Hausarbeit WPF

Kurzbeschreibung:

Die Anwendung ermöglicht es Textdateien auf einem Server zu verwalten.

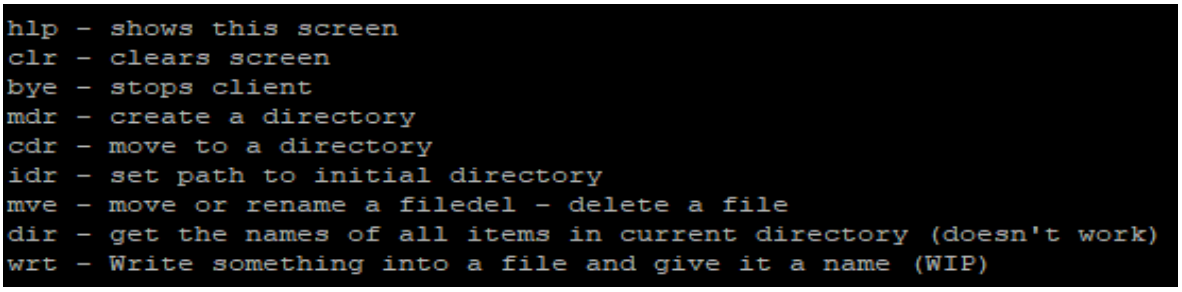
Der Funktionsumfang beinhaltet die Möglichkeit alle auf dem Dateiserver existierenden Dateien anzeigen zu lassen. Im Weiteren können einzelne Dateien ausgelesen und beschrieben werden. Außerdem ist es möglich Daten vollständig zu löschen.

Beim Start des Clients wird eine kurze Übersicht der enthaltenen Befehle angezeigt wobei deren Funktion zusätzlich hinterlegt ist. Falls erneut Bedarf des Hilfe-screens aufkommt, kann dieser mittels des „hlp“ Befehls aufgerufen werden.

Funktionsbeschreibung:

Als Kommunikations Protokoll für die Anwendung haben wir uns für UDP (User Datagram Protokoll) entschieden. UDP nutzt Ports um versendete Daten der richtigen Anwendung auf dem Ziel-Computer zukommen zu lassen.

Zu Programmstart wird dem Client eine Vielzahl von möglichen Anfragen an den Server angezeigt. Falls man die Befehlsübersicht erneut aufrufen möchte, genügt es den “hlp“ Befehl einzugeben.



Im Folgenden werde Ich die einzelnen Anfragen genauer erklären.

Der “dir“ Befehl schickt eine Anfrage an den Server, welcher dann eine Liste mit allen Dateien aus dem aktuellen Verzeichnis zurückschickt.

Die beiden Befehle “wrt“ (write) und “get“ sind für den Inhaltsaustausch der einzelnen Dateien verantwortlich. Bei “wrt“ wird der Client zuerst nach dem Text gefragt, den er in die Textdatei schreiben möchte. Danach wird er aufgefordert einen Namen für das Dokument anzugeben. Falls das Erstellen der Datei erfolgreich war, sendet der Server dem Client den Status 200 zurück. Sollte es Probleme beim Erstellen geben, erhält der Client den Fehlercode 400. Im Weiteren kann der Server den Fehlercode 402 zurückschicken, welcher indiziert das die Datei bereits existiert.

Der“get“ Befehl ermöglicht es dem Client explizit den Inhalt bestimmter Dateien vom Server anzufordern hierfür werden streams verwendet. Im Weiteren gibt der Server die Standartstatusmeldung zurück (200=Ok 400=Fail).

```
}else if(bufCom == "wrt"){ //Datei beschreiben
        sendto(s, bufCom.c_str(), bufCom.length(), 0, (sockaddr *)&si_other, slen);
        recv(s, bufCode, 3, 0);
        bufferCode = bufCode;
        if(bufferCode == "200"){
                cout << "Write here: ";
                getline(cin,buffer);
                sendto(s, buffer.c_str(), buffer.length(), 0, (sockaddr *)&si_other, slen);
                recv(s, bufCode, 3, 0);
                bufferCode = bufCode;
                if(bufferCode == "200"){
                        cout << "Give the file a name: ";
                        getline(cin,buffer);
                        sendto(s, buffer.c_str(), buffer.length(), 0, (sockaddr *)&si_other, slen);
                        recv(s, bufCode, 3, 0);
                        bufferCode = bufCode;
                        if(bufferCode == "200") cout << "File created";
                        else if(bufferCode == "400") cout << "File created but not filled";
                        else if(bufferCode == "402") cout << "File already exists";
                        else cout << "Unexpected Error";
```

