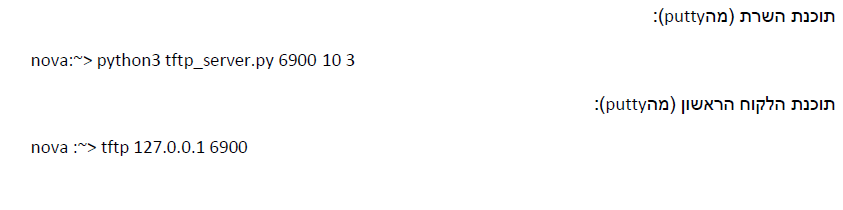
**תרגיל 2 – רשתות תקשורת מחשבים – Read Me**

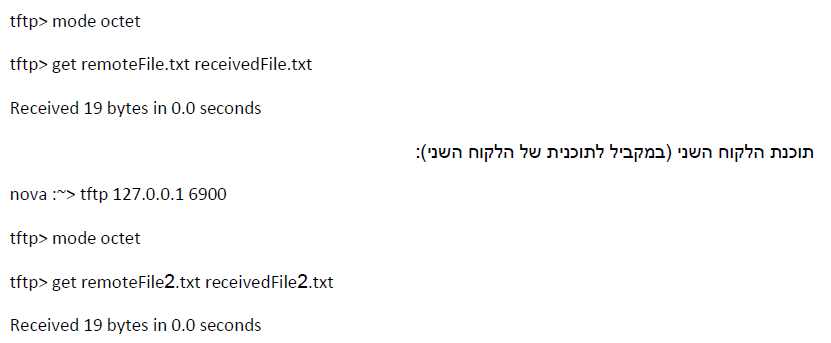
**אורין לוי 207543471, orinlevy@mail.tau.ac.il**

**חן מור יוסף 205580947, chenmoryosef@mail.tau.ac.il**

חלק 1 – תיאור קצר של התכנית וכיצד להריץ אותה

* מדובר בתכנית המממשת tftp\_server כמתואר בתיעוד ומשתמשת במתודת select לטובת תמיכה במספר לקוחות בו-זמנית. בדבר מקרי קצה נפרט בהמשך.
* צורת ההרצה של הקוד היא כמתואר בגוף התרגיל:





* מטה נפרט את ההחלטות העיצוביות שלקחנו בעת מימוש בו-הזמניות בעזרת select.

