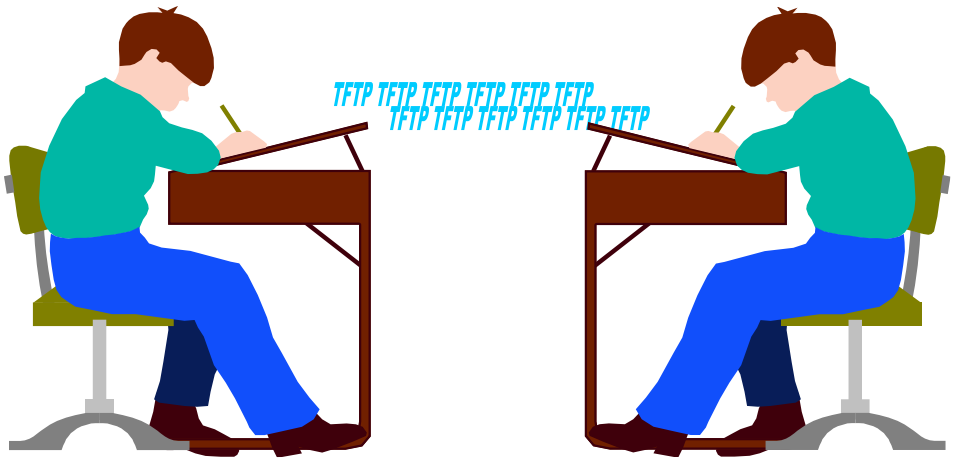
תרגיל בית #7 - רטוב-3



כללי

במהלך תרגיל זה, נממש גרסה **מפושטת** של שרת TFTP (Trivial File Transfer Protocol). לפני שנעבור לתאר את המימוש הנדרש מכם במסגרת התרגיל ניתן מספר פרטים על פרוטוקול TFTP המלא.

פרוטוקול TFTP

פרוטוקול TFTP משמש להעברת קבצים בין מחשבים שונים ומהווה גרסה מצומצמת מאוד של פרוטוקול FTP (File Transfer Protocol). בגלל הפונקציונליות המוגבלת וחוסר security, השימוש בפרוטוקול זה בזמננו הוא מוגבל מאוד. כיום הוא נמצא בעיקר ברשתות סגורות, מאובטחות ומבודדות (ללא יציאה החוצה), ומשמש בהן לטעינת ה image של ה- kernel של מערכת ההפעלה בפלטפורמות שאינן מכילות כונן קשיח (או אמצעי אחסון לא מחיק אחר).

בנוסף, נזכיר את השימוש שעושים בו יוצרי וירוסים כמנגנון הפצת תולעים (computer warms).

התכונות העיקריות של הפרוטוקול הן:

- שימוש ב UDP (ולא TCP)
- חוסר תמיכה בהזדהות או הצפנה של התוכן
- תמיכה בהעברת נתוני ascii ובינארי (ההבדל הוא בהמרה של תו מעבר שורה אשר שונה מפלטפורמה לפלטפורמה). סוג נתונים נוסף שכמעט ולא נתמך הוא mail.

תוכלו למצוא פרטים נוספים על הפרוטוקול באינטרנט, למשל ב:
http://en.wikipedia.org/wiki/Trivial_File_Transfer_Protocol
כמו כן הפרוטוקול (בגרסה 2) מתואר במלואו כאן:
https://datatracker.ietf.org/doc/html/rfc1350

המימוש הנדרש

על מנת להקל על מלאכת המימוש, להלן מספר הנחות:

- תמיכה אך ורק בפעולת ה WRQ (Write Request).
- תמיכה בחיבור בו-זמני של לקוח אחד בלבד.
- תמיכה רק בחבילות מסוג octet (חבילות מידע בינאריות אשר לא דורשות תרגום, בניגוד לחבילות ascii).

