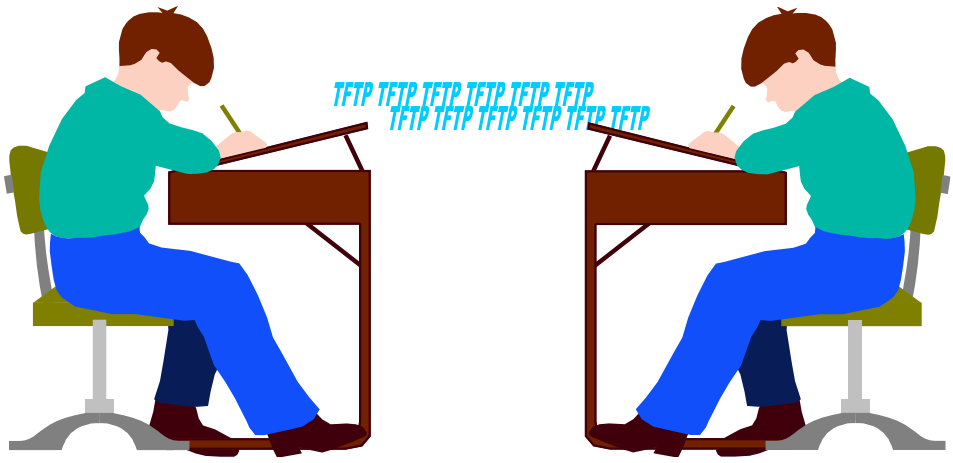
תרגיל בית #7 - רטוב-3



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: http://tools.ietf.org/html/rfc1350.

המימוש הנדרש

על מנת להקל על מלאכת המימוש, להלן מספר הנחות:

- מימוש שרת TFTP בלבד.
- תמיכה אך ורק בפעולת ה WRQ (Write Request).
- שליחת ACK בלבד
- תמיכה בחיבור בו-זמני של client בודד.
- תמיכה רק בחבילות מסוג octet (חבילות מידע בינאריות אשר לא דורשות תרגום, בניגוד לחבילות ascii)

מהלך תקשורת תקין

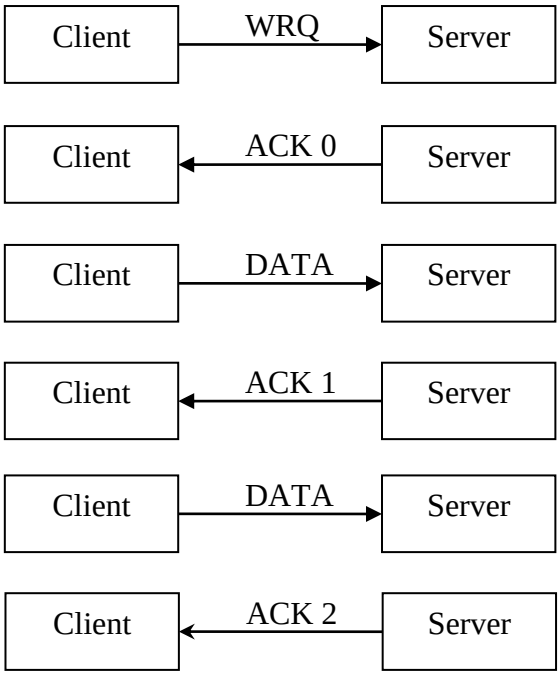
השרת מאזין על UDP Port מסוים. ברגע שמגיעה בקשת WRQ השרת מחזיר packet של ack לאחר מכן השרת מחכה למידע מה-client ולאחר כל packet של מידע השרת צריך להגיב עם ACK packet - פירוט של מבנה כל packet מופיע בהמשך. התקשורת מסתיימת ברגע שהשרת מקבל packet של data באורך קצר מ 516 בתים. במידה שגודל הקובץ מתחלק ב 512 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-packet client 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. 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 (block_num+1 של הack הקודם)	• שגיאה חמורה – זונחים את תהליך ההעברה
מונה הכשלונות גדול מערך מסויים (7 בדוגמא להלן)	• שגיאה חמורה – זונחים את תהליך ההעברה

