תרגיל מעשי 2 – רשתות תקשורת מחשבים

סמסטר חורף **2017**, אוניברסיטת תל אביב

מאיה כהנא 205973225

שון מולגה-נגר 204908859

ערי כבדיאל 303064463

תיאור הפרוטוקול

Greeting – לאחר התחברות הלקוח לשרת, השרת שולח לו הודעת כניסה שבתוכן שלה נמצאת מחרוזת עם ברכת כניסה למשתמש+שם המשתמש שלו. מודול הclient מקבל את ההודעה ומדפיס אותה למסך של הלקוח – לאחר מכן ממתין להמשך קליטת הודעות מהמשתמש.

list_of_files – פקודה שהלקוח יכול לשלוח לשרת, ולקבל בחזרה את שמות הקבצים השמורים עבורו במערכת. הלקוח שולח הודעת "list_of_files" לשרת. השרת נכנס לתיקיית הקבצים של הלקוח, בונה מחרוזת המורכבת מכל הקבצים הנמצאים שם ושולח מחרוזת זו להדפסה בצד הלקוח.

delete_file – פקודה שהלקוח שולח לשרת על מנת למחוק קובץ מסויים מתיקיית הקבצים שלו.

Add_file – הלקוח שולח לשרת את המחרוזת "add_file" ולאחר מכן path לקובץ. הלקוח קורא את תוכן הקובץ ומעביר את התוכן באמצעות buffer למשרת. השרת מצידו מעביר את תוכן הbuffer לקובץ ושומר אותו בתיקיה של הלקוח.

Get_file – פקודה שהלקוח שולח לשרת על מנת לשמור קובץ מסוים ששמור על השרת במחשב של הלקוח. הלוקח שולחת לשרת את המחרוזת "get_file" ולאחר מכן את שם הקובץ ואת הpath לקובץ על מחשב הלקוח שבו נרצה לשמור את הקובץ. השרת מעתיק את תוכן הקובץ המבוקש לתוך buffer ומעביר אותו לצד הלקוח, ששומר אותו בpath.

Quit – הלקוח שולח פקודה זו וצד הלקוח מתנתק מצד השרת באמצעות פקודת close.

Read_msgs – כאשר הלקוח רוצה לקרוא את ההודעות שנשלחו אליו בזמן שלא היה מחובר לשרת, הוא קורא לפקודה זו וההודעות מודפסות לו על המסך.

Msg <x>: <message_content> – פקודה שהלקוח יכול להשתמש בה כדי לשלוח ללקוח אחר הודעה. אם הלקוח השני מחובר, ההודעה תוצג לו על המסך ואם אינו מחובר, ההודעה תיכתב לקובץ האופליין.

Online_users: פקודה שמחזירה ללקוח את שמות המשתמשים שבאותו רגע מחוברים לשרת

מבנה התוכנית

file_client.c – ניתוח ארגומנטים ראשוניים מהלקוח – hostname וport דרכם ירצה להתחבר.

file_server.c – עלייתו הראשונית של השרת. קבלת קובץ המשתמשים, יצירת תיקיות רלוונטיות והעלאת השירות.

client_protocol – לוגיקת הפרוטוקול מצד הלקוח

server_protocol – לוגיקת הפרוטוקול צד שרת

Network – מודול המחבר בין הלקוח לשרת שמכיל פונקציות שליחה וקבלה של הודעות מצד אחד לצד שני

תיאור צורת התקשורת בין הלקוח לבין השרת

השרת והלקוח מתקשרים זה עם זה באמצעות מבנה קשיח ייעודי בשם Message.

משתנה מסוג header שיודע את אורך ההודעה (אורך הstring) שאנו שולחים לשרת, ואת סוגו (enum).