מהלך תקשורת תקין

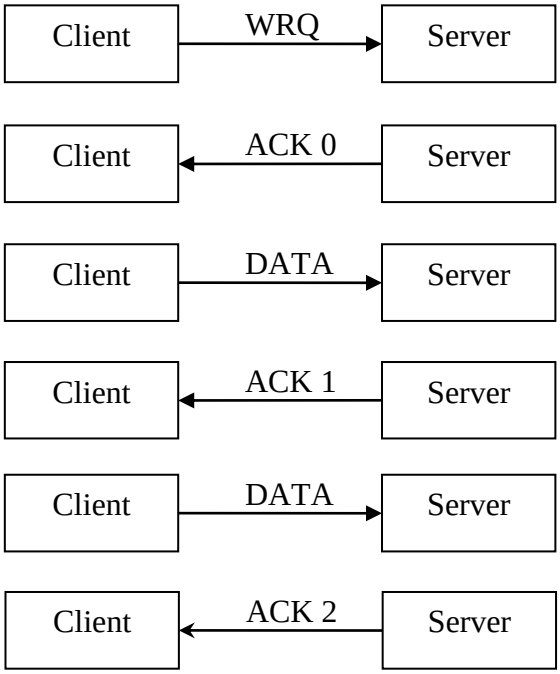
השרת מאזין על UDP Port מסוים. ברגע שמגיעה בקשת WRQ השרת מחזיר packet של ack לאחר מכן השרת מחכה למידע מה-client ולאחר כל packet של מידע השרת צריך להגיב עם ACK packet - פירוט של מבנה כל packet מופיע בהמשך. התקשורת מסתיימת ברגע שהשרת מקבל packet של data באורך קצר מ 516 בתים. במידה שגודל הקובץ מתחלק ב 512 ללא שארית אזי ה packet האחרון שישלח יהיה באורך של 4 בתים, דהיינו יכיל רק את ה header.

מבנה של data packet מוכל מסה"כ 516 בתים: 4 בתים של HEADER ה TFTP ועוד (עד) 512 בתים של נתוני הקובץ.

דוגמא להתקשרות אופיינית (העברת קובץ בגודל 512-1024 בתים):

1. השרת מאזין על UDP port מספר 69 (שימו לב כי הנכם נדרשים לקלוט את מספר הפורט להאזנה בשורת הפרמטרים).
2. השרת מקבל בקשת כתיבה מה-client (packet מסוג WRQ).
3. השרת מגיב עם ack packet עם מספר בלוק = 0.
4. ה-client שולח את ה-packet הראשון מסוג DATA. מספר בלוק נתונים = 1. אורך הנתונים שנשלחו 512 (שהם 512 הבתים הראשונים של הקובץ הנדרש).
5. השרת שולח ל-client packet מסוג ACK (Acknowledge) עם מספר בלוק = 1.

6. ה-client שולח packet נוסף מסוג DATA. מספר בלוק נתונים = 2. אורך הנתונים שנשלחו **פחות מ-512**.
7. השרת שולח לשרת packet מסוג ACK (Acknowledge) עם מספר בלוק = 2.
8. הגענו לסוף של ה-session מסוג WRQ. השרת חוזר להאזין על ה-UDP port מספר 69.



להלן פירוט של תוכן החבילות העוברות בין לקוח לשרת בתסריט זה:

WRQ					
Size	2 bytes	string	1	string	1
Field description	Opcode	File name	String terminator	Transmission mode	String terminator
Sample content	2	file.txt	0	octet	0

ACK 0		
Size	2 bytes	2 bytes
Field description	Opcode	Block number
Sample content	4	0

Data 1			
Size	2 bytes	2 bytes	512
Field description	Opcode	Block number	Data
Sample content	3	1	*Data from the file (512 bytes)*

ACK1

Size	2 bytes	2 bytes
Field description	Opcode	Block number
Sample content	4	1

Data 2			
Size	2 bytes	2 bytes	Less than 512
Field description	Opcode	Block number	Data
Sample content	3	2	*Data from the file – in our example this is the last block so its size is less then 512 bytes*

ACK 2		
Size	2 bytes	2 bytes
Field description	Opcode	Block number
Sample content	4	2

טיפול בתקלות תקשורת

פרוטוקול TFTP עובד מעל UDP שאינו מספק טיפול בבעיות שעלולות להיווצר בהתקשרות מבוססת packets (למשל: packet לא מגיע ליעדו כיוון שנדחה על ידי אחד הנתבים בדרך בגלל עומס יתר, או אותו packet שמגיע פעמיים). לכן ה spec של TFTP מטפל בתקלות אלו ברמה של האפליקציה. השרת מגיב באמצעות שליחה חוזרת של הודעה או זניחה של תהליך העברה. במקרה של זניחה של תהליך העברה, השרת חוזר להמתין להודעת WRQ ולא ממשיך לנסות להשלים את העברת הקובץ הנוכחית. להלן רשימה של "מקרים ותגובות" שעליכם לממש:

