή έκδοσης πιστοποιητικών (CA) ή εάν το πιστοποιητικό υπογράφηκε (ψηφιακά) με το ιδιωτικό κλειδί του ίδιου του server (self-signed certificated) και εγκατασταθεί με την έγκριση του χρήστη στην «αποθήκη» αξιόπιστων πιστοποιητικών (Trusted Root Certification Authorities store) του client. Ο server μπορεί επίσης να απαιτήσει από τον client να προσκομίσει το πιστοποιητικό του όταν επιχειρεί να συνδεθεί σε αυτόν. Εάν το πιστοποιητικό του client δεν έχει υπογραφεί από μια γνωστή αρχή έκδοσης πιστοποιητικών, ο server μπορεί να απαιτήσει να του αποσταλεί το πιστοποιητικού ψηφιακά υπογεγραμμένο με το ιδιωτικό κλειδί του client για να το φορτώσει στο αποθηκευτικό χώρο των αξιόπιστων πιστοποιητικών του.

Υπάρχει διαθέσιμο το δωρεάν εργαλείο OpenSSL το οποίο υλοποιεί το πρωτόκολλο TLS/SSL και θα χρησιμοποιηθεί για να δημιουργήσει τα ασφαλή κανάλια (συνδέσεις ελέγχου και συνδέσεις δεδομένων) πάνω από τα οποία θα υλοποιήσετε το πρωτόκολλο FTP. Για περισσότερες λεπτομέρειες διαβάστε την επόμενη ενότητα.

III. Περιγραφή Ζητούμενων Εργασίας

Αντικείμενο της άσκησης είναι να αναπτύξετε ένα ασφαλή «πολυδιεργασιακό» ή «πολυνηματικό» εξυπηρετητή αρχείων, **FTPS server**. Πιο συγκεκριμένα θα υλοποιήσετε το πρωτόκολλο FTP πάνω από το πρωτόκολλο SSL/TLS (Implicit FTPS). Δεν θα υλοποιήσετε το SSL/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 FTP. Η λειτουργία του πρωτοκόλλου FTP περιγράφεται στο τεχνικό άρθρο Request For Comments 959: http://tools.ietf.org/html/rfc959. Προτού περιγράψουμε τις εντολές του FTP που θα

υλοποιήσετε, θα παρουσιάσουμε κάποια απλά παραδείγματα χρήσης του εργαλείου OpenSSL.

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

```
#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;

    /* 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 and TLSv1.2.
     */
    method = SSLv23_server_method();
    // deprecated, new function is TLS_server_method()

    /* 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;
    SSL_CTX *ctx;

    /* 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) {
        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
             */
            SSL_write(ssl, reply, strlen(reply));
        }

        /* 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.

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
```

Για να δείτε ότι τρέχει το πρόγραμμα, χρειάζεστε ένα πρόγραμμα πελάτη (client). Αυτό μπορεί να γίνει με 2 τρόπους. Ο πρώτος τρόπος είναι με το εργαλείο openssl από τη γραμμή εντολών:

```
openssl s_client -connect b103ws12.in.cs.ucy.ac.cy:4433
```

(προσέξετε να δώσετε το σωστό όνομα μηχανής πάνω στο οποίο τρέχει ο εξυπηρετητής και τη σωστή θύρα). Μετά την εκτέλεση της εντολής θα δείτε στην οθόνη τη διαδικασία ανταλλαγής πιστοποιητικών (SSL handshake and negotiation phase) και στο τέλος το μήνυμα test που στάλθηκε από τον εξυπηρετητή.

Ο δεύτερος τρόπος είναι γράφοντας ένα μικρό πρόγραμμα σε γλώσσα C (δείτε πρόγραμμα tls_client.c στο as4-supplementary.zip).

Στη δική σας άσκηση θα τροποποιήσετε τον πιο πάνω κώδικα (tls_server.c) για να δημιουργήσετε ένα ασφαλή, πολύ-διεργασιακό ή πολύ-νηματικό FTPS server. Ο FTPS server θα πρέπει να παρέχει ασφάλεια (μέσω TLS/SSL) και στη σύνδεση ελέγχου και στη σύνδεση δεδομένων. Για την πολύ-διεργασιακή ή πολύ-νηματική λειτουργία δείτε την ενότητα V.

