TFTP

Client

התוכנית מתחילה בקריאת הארגומנטים שהועברו לתוכנית (IP ו-PORT)

במידה וארגומנטים אלו לא סופקו התוכנית תופסק(הצגת הודעה שגיאה מתאימה).

במידה והארגומנטים תקינים אז התוכנית מנסה ליצור חיבור TCP בינה לבין הכתובת שסופקה כארגומנט לתוכנית.

במידה והחיבור נכשל מודפסת הודעה שגיאה ובמידה והחיבור הושלם בהצלחה

התוכנית נכנסת לולאה אינסופית וקוראת לתוך BUFFER פקודה שהמשתמש הקליד.

התוכנית מבצעת קיצוץ רווחים מה-BUFFER (trim) ומשווה האם הפקודה שהוקלדה מתחילה באחד מהטקסטים הבאים:

* PUT
* GET
* MGET
* MPUT
* CmyD
* CD
* LIST
* CLOSE

במידה והפקודה שהוקלדה לא מתחילה באחת מהאופציות התוכנית מדפיסה

Invalid command ותבקש מהמשתמש להקליד שוב פקודה מחדש(לולאה אינסופית).

התהליך חוזר על עצמו עד אשר המשתמש יקליד אחד מהאופציות שרשומות.

לאחר שהמשתמש הקליד פקודה שקיימת התוכנית מפעילה פונקציה בהתאם לפקודה שהוקלדה לכול פקודה ישנה פונקציה אחת.

|  |  |
| --- | --- |
| פקודה שהוקלדה | פונקציה |
| PUT | int putFile(char\* strParam,int\* totalBytes,char\*\* errorMsg); |
| GET | int getFile(char\* strParam,int\* totalBytes,char\*\* errorMsg); |
| MPUT | void mPutFiles(char\*\* parameters); |
| MGET | void mGetFiles(char\*\* parameters); |
| CmyD | int changeDirectory(const char \*path); |
| CD | int changeServerDir(char\* dir,char\*\* errorMsg); |
| LIST | int getServerFileList(char\*\* errorMsg); |
| CLOSE | יציאה מהתוכנית על exit |

הפונקציות (מלבד mPutFiles (MPUT),(MGET) mGetFiles ) מחזירות 0 בהצלחה

ו-1- בכישלון.

במידה ופונקציה נכשלה, מודפסת סיבת הכישלון ובמידה והפונקציה הצליחה מודפס שהפקודה הושלמה.

הפונקציה (MPUT) mPutFiles -"קוראת" לפונקציה (PUT) putFile מספר פעמיים בלולאה (כמספר הפרמטרים שהוקלדו לאחר הפקודה MPUT )

לדוגמא :

אם המשתמש הקליד

1.txt,2.txt,3.txt MPUT

אז הפונקציה mPutFiles תפעיל בלולאה 3 פעמיים את הפונקציה putFile כול פעם אם פרמטר אחר בפעם הראשונה עם הפרמטר 1.txt

בפעם השנייה עם הפרמטר 2.txt

ובפעם השלישית עם הפרמטר 3.txt

בכול פעם ש- mPutFiles קוראת לפונקציה putFile היא בודקת האם putFile החזירה אפס או מינוס אחד כלומר האם היא הצליחה או נכשלה .

mPutFiles מציגה כמה קבצים נשלחו בהצלחה וכמה נכשלו ומה הייתה סיבה לכישלון כמו כן בעת הצלחה היא מדפיסה כמה BYTE הועברו ללא HEADER.

לדוגמא

אם לאחר הפקודה

1.txt,2.txt,3.txt MPUT

putFile החזירה 0 בפעם הראשונה

putFile החזירה 0 בפעם השנייה

putFile החזירה -1 בפעם השלישית

אז mPutFiles מדפיסה 2 קבצים עברו בהצלחה וכמו כן את גודלו של כול קובץ

ו-1 קובץ נכשל ומדפיסה את סיבת הכישלון.

