TEL AVIV UNIVERSITY אוניברסיטת תל-אביב

רשתות תקשורת מחשבים 03683030
ביה"ס למדעי המחשב, אוניברסיטת ת"א
סמסטר א 2016/17

תרגיל בית מעשי מס' 1

להגשה עד ה – 28.11.2016

שימו לב: באתר הקורס תוכלו למצוא מסמך עם הנחיות מפורטות לגבי הגשת תרגילים מעשיים. על כל התרגילים המעשיים שתגישו הסמסטר לעמוד בהנחיות הללו .

מטרה:

- מימוש של פרוטוקול.
- שימוש נכון בsocket programming.

בתרגיל זה נממש שרת דוא"ל פשוט ונכתוב לקוח עבורו. השרת והלקוח ישוחחו בפרוטוקול אפליקציה אשר תגדירו בעצמכם .

בתרגיל המעשי הבא, נרחיב את האפליקציה והפרוטוקול ונוסיף להם יכולות.

השרת:

השרת ישמור רשימת לקוחות, וישמור את המיילים הממתינים לכל לקוח.

אתחול השרת מתבצע בשורת הפקודה

mail_server users_file [*port*]

כש – users_file הוא ה – path לקובץ טקסטואלי tab-delimited שמכיל שתי עמודות: שם משתמש והסיסמה המוגדרת למשתמש. למשל:

Moshe	1234
Yossi	password123
Esther	abcde

(בין שני השדות יש tab).

הפורט להקשבה הוא פרמטר אופציונלי עם ערך ברירת מחדל 6423.

לאחר מכן השרת ממתין ומשרת לקוחות. לשם פשטות, מרגע שהשרת מתחיל לרוץ הוא אינו מסיים את ריצתו. בכל רגע נתון הוא נדרש לטפל רק בלקוח אחד (ולכן מותר לכם להשתמש בפעולות חוסמות). מצד שני, הוא צריך להיות מוכן לשרת מספר רב של לקוחות לאורך חייו (כל עוד הם לא מנסים להתחבר באותו הזמן).

לשם פשטות, אתם יכולים להניח שהשרת נדרש לשמור מספר קטן יחסית של מיילים, ולכן אתם יכולים להחזיק הכל בזיכרון ולא נדרשים לשמור מידע באופן persistent (על הדיסק למשל).

כשלקוח חדש מתחבר לשרת השרת שולח לו ברכה כלשהי, שהיא שורת טקסט לבחירתכם. למשל:

Welcome! I am simple-mail-server.

לאחר מכן הוא ממתין ללקוח שיכניס שם משתמש וסיסמה. אם הם נכונים, השרת מקבל את הלקוח וממתין לפקודות.

הלקוח:

אתחול הלקוח מתבצע בשורת הפקודה:

mail_client [*hostname* [*port*]]

כאשר hostname ו-port הם פרמטרים אופציונאליים. ערך ברירת המחדל הוא:

- hostname = localhost
- port = 6423

שימו לב - לא ניתן לספק port ללא hostname. הפרמטר hostname יכול להיות שם או כתובת IP.

אחרי שהלקוח מקבל את הודעת הברכה של השרת הוא קולט מהמשתמש שתי שורות קלט:
שם משתמש וסיסמה אותן הוא שולח לשרת .

הן מוכנסות בפורמט הבא:

User: Moshe

Password: 1234

בהנחה שהשרת אישר את המשתמש ,יודפס למשתמש:

Connected to server

המשתמש יכול להכניס את הפקודות הבאות:

.1 SHOW_INBOX

מציגה את המיילים הממתינים למשתמש: כל מייל הוא שורה בפורמט הבא:

mail_id sender "subject"

כלומר: מספר סידורי של המייל (שישמש את המשתמש אח"כ כדי לבקש לקרוא את המייל הזה וכו'), שם השולח, והנושא. הנושא יהיה מוקף בגרשיים.

למשל:

1 Yossi "Hi there!"
2 Yossi "Funny pictures"
3 Moshe "reports of the last year"

לשם פשטות ,המספר הסידורי של המייל הראשון שהמשתמש יקבל הוא 1, של המייל השני הוא 2 וכן הלאה .
המספר הזה לא משתנה אף פעם מרגע שנקבע - למשל, אם הלקוח קיבל מיילים שמספרם 1,2,3 ומבקש למחוק את מייל 2, אזי המיילים שישארו יהיו עדיין מספר 1,3 – אין צורך לשנות את המספרים שלהם ל - 1,2.
באופן זה המספר הסידורי מזהה את המייל באופן חח"ע בכל ההתחברויות של הלקוח לשרת. תוכלו להניח שמספר המיילים שמשתמש יכול לקבל לכל אורך חיי השרת מוגבל ל – 32000.