Για να δοκιμάσετε τον FTP server σας θα χρειαστείτε ένα FTPS client για να ανεβάζετε αρχεία στο server ή να κατεβάζετε αρχεία από το server. Μπορείτε να γράψετε το δικό σας client σε γλώσσα C (στη βάση του tls_client) ή να δοκιμάσετε αν κάποιος εμπορικός FTPS client (π.χ. Filezilla) μπορεί να συνδεθεί με τον server σας. Στην περίπτωση που ο server σας θα μπορεί να εξυπηρετήσει εμπορικό FTPS client, θα έχετε επιπλέον BONUS στη βαθμολογία σας. Επίσης θα πρέπει να δημιουργήσετε ένα κατάλογο με όνομα ftphome/ μέσα στον ίδιο κατάλογο που είναι το .c πρόγραμμά σας ο οποίος θα περιέχει τα αρχεία που μπορεί να «σερβίρει» ο FTP server και στον οποίο επίσης να μπορεί να τοποθετεί τα αρχεία που θα ανεβάζουν οι clients (το όνομα του καταλόγου αυτού μπορείτε να το βρίσκετε και μέσα στο config file – δες ενότητα V).

Για τις ανάγκες της άσκησης, πρέπει να μπορείτε να χειριστείτε τις εντολές **USER**, **PASS CWD, CDUP** και **QUIT** [RFC959/§4.1.1], την εντολή **PASV** [RFC959/§4.1.2] καθώς και τις εντολές **RETR, STOR, LIST, MKD**, και **STAT** [RFC959/§4.1.3]. Αν δοθεί κάποια από αυτές τις εντολές ο εξυπηρετητής πρέπει να απαντά με κάποιο θετικό κωδικό. Αν δοθεί κάποια διαφορετική εντολή από τον πελάτη, τότε ο εξυπηρετητής πρέπει να απαντά με κάποιο αρνητικό κωδικό. Οι κωδικοί αυτοί θα αναφερθούν πιο κάτω αλλά περιγράφονται πιο αναλυτικά στο RFC959/§4.2.1 και §4.2.2.

Αρχικά, ο εξυπηρετητής FTP ξεκινά δημιουργώντας ένα TCP socket στη θύρα 21. Όταν πελάτης FTP (π.χ. WinSCP, Filezilla[†], κτλ.) επιθυμεί να στείλει ή να λάβει ένα αρχείο, δημιουργεί μια σύνδεση TCP με τον εξυπηρετητή FTP. Όταν θα εγκαθιδρυθεί η σύνδεση στη θύρα 21, ο εξυπηρετητής FTP στέλνει ένα χαιρετισμό. Αυτή η σύνδεση ονομάζεται σύνδεση ελέγχου. Στη συνέχεια, ο πελάτης και ο εξυπηρετητής FTP ανταλλάζουν εντολές και απαντήσεις αντίστοιχα μέσω της σύνδεσης ελέγχου. Αν ο πελάτης αποφασίσει να λάβει ή να στείλει ένα αρχείο από και προς τον εξυπηρετητή, αιτείται τη δημιουργία μιας σύνδεσης δεδομένων. **Στην άσκηση αυτή, καλείστε να υλοποιήσετε την παθητική λειτουργία όταν θα δημιουργείται μια ασφαλισμένη σύνδεση δεδομένων**.

Η κάθε σύνδεση δεδομένων χρησιμοποιείται για τη μεταφορά ενός αρχείου και μετά καταργείται. Η επικοινωνία πελάτη-εξυπηρετητή τερματίζεται με την κατάργηση της σύνδεσης ελέγχου, από την πλευρά του πελάτη.
Οι εντολές FTP αποτελούνται από μια case-insensitive λέξη-κλειδί (command), πιθανώς ακολουθούμενη από κανένα ή ένα όρισμα. Όλες οι εντολές στέλνονται μέσα από τη σύνδεση ελέγχου. 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.