Es gibt verschiedene Wege sich durch das auf dem Server liegende Dateisystem zu bewegen. Hierfür werden die Befehle “cdr“ und “idr“ benötigt. Mit “cdr“ (change Directory) kann man das aktuelle Verzeichnis auf dem Server wechseln.

```
}else if(bufCom == "cdr"){ //Verzeichnispfad ändern
        sendto(s, bufCom.c_str(), bufCom.length(), 0, (sockaddr *)&si_other, slen);
        recv(s, bufCode, 3, 0);
        bufferCode = bufCode;
        if(bufferCode == "200"){
                cout << "Directory name: ";
                getline(cin,buffer);
                sendto(s, buffer.c_str(), buffer.length(), 0, (sockaddr *)&si_other, slen);
                memset(bufCode, 0, sizeof(buf));
                recv(s, bufCode, 3, 0);
                bufferCode = bufCode;
                if(bufferCode == "200")cout << "Successfully moved";
                else if (bufferCode == "400")cout << "Did not move";
                else cout << "Unexpected error";
        }else cout << "Connection Error";
```

Wohingegen man mit “idr“ zurück zum root Verzeichnis „repo“ kommt.

```
}else if(bufCom == "idr"){ //Verzeichnispfad zurücksetzen
        sendto(s, bufCom.c_str(), bufCom.length(), 0, (sockaddr *)&si_other, slen);
```

Bei beiden Befehlen gibt der Server jeweils die Standartstatusmeldung (200=Ok 400=Fail) zurück.

Im Weiteren gibt es den “mve“ Befehl, mit welchem man eine Datei von ihrem Ursprungsverzeichnis auf dem Server in ein anderes Verzeichnis schieben kann. Zuerst sendet der Client dem Server den Namen der Datei, dieser wiederum sendet die Standartstatusmeldung (200=Ok 400=Fail) zurück. Sollte der Status 200 lauten, wird der Client aufgefordert den Namen des Verzeichnisses in das die Datei verschoben werden soll zu nennen. Der Server gibt darauf wiederhin die Standartstatusmeldung.

```
}else if(bufCom == "mve"){ //Datei Umbenennen oder verschieben
        sendto(s, bufCom.c_str(), bufCom.length(), 0, (sockaddr *)&si_other, slen);
        recv(s, bufCode, 3, 0);
        bufferCode = bufCode;
        if(bufferCode == "200"){
                cout << "File name: ";
                getline(cin,buffer);
                sendto(s, buffer.c_str(), buffer.length(), 0, (sockaddr *)&si_other, slen);
                memset(bufCode, 0, sizeof(buf));
                recv(s, bufCode, 3, 0);
                bufferCode = bufCode;
                if(bufferCode == "200"){
                        cout << "Dir. or File Name: ";
                        getline(cin,buffer);
                        sendto(s, buffer.c_str(), buffer.length(), 0, (sockaddr *)&si_other, slen);
                        memset(bufCode, 0, sizeof(buf));
                        recv(s, bufCode, 3, 0);
                        bufferCode = bufCode;
                        if(bufferCode == "200")cout << "Move or Rename was Succesful";
                        else if(bufferCode == "400") cout << "Move or Rename was Unsuccessful";
                        else cout << "Unexpected Error";
                }else cout << "Connection Error";
        }else cout << "Connection Error";
}else if(bufCom == "dir"){ //Funktioniert nicht
```