אותו עיקרון גם בפונקציה mGetFiles (MGET)

שפעילה את הפונקציה getFile מספר פעמיים.

מקרים שבהם התוכנית מופסקת

1. הוקלדה הפקודה CLOSE (במקרה זה התוכנית סוגרת את החיבור לשרת ויוצאת מהתוכנית בצורה מסודרת על ידי exit)
2. מתרחש שגיאה קריטית (כגון אי הקצאת זיכרון ואז גם כן התוכנית מופסקת על ידי ה-exit .

בדיקת תקינות הפקודות המקלדות:

כל פקודה שמוקלדת התוכנית בודקת האם היא אחת מהפקודות ברשימה.

במידה והפקודה מקבלת פרמטרים אז נבדק שהפורמט של הפרמטרים תקין

(כלומר בין פרמטר לפרמטר יש פסיק אחד בלבד שמפריד ביניהם)

למשל במקרה

1,,, MPUT

יודפס שגיאה(invalid command)

פירטו הפקודות

כאשר המשתמש הקליד את פקודה XXX GET

מופעלת הפונקציה getFile

התוכנית מנסה לפתוח לכתיבה את הקובץ המבוקש ובמידה והיא נכשלת מוצגת הודעת שגיאה והפונקציה מופסקת.

התוכנית מייצרת Buffer בגודל 2+אורך +1+XXX6(MODE octet)

ושולחת את הבקשה לשרת(sendRequest)

2 בתים ראשונים OPCODE(RRQ)

XXX+1=ארוך שם קובץ + 0 בסוף

6=MODE ==> octet

התוכנית ממתינה לתשובה (opcode=ERROR או opcode=DATA)

במידה והתוכנית קיבלה opcode=ERROR מודפסת השגיאה שהתקלבה מהשרת והפונקציה מופסקת.

במידה ומקבלים opcode=DATA אז נשלחת הודעת ACK לשרת שמכילה את מספר ה-BLOCK שהתקבל אחרון+opcode=ACK.

התהליך הזה חוזר על עצמו עד שמקבלים מהשרת Block נתונים שמספרו קטן מ-516 כלומר זה מציין שזהו ה-BLOCK האחרון או שמקבלים opcode=ERROR.

הבלוקים שמתקבלים נרשמים לספרייה הנוכחית(במידה ואי אפשר לרשום לספרייה הנוכחית מוצגת הודעה שגיאה והפונקציה מופסקת).

בסיום נסגר הקובץ.

כאשר המשתמש הקליד את פקודה XXX PUT

מופעלת הפונקציה putFile

התוכנית מנסה לפתוח לקריאה את הקובץ המבוקש ובמידה ואי נכשלת מוצגת הודעת שגיאה והפונקציה מופסקת.

במידה והקובץ נפתח לקריאה אז התוכנית מייצרת Buffer בגודל 2+אורך +1+XXX6(MODE octet)

ושולחת את הבקשה לשרת(sendRequest)

2 בתים ראשונים OPCODE(WRQ)

XXX+1=ארוך שם קובץ + 0 בסוף

6=MODE ==> octet

התוכנית ממתינה לתשובה (opcode=ERROR או opcode=ACK)

כאשר בפעם הראשונה צריך להגיע ACK=0

במידה ומגיע opcode=ERROR מודפסת הודעת שגיאה והפונקציה מופסקת.

במידה ומגיע opcode=ACK אז התוכנית שולחת את ה-Block מספר 1 וממתינה ל--ACK1 וכך התהליך חוזר על עצמו כאשר כול פעם שלוחים בלוק מספר i ומחכים ל-ACK מספר i

התהליך מפסיק או בקבלת opcode=ERROR או שמספר ה-BYTES שנשלחו קטן מ-512.

כאשר המשתמש הקליד את פקודה,YYY XXX PUT M

במקרה זה מופעלת הפונקציה mPutFiles

פונקציה זו בודקת שהפרמטרים שמצורפים לפקודה בפורמט נכון.

במידה ולא מודפסת הודעת שגיאה.