כיוון שהאלגוריתם הדרוש הינו מורכב למדי, להלן שלד של שגרה הכתובה ב C בו תוכלו להיעזר למימוש האלגוריתם. פשוט מלאו את החלקים החסרים. הקוד הבא מניח כי הקובץ שמקבלים עבורו את הנתונים להעברה כבר נוצר ובקשת השליחה התקבלה ונשלח ack 0 ובנוסף ה-sockets כבר קונפגו.

```
const int WAIT_FOR_PACKET_TIMEOUT = 3;
const int NUMBER_OF_FAILURES = 7;


do
{
    do
    {

        do
        {
            // TODO: Wait WAIT_FOR_PACKET_TIMEOUT to see if something appears
            //       for us at the socket (we are waiting for DATA)

            if ()// TODO: if there was something at the socket and
                 //       we are here not because of a timeout
            {
                // TODO: Read the DATA packet from the socket (at
                //       least we hope this is a DATA packet)
            }

            if (...) // TODO: Time out expired while waiting for data
                     //       to appear at the socket
            {
                     //TODO: Send another ACK for the last packet
                     timeoutExpiredCount++;
            }

           if (timeoutExpiredCount>= NUMBER_OF_FAILURES)
            {
            // FATAL ERROR BAIL OUT
             }


        }while (...) // TODO: Continue while some socket was ready
                     //       but recvfrom failed to read the data (ret 0)

        if (...) // TODO: We got something else but DATA
        {
            // FATAL ERROR BAIL OUT
        }

        if (...) // TODO: The incoming block number is not what we have
                 //       expected, i.e. this is a DATA pkt but the block number
                 //       in DATA was wrong (not last ACK's block number + 1)
        {
             // FATAL ERROR BAIL OUT
        }
    }while (FALSE);
    timeoutExpiredCount = 0;
    lastWriteSize = fwrite(...); // write next bulk of data
   // TODO: send ACK packet to the client


}while (...); // Have blocks left to be read from client (not end of transmission)
```

בנוסף, תוכלו להיעזר ב-spec המלא של הפרוטוקול (http://tools.ietf.org/html/rfc1350) אם חסרים לכם פרטים בתיאור הנ"ל.

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 (למשל, הדבר חשוב כאשר נגדיר מבנה עבור בלוק ה WRQ) יש להשתמש בסינטקס הבא:

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

הדפסות ניטור

הנכם נדרשים להדפיס הדפסות ניטור במקרים הבאים:

מקרה	פורמט ההדפסה
קבלת packet מסוג WRQ	IN:WRQ,<*filename*>,<*mode*> Where <*filename*> and <*mode*> are values of appropriate fields in the packet.
שליחת packet מסוג ACK	OUT:ACK,<*block number*> Where <*block number*> is the block number field of the packet or confirmation of WRQ (0).
קבלת packet מסוג DATA	IN:DATA, <*block number*>,<*packet length*> Where <*block number*> is the block number field of the packet and <*packet length*> is the length of the packet
כתיבה של בלוק מידע מהקובץ הנשלח	WRITING: <*size*> Where <*size*> is the number of bytes that was written.
סיום תקין של קבלת קובץ	RECVOK
סיום לא תקין של שידור קובץ	RECVFAIL
כל אחד מהמקרים של תקלה בזרימת האלגוריתם: **לא התקבל שום packet בזמן שהוקצב** **התקבל packet שונה מ-ACK קודם+1** **מונה הכשלונות גדול מערך הסף**	FLOWERROR:<*description*> Where <*description*> is some message describing the error
שגיאה כלשהי בקריאת system-call	TTFTP_ERROR:<*error message*> Use perror function to print out the error message.

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. אם התקבלו מספר שגוי של פרמטרים או פרמטר לא תקין יש להדפיס הודעת שגיאה למסך ולצאת מהתוכנית.
- 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.
- אם שם קובץ קיים על השרת דורסים אותו. 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.
- במידה וקריאת מערכת נכשלת יש להדפיס הודעה באמצעות perror ולצאת מהתוכנית.
- לאחר זניחת תהליך ההעברה יש לחזור ולהמתין להודעת WRQ מהלקוח הבא.

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 לבדוק כל כמה שניות (במקרה שלנו כל 3 שניות) אם קיים מידע לקרוא. אם עבר יותר מידי זמן (7 פעמים קרה timeout) אז מפסיקים את התהליך (כפי שרשום למעלה בטיפול בשגיאות)

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!!!