חלק 1 - מבנה התכנית

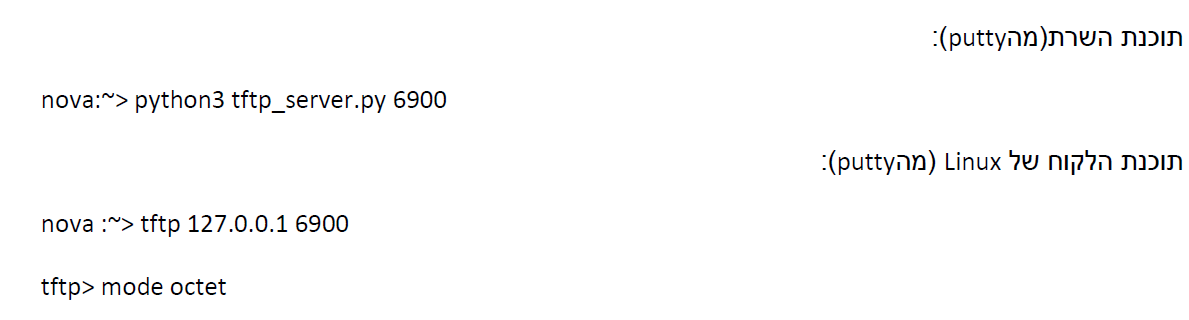
* התכנית מכילה קובץ יחיד כיוון שכך התבקשנו.
* התכנית מחולקת לפונקציות כך שפונק' הmain מקבלת פקטות מקליינטים שונים באמצעות הסוקט הראשי של הסרבר ובעת שקיבלה פקטה פותחת סוקט ייחודי לתקשורת מול הקלינט שממנו הגיעה הפקטה.
* בכדי לתמוך בבו-זמניות, תיחזקנו מספר מבני נתונים שסייעו לנו להבין מה הסטטוס של כל אחד מהסוקטים (האם עלינו לשלוח לו פקטה/לקבל ממנו פקטה). על כן, תחזקנו 2 רשימות, האחת מכילה את כל הסוקטים אליהם אנו צריכים לשלוח פקטה והשניה מכילה את כל הסוקטים מהם אנו צריכים לקבל פקטה. במקביל, תחזקנו מילון שמפתחיו הינם סוקטים והערך שלהם הוא class ייעודי שמשמר את המידע הדרוש לסוקט.
* המחלקה שהגדרנו מכילה את הערכים הבאים: הפקודה שאותה הלקוח ביקש לבצע(RRQ/WRQ), שם הקובץ אותו הלקוח העביר בפקטה הראשונה, ההודעה (בבתים) שנרצה לשלוח לסוקט זה, האופסט שממנו אנו צריכים לקרוא במידת הצורך, כתובת הלקוח, מספר הבלוק אותו אנו נצטרך לשלוח בפקטת ACK או בDATA, אינדיקטור המעיד אם סיימנו את בקשת הלקוח או לא, מונה שמעיד על כמות השליחות מחדש שעשינו עבור פקטה מסוימת, טיימר ט'רד עבור השליחה מחדש.
* בכדי למנוע חסימות של sendto/recvfrom, נעזרנו במבנים שציינו לעיל ובמתודה select בכדי לשלוח ולקבל את הפקטות המתאימות ללקוחות הזמינים. בסיום ההתקשרות נסגור את הסוקט שפתחנו עבור קליינט זה.
* תהליך זמן הריצה – הmain נמצא בלולאה אינסופית שכן השרת מחכה לקבל לקוחות חדשים. בהינתן לקוח חדש ששלח פקטה ראשונה לשרת (שכן זהו UDP), השרת פותח לו סוקט ייעודי ומוסיף אותו לרשימת send\_to שכן עלינו לשלוח לו פקטה ראשונה (בין אם קיבלנו בקשת RRQ בה עלינו לשלוח data, בין אם זה WRQ בה עלינו לשלוח ACK עם מספר בלוק 0), אלא אם כן קיבלנו ERROR ואז ישירות אנו מסמנים TRUE באינדיקטור של "האם סיימנו". לאחר מכן, בודקים אם הסוקט של הלקוח ניתן לכתיבה, אם כן שולחים לו את הפקטה, מעבירים אותו לרשימת recv\_from וכן פותחים לו טיימר ט'רד שמפעיל את פונ' timer\_handler במקרה והזמן נגמר.
* כאשר הסוקט נמצא ברשימת recv\_from אנו בודקים כי הוא ניתן לקריאה, אם כן, אנו מקבלים ממנו את ההודעה ובודקים כי זו ההודעה שציפינו לראות ופועלים בהתאם. כלומר, מוסיפים את הסוקט לרשימת send\_to.
* הפונ' update\_before\_send\_dict מעדכנת את ההודעה שיש לשלוח עבור כל סוקט שאמורה להישלח אליו הודעה.
* אם הטיימר ט'רד נגמר אנו עוברים לtimer\_handler, אם יש לנו עוד נסיונות לשלוח אנו נוסיף את הסוקט לרשימת send\_to בכדי לשלוח לו את ההודעה מחדש ואם אין עוד נסיונות נסגור אותו ונסיר את הסוקט מהמילון ומהרשימות בהן נמצא.

חלק 1 - הערות והחלטות עיצוביות

* החלטנו להתייחס לכל השגיאות באופן זהה - כלומר הדפסנו את השגיאה וסגרנו את הסוקט במקרה והיה פתוח. במקרה של ctrl+c הדפסנו הודעת "ביי ביי" וסגרנו את הסוקטים שהיו פתוחים וכן יצאנו מהתוכנית.

חלק 2 - תיאור של התכניות וכיצד להריץ אותן

* ישנו קובץ tftp\_server המתאר את התנהגות השרת המממשת פרוטוקול TFTP כמתואר בתיעוד. בדבר מקרי קצה נפרט בהמשך.
* בכדי להריץ את התכנית יש לפעול כמתואר בגוף התרגיל.



חלק 2 - תיאור של הפרוטוקול שמימשנו

* מימשנו את צד השרת של פרוטוקול TFTP במוד octet כפי שמתואר במסמך התיעוד המצורף בגוף התרגיל.