משתנה מסוג *char שמכיל את המחרוזת שאנחנו רוצים להעביר לשרת/ללקוח
ישנם מספר סוגים של הודעות שיכולות להישלח בין השרת לבין הלקוח:
LOGIN_DETAILS – הודעה שמטרתה לסייע בתהליך ההזדהות של הלקוח אל מול השרת.
LIST_OF_FILES – הודעה שמטרתה לסייע בתהליך ביצוע הפקודה list_of_files.
FILE_DELETE – הודעה שמטרתה לסייע בתהליך ביצוע הפקודה delete_files.
ADD_FILE – הודעה שמטרתה לסייע בתהליך ביצוע הפקודה add_file.
GET_FILE – הודעה שמטרתה לסייע בתהליך ביצוע הפקודה get_file.
QUIT – הודעה שמטרתה לסיים את החיבור בין השרת לבין הלקוח.
ERROR – הודעה שמטרתה ליידע את צד הלקוח שקרתה תקלה בצד השרת.
GREETING – הודעת ברכת שלום מהשרת ללקוח בהתחברות ראשונית.
FILE_CONTENT – הודעה שמכילה תוכן של קובץ על אחד מהצדדים (צד לקוח או צד שרת).
INVALID_LINE – הודעה שלא תואמת אף פורמט מוכר ע"י הפרוטוקול.
LIST_OF_ONLINE_USERS – הודעה שמטרתה לסייע בתהליך ביצוע הפקודה online_users
READ_MSGS – הודעה שמטרתה לסייע בתהליך ביצוע הפקודה read_msgs
MSG – הודעה שמטרתה לסייע בתהליך השליחה של הודעה מלקוח אחד לשני

תיאור הפרוטוקול ואופן זרימת המידע:

ראשית, השרת שולח הודעת ברכה לשרת – "Welcome! Please log in."
בתגובה, הלקוח שולח לשרת הודעה מסוג LOGIN_DETAILS את שם המשתמש ואת הסמא שלו בפורמט המבוקש: "User: username", "Password: password" כאשר בסוף ההקלה הוא לוחץ על enter. במידה והשם המשתמש ו/או הסמא לא מוכרים לשרת, או במידה והפורמט שגוי, השרת שולח ללקוח הודעה מתאימה.
לאחר תהליך ההתחברות, השרת שולח הודעה שמאשרת כי הוא מודע להתחברות וקישר את הלקוח ליוזר קיים במערכת.
כעת היוזר יכול להמשיך ולשלוח כל פקודה מבין : list_of_files, msg:, read_msgs, online_users, add_file, get_file, או delete_file עד ששולח את פקודת quit אשר מנתקת אותו מהשרת.
בתרגיל זה הוספנו למימוש מערך של משתמשים מחוברים, כלומר מערך של 15 structs מסוג connection_t שמכיל שדה סוקט ושדה שם משתמש (בהנחה כי שמות המשתמשים הם unique). מערך זה מתחזק את רשימת המשתמשים המחוברים בכל רגע נתון אל השרת, כאשר יכולים להיות עד 15 משתמשים כאלו. כאשר משתמש רוצה להתחבר, הוא קורא לfile_client.c והבקשה נרשמת כחיבור במערך המשתמשים (רק של הסוקט). לאחר שמגיעים לאותו סוקט בלולאה הבאה (שעוברת על כל הFDים) מחכים או מקבלים את פרטי ההתחברות שלו, ואם הם נכונים רושמים גם את שם המשתמש שלו לצד הסוקט שלו(שכבר רשום) במערך המשתמשים המחוברים.
כאשר הלקוח רוצה להתנתק הוא שולח הודעת quit ונמחק מרשימת המשתמשים המחוברים.
בצד השרת, בלולאת ה while הראשית לאחר ביצוע selevt – עוברים כל פעם על הsocket listen על מנת לקלוט חיבורים חדשים לשרת על הread_fds על מנת לקלוט פקודות חדשות מהמשתמשים המחוברים.
לאחר ביצוע select, נבצע login למשתמשים חדשים שרוצים להתחבר ונעבור על מערך המשתמשים המחוברים ובאמצעות FD_ISSET נזהה את הלקוחות ששלחו אל השרת בקשות שירות. **השרת יטפל באותו לקוח עד לסיום הבקשה שלו** ורק לאחר מכן יעבור ללקוח הבא.
בצד הלקוח, בלולאת הwhile הראשית יתבצע כל פעם select בין הsocket של הלקוח שאיתו הוא מחובר לשרת לבין ה STDIN שלו – זאת על מנת לקבל גם הודעות מהשרת, גם אם הלקוח לא פנה אליו באופן אקטיבי (מתרחש רק עבור קבלת הודעות ממשתמשים אחרים) וגם הודעות מהלקוח עצמו, שמכניס פקודות לשרת דרך הSTDIN.