מקרה	תגובה
לא התקבל שום packet בזמן שהוקצב (3 שניות בדוגמא להלן)	• שליחת ack נוסף (עם אותו המספר הבלוק של הDATA הקודם שהתקבל) • להגדיל את מונה הכשלונות
התקבל packet שונה ממה שמצפים אליו	• שגיאה חמורה – זונחים את תהליך ההעברה ושולחים הודעת שגיאה ללקוח (ראו פירוט בהמשך)
מונה הכשלונות גדול גדול מהמקסימום	• שגיאה חמורה – זונחים את תהליך ההעברה ושולחים הודעת שגיאה ללקוח (ראו פירוט בהמשך)

Alignment של שדות ב-struct

כאשר מגדירים struct בעל מספר שדות, המהדר דואג לעשות alignment לשדות לגבול אינטגרלי של הזיכרון (הדבר נועד לאפשר גישה מהירה יותר לנתונים), לכן כשמגדירים את המבנה הבא:

```
struct my_struct{
    char    a;
    char    b;
};
```

המבנה שנוצר בפועל בזיכרון הוא כזה:

Offset in bytes	+0	+1	+2	+3	+4	+5	+6	+7
Content	A	unused	Unused	unused	b	unused	unused	unused

על מנת ליצור את ה-struct כך שהשדות העוקבים יוצמדו זה לזה בזיכרון יש להשתמש בסינטקס הבא:

```
struct my_struct{
    char    a;
    char    b;
} __attribute__((packed));
```

שגיאות System calls

במקרה שקריאת מערכת נכשלת יש להדפיס למסך:

TTFTP_ERROR:<*error message*>
Use perror function to print out the error message.

ולצאת מהתוכנית.

סיום השרת

השרת רץ לנצח בלולאה אינסופית.

שליחת הודעת Error ללקוח

במקרים שבהם השרת מקבל חבילות שאינו מצפה להן או כאשר מונה הכשלונות גדול מהערך המקסימלי, יש לשלוח ללקוח הודעה מסוג Error (ראו פורמט בהמשך):

מקרה	קוד שגיאה	הודעה
התקבלה חבילה מלקוח שלא שלח חבילת WRQ	7	Unknown user
התקבלה חבילת WRQ עם קובץ שכבר קיים	6	File already exists
התקבלה חבילת WRQ מלקוח שנמצא במהלך שידור קובץ או התקבלה חבילה כלשהיא מלקוח A בזמן שהשרת מטפל בלקוח B	4	Unexpected packet
התקבלה חבילת DATA עם מספר בלוק לא תקין	0	Bad block number
מונה הכשלונות גדול מהערך המקסימלי	0	Abandoning file transmission

Error				
Size	2 bytes	2 bytes	string	1
Field description	Opcode	Error code	Error message	string terminator
Sample content	5	7	Unknown user	0

שונות

שימו לב שאתם:

- לא שוכחים להשתמש בפונקציות שינוי סדר הבתים (htons,htonl,ntohs,ntohl).
- מטפלים בערכי שגיאה המוחזרים על ידי קריאות מערכת ולא "משתיקים" אותם.
- משחררים את כל המשאבים אותם הקצתם.
- על שם ה-executable להיות ttftps (פירוש Trivial Trivial FTP Server) ועליו לקבל משורת הפקודה כפרמטר את מספר הפורט עליו השרת יאזין, את ערך ה-timeout בשניות ואת מספר הכשלונות המקסימלי. טווח תקין של הפרמטרים הוא חיובי ממש (גדול מאפס) ובגבולות ה-unsigned short. אם התקבלו מספר שגוי של פרמטרים או פרמטר לא תקין יש להדפיס הודעת שגיאה למסך (TTFTP_ERROR: illegal arguments) ולצאת מהתוכנית.
  ./ttftps <port> <timeout> <max_num_of_resends>
- כשתדבגו את השרת עליכם לוודא כי אינכם משתמשים ב-Port של אחת האפליקציות האחרות. יש להשתמש במספר פורט גדול מ-10000.
- על מנת לבדוק את השרת שלכם השתמשו ב TFTP client אותו תוכלו להוריד מ- moodle כחלק מהקבצים של התרגיל. שם הקובץ הוא tftp. זהו executable אותו תוכלו להעתיק למכונה הווירטואלית שלכם. על מנת שתוכלו להריצו, יש לוודא כי לקובץ יש הרשאות execute. השתמשו בפקודת chmod לשם כך.
- הקבצים יעלו לתיקיה ממנה הופעל השרת. אם תהליך ההעברה נכשל אין ליצור קבצים בשרת.
- בשביל לבדוק, נא לפתוח 2 טרמינלים במקביל, אחד עם השרת והשני עם הclient.
- מימוש לא נכון של השרת יכול לגרום לclient לקרוס עם segmentation fault.
- יש לממש את התרגיל ב-C/C++ בלבד.
- ניתן להניח שגודל חבילה מקסימלי שיתקבל על ה-socket הוא 516 בתים.
- אין לייצר תהליכי בנים או חוטים נוספים על מנת לממש את התרגיל. יש להשתמש בחוט יחיד.
- זיהוי לקוח מתבצע באמצעות כתובת IP ופורט של מקור ההודעה.
- בגלל שהשרת מבוסס UDP, יש לייצר socket יחיד בשרת.
- שאלות על התרגיל יש לפרסם בפורום תרגילי בית רטובים. יש לעקוב אחרי הדיון "עדכונים תרגיל רטוב 3#" במידה ונעדכן את דרישות התרגיל.