חלק 2 – מבנה התכנית

* התכנית מכילה קובץ יחיד כיוון שכך התבקשנו. התכנית מחולקת לפונקציות כך שפונק' הmain מקבלת פקטות מקליינטים שונים באמצעות הסוקט הראשי של הסרבר ובעת שקיבלה פקטה פותחת סוקט ייחודי לתקשורת מול הקלינט שממנו הגיעה הפקטה. במידה והפקטה היא פקודת WRQ נעבור לפונ' write שתנהל את התקשורת מול אותו הקליינט דרך הסוקט שפתחנו לו, כנ"ל לגבי RRQ עם פונ' read. בכל מקרה אחר, נשלח את השגיאה המתאימה לאותו הקליינט, גם באמצעות הסוקט המתאים. בסיום ההתקשרות נסגור את הסוקט שפתחנו עבור קליינט זה.
* ישנן בקוד פונ' עזר כדוגמת send\_msg המטפלת בפעולת השליחה של פקטה,   
  wait\_for\_acknowledgement המממשת את המנגנון בו מחכים ל-ack מתאים עבור הפקטה ששלחנו וכן את מנגנון ה-retransmit.

חלק 2 - החלטות עיצוביות

כעת נתאר את התנהלות הפרוטוקול :

* אם הפקודה הראשונה שאנו מקבלים מהקליינט אינה פקודת read/write אנו שולחים לו שגיאה המעידה על כך שזו אינה פקודת TFTP תקינה וממשיכים להמתין לפקודות ממנו. (לפי ד"ר רוכמן, זוהי החלטה עיצובית שלנו).
* בהינתן פקטה כנ"ל, נתחיל תקשורת עם הקליינט המזוהה עם ה TID שלה, כאשר סוג התקשורת הוא ביצוע read או write בהתאם לפקודה שקיבלנו בפקטה הראשונה בהתקשרות, כלומר אם היא פקטת WRQ או RRQ , ואז נתחיל להזרים/לקבל את המידע ל/מ הקליינט המתאים, כמתואר בפרוטוקול.
* לפי המתואר בפרוטוקול, לאחר קבלת כל פקטה שהיא, ייפתח סוקט ייעודי בסרבר שהפורט שלו ייבחר רנדומית מבין 0 עד 65535 כולל. במידה והפורט שייבחר רנדומית זהה או תפוס על ידי הסרבר, bind תזרוק שגיאה מסוג OSError שתטופל כמו כל שגיאות מסוג זה. כלומר, במידה ושני הסוקטים פתוחים נסגור ואתם ונדפיס את השגיאה המתאימה.
* כעת נתאר את מנגנון ה acknowledgment עבור שליחת פקטה. כמתואר בפרוטוקול, עבור כל פקטה שנשלחה (פרט לאחרונה, לפי ד"ר רוכמן) . עבור פקטה שהתקבלה בשלב בו אנו מחכים ל acknowledgment , אם הפקטה שקיבלנו לא בפורמט הנכון (כלומר, אם אנחנו בwrite היינו צריכים לקבל פקטת Data, אם אנחנו בread היינו צריכים לקבל פקטת Ack וכן גם קבלת פקטת Error נחשבת כ acknowledgment – בשני המקרים) אז אנחנו מתעלמים וממשיכים לחכות עד שנקבל acknowledgment כלשהו, בפורמט הנכון. זאת כמובן הולך יד ביד עם מימוש מנגנון ה- re-transmit , כמתואר בפרוטוקול.
* בעת קריאה פקודת write בחרנו לפתוח את הקובץ עם הדגל "a-" כלומר להוסיף את הדאטה שנקבל לסוף הקובץ הדרוש אם קיים כבר קובץ בשם זה, או נפתח קובץ חדש וריק, ונמלא אותו במידע שהתקבל אם לא קיים כבר קובץ בשם זה בשרת. (החלטה זו התקבלה לאחר התייעצות עם ד"ר רוכמן בשעת קבלה)
* גודל הפקטה בrecvfrom שהוגדר הינו 1024, זהו גודל , גדול מספיק (כפעמיים מגודל החסם על גודל ה data שמועברת בפקטה במהלך הפרוטוקול ). גודל זה נקבע לאחר התייעצות עם ד"ר רוכמן.