.2 GET_MAIL mail_id

מציגה את שם השולח, שמות הנמענים, הנושא, והטקסט של המייל שמספרו mail_id. אם יש יותר מנמען אחד הם מופרדים בפסיקים .למשל:

From: Yossi
To: Moshe,Esther
Subject: Funny pictures
Text: How are you? Long time no see!

תוכלו להניח שכל השדות מכילים מידע.

.3 DELETE_MAIL mail_id

מוחקת את המייל שמספרו mail_id מהשרת.

.4 QUIT

מסיים את ריצתה של תוכנת הלקוח.

.5 COMPOSE

שולחת מייל חדש. אחרי פקודה זו המשתמש יכניס את פרטי המייל באופן הבא (ובדיוק בסדר הזה): נמען/ים, נושא, וגוף המייל.

אם יש יותר מנמען אחד הם מופרדים בפסיקים. למשל:

To: Moshe,Esther
Subject: Funny pictures
Text: How are you? Long time no see!

לשם פשטות, תוכלו להניח שגוף המייל מורכב משורה אחת (בלי enters באמצע המייל).

כל השדות חייבים להכיל מידע.

אחרי שהמייל נשלח בהצלחה (כלומר, כל המידע הגיע לשרת כשורה) תוכנת הלקוח מדפיסה למשתמש Mail sent.

דוגמת הרצה:

השרת:

mail_server ~/users.txt

בצד הלקוח (בפונט **ירוק** – קלט מהמשתמש):

```
mail_client
Welcome! I am simple-mail-server.
User: Yossi
Password: password123
Connected to server
COMPOSE
To: Moshe
Subject: Hi there!
Text: I have just installed the new mail client!
Mail sent
COMPOSE
To: Moshe,Esther
Subject: Funny pictures
Text: How are you? Long time no see!
Mail sent
SHOW_INBOX
QUIT
```

(שימו לב שלמשתמש לא היו מיילים ממתינים בתיבת הדואר ולכן לא הודפס דבר אחרי SHOW_INBOX).

כעת מתחבר לקוח שני:

```
mail_client
Welcome! I am simple-mail-server.
User: Moshe
Password: 1234
Connected to server
SHOW_INBOX
1 Yossi "Hi there!" 0
2 Yossi "Funny pictures" 2
GET_MAIL 1
From: Yossi
```

To: Moshe
Subject: Hi there!
Text: I have just installed the new mail client!
DELETE_MAIL 1
Attachment saved
QUIT

בונוס (5 נק'):

בפרוטוקול TCP ישנה בעיה של שליחת ההודעה האחרונה בתקשורת בין שני המחשבים.

את הבעיה והפתרון שלה ניתן לראות במפורט במאמר הבא:

http://blog.netherlabs.nl/articles/2009/01/18/the-ultimate-so_linger-page-or-why-is-my-tcp-not-reliable

אנא מימשו את פתרון לבעיה הנתונה (על ידי שימוש בshutdown, ובפונקציה recv לאחר מכאן).

בפתרון עליכם להבין מי משתמש בshutdown (השרת או הלקוח), ומדוע.

דרישות התרגיל

ראשית, עליכם לתכנן פרוטוקול אפליקציה מתאים שיעבוד מעל TCP. לאחר מכן, ממשו אותו כפי שנלמד בתרגול. האפליקציה סינכרונית, כלומר אפשר להשתמש בפקודות חוסמות.

את פרוטוקול האפליקציה יש לתעד בבירור, באופן שיאפשר לכל אדם לממש לקוח או שרת ש"ידברו" עם התוכנות שהגשתם.

הדגש בבדיקת הקוד שתגישו ינתן כמובן למימוש התקשורת ברשת. על המימוש להיות יעיל ורובוסטי (robust). אל תשכחו לבדוק שגיאות בערכי חזרה מפונקציות ולטפל בהם בהתאם.

למקרה שתבחרו להגיש בסביבת nova: ייתכן שזוגות רבים יבדקו את הפתרון שלהם על nova בו-זמנית . לכן, מומלץ להשתמש בפורט שרת שאינו פורט ברירת המחדל בעת בדיקת הקוד.

בהצלחה ☺