הסבר על מימוש פקודות הפרוטוקול:
List_of_files – הלקוח שולח לשרת Message מסוג LIST_OF_FILES. השרת מקבל את ההודעה, ומחזיר בתורו הודעה מסוג LIST_OF_FILES, כאשר השדה arg1 מכיל char* של כל שמות הקבצים ששמורים בתיקיה של המשתמש בצד השרת.
Add_File – הלקוח שולח לשרת 2 הודעות מסוג ADD_FILE. ההודעה הראשונה תכיל את הארוגמנט השני שהלקוח מקליד, שהוא שם הקובץ החדש שאנחנו רוצים ליצר בשרת. הלקוח יחלץ משורת הפקודה שמקליד המשתמש את הfile_path (הארגומנט הראשון אחרי "add_file"), ויקרא את תוכנו. ההודעה השניה שישלח הלקוח אל השרת יכיל את תוכן הקובץ שקרא. השרת בתורו יצור בתוך תיקית המשתמש את הקובץ המבוקש.
הנחות: אם המשתמש לא סיפק שם חדש שלקובץ בפקודה זו, יוחזר מהשרת INVALID_COMMAND.
Get_File – הלקוח שולח הודעה מסוג GET_FILE. ההודעה הראשונה מכיל את שם הקובץ אותו נרצה לקבל מהשרת. בתגובה הלקוח מקבל מהשרת הודעה מסוג GET_FILE שמכילה את תוכל הקובץ המבוקש. הלקוח מחלץ משורת הפקודה שמקליד המשתמש את file_path, וכותב לתוכו את תוכן הקובץ שקיבל בהודעה מהשרת. הקובץ בצד הלקוח יקרא באותו השם.
Delete_File – הלקוח שולח הודעה מסוג DELETE_FILE, שמכיל את שם הקובץ אותו יבקש מהשרת למחוק. השרת בתורו ימחק את הקובץ מהתיקייה הייעודית של המשתמש.
Quit – שרת הלקוח שולח הודעה מסוג QUIT. הלקוח מנתק את ההתקשרות עם השרת באמצעות הפקודה close(), והשרת ממשיך להאזין ע"מ לאתר לקוחות חדשים שירצו להתחבר אליו.
הנחות כלליות: כל פקודה מכילה מספר נכון של ארגומנטים, ובמידה ולא, לא תתבצע הפקודה.
Online_users- הלקוח שולח לשרת הודעה מסוג זה, השרת עובר על כל המשתמשים ובודק מי מחובר ומחזיר את רשימת השמות שלהם, שמודפסת ללקוח על המסך.
Read_msgs- השרת ניגש לקובץ Messages_received_offline.txt וקורא משם את ההודעות שנשלחו ללקוח בזמן שלא היה מחובר, מדפיס אותן ללקוח על המסך ומוחק את ההודעות מהקובץ.
Msg <x>: <message_content> - השרת בונה את ההודעה שנשלחת ומזהה את ה socketשל המשתמש שאליו ההודעה אמורה להישלח. אם הסוקט אינו מחובר, כלומר אינו נמצא ברשימת המשתמשים המחוברים (יחד עם השם משתמש) אזי ההודעה נכתבת לקובץ Messages_received_offline.txt שלו. אחרת, אם מחובר, שולחים את ההודעה (כמו הודעה רגילה מהשרת) למשתמש.

Macros
MAX_USERNAME_SIZE – מספר תווי שם המשתמש המקסימלי – 25
MAX_USERNAME_SIZE – מספר תווי הסמא המקסימלי – 25
MAX_CLIENT – מספר מקסימלי של משתמשים (יוזרים) – 15
MAX_FILES_PER_CLIENT – מספר מקסימלי של קבצים שניתן לשמור בתיקיית המשתמש בצד השרת – 15
MAX_FILE_SIZE – גודל קובץ מקסימלי – 512 בתי
MAX_PATH_NAME – מספר מקסימלי של תווים בשם הקובץ – 500
MAX_ARG_LEN – מספר מקסימלי של תווים של ארגומנט שמצורף לפקודה – 500
MAX_FILE_NAME – מספר מקסימלי של תווים של שם של קובץ – 50
MAX_COMMAND_NAME – מספר מקסימלי של תווים של שם של פקודת פרוטוקול – 15
HEADER_SIZE (sizeof(MessageHeader) – גודל של header מוגדר להיות הגודג של הMessageHeader.
MAX_PACKET_SIZ – גודל פקטה מקסימלית, כלומר הגודל הכולל של Message – 4096 בתים
MAX_DATA_SIZE (MAX_PACKET_SIZE - HEADER_SIZE) – גודל הדאטא המקסימלי מוגדר להיות גודל הפקטה המקסימלית פחות גודל הheader שלה.
MAX_MSG_CONTENT- גודל מקסימלי של הודעה בין משתמשים שונים – 100 (גם להודעות אופליין וגם לאונליין).