Οι λέξεις-κλειδιά μπορούν να χωριστούν σε: ελέγχου πρόσβασης, προσδιορισμού παραμέτρων ή αιτήσεις υπηρεσίας FTP, όπως δόθηκαν στην ενότητα ΙΙ. **Μερικές λέξεις-κλειδιά (όπως STAT, QUIT) μπορούν να σταλούν μέσα από τη σύνδεση ελέγχου ενόσω μεταφορά δεδομένων βρίσκεται σε εξέλιξη.**
Οι απαντήσεις στο πρωτόκολλο FTP αποστέλλονται (μέσω της σύνδεσης ελέγχου) για να διασφαλιστεί ο συγχρονισμός των αιτήσεων του πελάτη FTP και των ενεργειών του εξυπηρετητή FTP κατά τη διάρκεια της μετάδοσης ενός αρχείου, και για να υπάρχει εγγύηση ότι ο πελάτης γνωρίζει πάντα την κατάσταση του εξυπηρετητή. Κάθε εντολή πρέπει να δημιουργήσει τουλάχιστον μια απάντηση, αν και μπορεί να υπάρχουν περισσότερες από μία. Στην τελευταία αυτή περίπτωση, οι πολλαπλές απαντήσεις, πρέπει να διακρίνονται εύκολα. Επιπλέον, κάποιες εντολές εμφανίζονται σε διαδοχικές ομάδες, όπως οι εντολές USER και PASS. Μια αποτυχία σε οποιοδήποτε σημείο της ακολουθίας απαιτεί την επανάληψη ολόκληρης της ακολουθίας από την αρχή. Μια απάντηση FTP αποτελείται από έναν τριψήφιο αριθμό που ακολουθείται από κάποιο κείμενο. Μεταξύ του τριψήφιου αριθμού και του κειμένου μεσολαβεί SPACE ενώ η απάντηση τερματίζεται από το CRLF.

Η υλοποίησή σας θα πρέπει να επιστρέφει **τουλάχιστο** τους ακόλουθους κωδικούς απάντησης (ανάλογα με την εντολή του πελάτη):

150 File status okay; about to open data connection. [Ακολουθεί την εντολή RETR]

[†] Για να επιτευχθεί επικοινωνία με τον Filezilla, (α) πρέπει να εγκαταστήσετε το πρόγραμμα αυτό (είναι δωρεάν και υπάρχει και στις μηχανές των εργαστηρίων B103 και 103), (β) κατά τη δημιουργία μιας σύνδεσης στο File -> Site Manager -> New Site, στο πεδίο Host να βάλετε το IP της μηχανής σας, στο πεδίο Port τη θύρα, στο πεδίο Protocol να βάλετε FTP, στο πεδίο Encryption να βάλετε "Require implicit* FTP over TLS" και στο πεδίο Logon Type να βάλετε Normal και να δώσετε μετά Username και password, και (γ) να υλοποιήσετε την εντολή TYPE (RFC959, §4.1.2). Οποιαδήποτε άλλη εντολή στείλει το Filezilla μπορείτε να την "απορρίπτεται' απαντώντας 502 Command not implemented. Δείτε πιο κάτω τις απαντήσεις που μπορεί να στέλνει ο FTPS server σας.

200 Command okay. [Ακολουθεί σωστή εκτέλεση μιας εντολής όπως CWD, CDUP, LIST]
220 Service ready for new user. [Επιστρέφεται όταν ο πελάτης συνδεθεί επιτυχώς με τον εξυπηρετητή]
221 Service closing control connection. Logged out if appropriate.
226 Closing data connection. Requested file action successful. [Ακολουθεί και επιβεβαιώνει τη σωστή αποστολή ή λήψη αρχείου από το data connection]
227 Entering Passive Mode (h1,h2,h3,h4,p1,p2). [Ακολουθεί την εντολή PASV]
230 User logged in, proceed. [Ακολουθεί την εντολή PASS όταν αυτή είναι επιτυχής]
250 Requested file action okay, completed. [Ακολουθεί σωστή εκτέλεση της εντολής STOR]
257 "PATHNAME" created. [Ακολουθεί την εντολή MKD]
331 User name okay, need password. [Ακολουθεί την εντολή USER όταν αυτή είναι επιτυχής]
332 Need account for login. [Όταν το USER σταλεί χωρίς username]
426 Connection closed; transfer aborted. [Όταν διακόπτεται η μετάδοση δεδομένων απότομα]
451 Requested action aborted: local error in processing. [Μήνυμα λάθους για πολλές εντολές όπως CWD, CDUP, MKD, STOR, PASV, LIST, STAT κ.α.]
500 Syntax error, command unrecognized. [Όταν ή εντολή είναι λεκτικά ορθή αλλά συντακτικά λάθος π.χ. filename STOR αντί ανάποδα]
502 Command not implemented. [Όταν η εντολή που δίδεται δεν έχει υλοποιηθεί π.χ. STAR αντί STOR]
503 Bad sequence of commands. [Όταν δοθεί λανθασμένη εντολή που δεν αναμένεται π.χ. μετά το USER να δοθεί STOR αντί PASS]
530 Not logged in. [Όταν δίνονται εντολές πριν δοθεί το USER]
550 Requested action not taken. File unavailable. [Όταν δεν υπάρχει το ζητούμενο αρχείο π.χ. μετά την εντολή RETR]