פונקציות שימושיות

להלן רשימה של פונקציות שיכולות לעזור לכם במימוש: socket, bind, sendto, recvfrom, recv, select, ioctl,

```
int select(int nfds, fd_set *readfds, fd_set *writefds,
fd_set *exceptfds, const struct timeval *timeout);
```

פונקציה זו מאפשרת לבדוק אם הסוקט מוכן לקריאה\כתיבה. פונקציה זו **אינה פונקציה חוסמת לצמיתות** (חוסמת ל(min(timeout, time_until_packet_arrives). צריך להשתמש בפונקציה זו ע"מ לבדוק אם קיים מידע לקרוא במשך זמן מסוים. אם לא קיים מידע וחיכינו זמן מוגדר מראש (בתוך הstruct של timeval) אז select מחזיר 0. אם קיים מידע אז מחזיר ערך חיובי ואחרת (שגיאה כלשהי) מחזיר ערך שלילי. שימו לב כי nfds צריך להכיל את המספר של הfd הכי גבוהה (מבין אלה שבודקים) ועוד 1!! ז"א אם יש fd שערכו 2 ורק אותו מעוניינים לבדוק nfds יהיה 3.

דוגמא לשימוש, מידע נוסף ושימוש בstruct timeval:

http://manpages.courier-mta.org/htmlman2/select.2.html

אפשר להשתמש בפונקציה זו בלולאה ע"מ לממש את המנגנון של ה-timeout.

הידור קישור ובדיקה

יש לוודא שהקוד שלכם מתקמפל ע"י הפקודה הבאה:

אם כתבתם ב-C++:

```
> g++ -std=c++11 -Wall -Werror -pedantic-errors -DNDEBUG *.cpp -o ttftps
```

אם כתבתם ב-C:

```
> gcc -std=c99 -Wall -Werror -pedantic-errors -DNDEBUG *.c -o ttftps
```

יש לוודא שנוצר קובץ הרצה ללא שגיאות או warnings.

עליכם לספק Makefile עבור בניית הקוד. הכללים המינימליים שצריכים להופיע ב-Makefile הינם:

- כלל ttftps שיבנה את התוכנית ttftps.
- כלל עבור כל קובץ נפרד שקיים בפרויקט.
- כלל clean אשר מוחק את כל תוצרי הקימפול.
- יש לוודא שהתוכנית נבנית ע"י הפקודה make.
- יש לקמפל ע"י הדגלים המופיעים בחלק "הידור קישור ובדיקה" לעיל.

לתרגיל זה מצורף סקריפט check_submission.py המוודא (בצורה חלקית) את תקינות ההגשה. הסקריפט מצורף לנוחיותכם, ובנוסף לבדיקה באמצעות הסקריפט, עליכם לוודא את תקינות ההגשה.

הסקריפט מצפה ל-2 פרמטרים: נתיב ל-zip, ושם קובץ ההרצה. לדוגמא:

```
> ./check_submission.py 123456789_987654321.zip ttftps
```

הגשה:

הנחיות כלליות על אופן הגשת תרגילי הבית הרטובים ניתן למצוא באתר הקורס תחת הכותרת "עבודות בית – מידע ונהלים".

בבקשה, בדקו שהתוכניות שלכם עוברות קומפילציה.
תוכנית שלא תעבור קומפילציה לא תבדק!

בהצלחה!!!