במידה והפרמטרים תקינים מתבצעת קריאה לפונקציה putFile כמה פעמיים כמספר הפרמטרים.

כאשר המשתמש הקליד את פקודה,YYY XXX GET M

במקרה זה מופעלת הפונקציה mGetFiles

פונקציה זו בודקת שהפרמטרים שמצורפים לפקודה בפורמט נכון.

במידה ולא מודפסת הודעת שגיאה.

במידה והפרמטרים תקינים מתבצעת קריאה לפונקציה getFile כמה פעמיים כמספר הפרמטרים.

כאשר המשתמש הקליד את פקודהXXX CD

במקרה זה מופעלת הפונקציה changeServerDir

התוכנית מייצרת Buffer בגודל 2+אורך +1+XXX6(MODE octet)

ושולחת את הבקשה לשרת(sendRequest)

2 בתים ראשונים OPCODE(CD)

XXX+1=ארוך שם ספרייה + 0 בסוף

6=MODE ==> octet

התוכנית ממתינה לתשובה (opcode=ERROR או opcode=ACK)

במידה ומגיע opcode=ACK מוצגת הודעת הצלחה ובמידה ומגיע opcode=ERROR מוצגת הודעת כישלון שהגיע מהשרת.

כאשר המשתמש הקליד את פקודה LIST

במקרה זה מופעלת הפונקציה getServerFileList

התוכנית מייצרת Buffer בגודל 2++1+6(MODE octet)

ושולחת את הבקשה לשרת(sendRequest)

2 בתים ראשונים OPCODE(LIST)

1= 0 מפריד

6=MODE ==> octet

התוכנית ממתינה לתשובה (opcode=ERROR או opcode=DATA)

במידה ומגיע opcode= DATA מודפס למסך תוכן המידע שהגיע ונשלח ACK על אותו בלוק

בעת קבלת opcode=ERROR תודפס הודעת השגיאה זו שנשלחה מהשרת.

גם כן התהליך מופסקת בעת קבלת בלוק נתונים שקטן 516 או בקבלת הודעת בשגיאה.

כאשר המשתמש הקליד את פקודה CmyD

מופעלת הפונקציה changeDirectory

שמפעילה את הפונקציה chdir

במידה וזו החזירה את הערך 0 תודפס הודעת הצלחה אחרת תודפס הודעת כישלון.

כאשר המשתמש הקליד את פקודה CLOSE

התוכנית סוגרת את החיבור לשרת ומסתיימת.

**צד השרת:**

לוגיקת השרת:

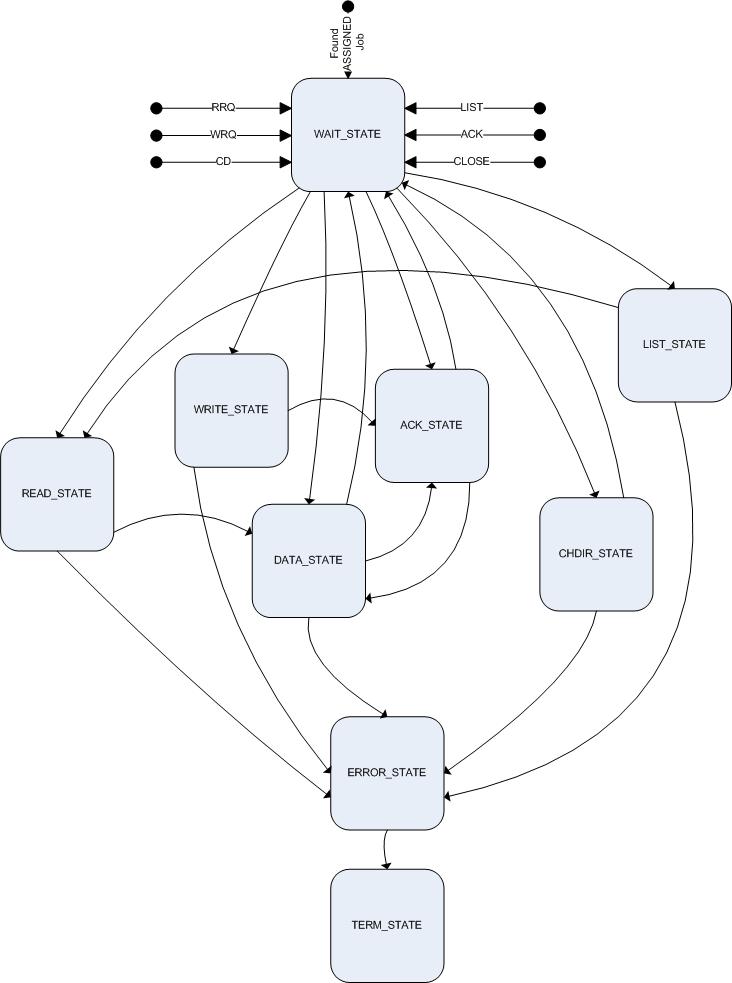
השרת מורכב משלושה תהליכונים (Threads) – אחד ראשי בו מתחילה התוכני ושניים נוספים המיוצרים בתהליכון הראשי ומשמשים כ-"משרתים" עבור כל בקשה.