IV. Παράδειγμα Συνόδου FTP

Στο πιο κάτω παράδειγμα, ο πελάτης U θέλει να μεταφέρει αρχεία από/προς τον εξυπηρετητή S. Ένα τυπικό παράδειγμα τέτοιου σεναρίου (μετά την εγκατάσταση ασφαλούς καναλιού επικοινωνίας) φαίνεται πιο κάτω. Οι εντολές που στέλνονται από τον πελάτη δίδονται με το '---->' ενώ οι απαντήσεις του εξυπηρετητή με το '<---'.

```
Connect to host S, port L, establishing control connection.
<---- 220 Service ready for new user<CRLF>.
USER Doe<CRLF>---->
<---- 331 User name ok, need password <CRLF>.
PASS mumble<CRLF>---->
<---- 230 User logged in, proceed<CRLF>.

Client-FTP initiates PASV operation.
PASV<CRLF>---->
<---- 227 Entering Passive Mode (208,75,230,189,144,129)<CRLF>.
Connect to host S, port L2, establishing data connection.
Client-FTP opens local file in ASCII.
RETR test.pl1<CRLF> ---->
<---- 150 Accepted data connection<CRLF>.
Server sends data through data connection.
<---- 226 File status okay; about to open data connection<CRLF>.
Server closes data connection.

Connect to host S, port L2, establishing data connection.
Client-FTP opens local file in ASCII.
STOR test2.txt<CRLF> ---->
```

```
<---- 550 Requested action not taken. File unavailable<CRLF>
QUIT <CRLF> ---->
Server closes all connections.
```

V. Παραλληλη (πολύ-διεργασιακή ή πολύ-νηματική) λειτουργία εξυπηρετητή

Ο FTPS 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 FTPS server. Μια ιδέα θα ήταν, όταν παίρνει μια αίτηση από πελάτη, να δημιουργεί μια διεργασία ή νήμα για να την εξυπηρετήσει, ενώ η αρχική διεργασία ή νήμα να περιμένει νέες αιτήσεις. Η εξυπηρέτηση των αιτήσεων αυτών θα γίνεται από νέες διεργασίες ή νήματα που θα δημιουργεί η αρχική διεργασία ή νήμα. 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 ή διεργασίων 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 ή process-pool, δηλαδή να δημιουργήσει εξ αρχής ένα σταθερό αριθμό νημάτων ή διεργασιών εργατών (που το πλήθος τους να δίνεται) και όταν υπάρχει αίτηση για εξυπηρέτηση να την αναθέτει σε κάποιο από τα νήματα ή διεργασία αυτά που δεν έχει δουλειά. Οι διεργασίες ή τα νήματα, αφού εξυπηρετήσουν ένα πελάτη, δεν τερματίζουν, αλλά μεταβαίνουν σε κατάσταση αναμονής. Φυσικά, αν δεν υπάρχει διαθέσιμο νήμα ή 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 FTPS 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 (παράδειγμα για νήματα):

```
# FTPS server Configuration File

# The Number of Threads in the Threadpool
THREADS=40

# The Port number of the FTPS server
PORT=30000

# The HOME folder of the FTP server
HOME=./ftphome
…
```

VI. 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.

VII. Αξιολόγηση
Α) Τι πρέπει να παραδώσετε;

- **Στο Moodle:** Ενα αρχείο **ftps.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 (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 RFC959. 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 (RFC959) καθώς επίσης να ανακαλύψετε νέες συναρτήσεις πέρα από αυτές που διδαχθήκατε ήδη στις διαλέξεις.

Καλή Επιτυχία !