בתהליך הראשי ישנו תור עבודות מעגלי שאיברים הם מבנים לתיאור עבודה, בעל שני מצביעים: אחד להכנסת עבודה כאשר מגיע בקשה מהלקוח והשני להוצאת עבודה לביצוע ע"י ה-"משרת" הפנוי הראשון.

התהליך הראשי מאזין לפורט TCP הקבוע מראש (ע"י המשתמש או ברירת המחדל שהיא 5069), כאשר מגיע בקשה, אם התור פנוי אזי

השרת ממלא את המבנה במידע על העבודה (כגון שם הקובץ, מספר הסוקט וכו'),במקום עליו מצביע מצביע ההכנסה ומסמן אותו כעבודה לביצוע, ואז חוזר להאזין לבקשות נוספות.

תהליכוני השירות סורקים את התור וכאשר מגיעים לעבודה המחכה לטיפול הם מסמן הראשון שהגיע אליה שהעבודה מטופלת על ידו. הטיפול בעבודה נעשה באמצעות מכונת מצבים:



כאשר מכונת המצב מגיעה למצב סופי אזי מרוקן מבנה העבודה ומתפנה לקראת הכנסת עבודה חדשה.

מכונת המצבים מומשה באמצעות לולאת WHILE שבתוכנה מקונן מבנה Switch-case.

מבני נתונים:

1. מבנה לייצור ופענוח כותרת TFTP:

בשידור מבנה זה ממולא ומועתק לחוצץ השידור ,ובקליטה חוצץ הקליטה מושלך על המבנה וכך מפוענחת הכותרת.

typedef struct \_s\_jobDescriptor

{

char cp\_client[16];//max string is '111.111.111.111\0'

char cp\_fileName[MAX\_STRING+1];

Status status; // transaction status

unsigned int ui\_packets; // number of packets transfered

double d\_time; // transfer time

int i\_socketId; // remote socket id

struct sockaddr\_in sa\_address; // remote ip address

socklen\_t i\_socketSize; // auxiliry variable to hold the size of the sockaddr structure

int i\_slave;

}s\_jobDescriptor;

struct tftphdr {

short th\_opcode; /\* packet type \*/

union {

unsigned short tu\_block; /\* block # \*/

short tu\_code; /\* error code \*/

char tu\_stuff[1]; /\* request packet stuff \*/

} \_\_attribute\_\_ ((\_\_packed\_\_)) th\_u;

char th\_data[1]; /\* data or error string \*/

} \_\_attribute\_\_ ((\_\_packed\_\_));

1. מבנה תאור עבודה:
2. מבנה תור העבודות:

typedef struct \_s\_jobQ

{

s\_jobDescriptor Q[MAX\_JOBS];

int input,output;

}s\_jobQ;