Als letztes hat der Client die Möglichkeit mit “mdr“ (make Directory) und “del“ Ordner zu erstellen und Dateien zu löschen. Beim “mdr“ Befehl muss der Client den Namen des zu erstellenden Ordners angeben, der Server sendet daraufhin die Standartstatusmeldung ob das Erstellen Erfolgreich war.

```
else if(bufCom == "mdr"){ //Ein neues Verzeichnis
        sendto(s, bufCom.c_str(), bufCom.length(), 0, (sockaddr *)&si_other, slen); //Befehl senden
        recv(s, bufCode, 3, 0); //Auf antwort warten
        bufferCode = bufCode; //Von C-String in string umwandeln
        if(bufferCode == "200"){ //Falls Nachricht erfolgreich erhalten wurde
                cout << "Give directory a name: "; //Verzeichnisname eingeben
                getline(cin,buffer);
                if(buffer.find("..") == 0)buffer.clear(); //falls zwei Punkte gefunden werden wird der buffer gelöscht
                sendto(s, buffer.c_str(), buffer.length(), 0, (sockaddr *)&si_other, slen); //Verzeichnisname senden
                memset(bufCode, 0, sizeof(buf)); //buffer clearen
                recv(s, bufCode, 3, 0); //Auf Antwort hören
                bufferCode = bufCode; //Von C-String in string Umwandeln
                if(bufferCode == "200")cout << "Directory created"; //Erfolgreich
                else if(bufferCode == "400")cout << "Directory could not be created"; //Nicht erfolgreich
                else cout << "Unexpected error"; //Unerwartetes ist passiert
        }else cout << "Connection Error"; //Verbindung unerfolgreich
```

Für den “del“ Befehl muss der Client einen Dateinamen angeben, der daraufhin zum Server gesendet wird. Danach erhält der Client die Standartstatusmeldung ob das Löschen fehlerfrei funktioniert hat.

```
}else if(bufCom == "del"){ //Datei löschen
        sendto(s, bufCom.c_str(), bufCom.length(), 0, (sockaddr *)&si_other, slen);
        recv(s, bufCode, 3, 0);
        bufferCode = bufCode;
        if(bufferCode == "200"){
                cout << "File name: ";
                getline(cin,buffer);
                sendto(s, buffer.c_str(), buffer.length(), 0, (sockaddr *)&si_other, slen);
                memset(bufCode, 0, sizeof(buf));
                recv(s, bufCode, 3, 0);
                bufferCode = bufCode;
                if(bufferCode == "200")cout << "File deleted";
                else if (bufferCode == "404")cout << "File does not exist";
                else cout << "File could not be deleted due to an unknown error";
        }else cout << "Connection Error";
```

Kommunikationsprotokoll:

dir:

Client Anfrage zu Server – Fragt eine Liste aller Dateien im momentanen Ordner an

Server Antwort zu Client – Gibt Liste zurück

200: Daten konnten ohne Probleme zurückgegeben werden

400: Keine Daten gefunden

get:

Client Anfrage zu Server – Fragt Inhalt einer Datei an

Server Antwort an Client – Sendet Inhalt

200: Daten konnten ohne Probleme zurückgegeben werden

400: Allgemeiner Fehlercode

404: Datei konnte nicht gefunden werden

wrt:

Client zu Server – Schickt den Gewollten Inhalt

Server Antwort zu Client – Sendet Bestätigung

Client zu Server – Schickt den Gewollten Namen der Datei

Server Antwort zu Client – Sendet Status

200: Daten konnten ohne Probleme erstellt werden

400: Datei erstellt, konnte jedoch nicht gefüllt werden

402: Datei existiert schon

mdr:

Client Anfrage zu Server – Sendet eine Anfrage an den Server einen neuen Ordner im aktuellen Verzeichnis zu erstellen

200: Ordner wurde erfolgreich erstellt

400: Ordner konnte nicht erstellt werden

mve:

Client Anfrage zu Server – Sendet Namen der Datei

Server Antwort an Client – Schickt Bestätigung

Client zu Server – Sendet weiteren Namen von Datei oder Ordner

Server Antwort zu Client – Sendet Status

200: Datei konnte verschoben oder umbenannt werden

400: Datei konnte nicht verschoben oder umbenannt werden

del:

Client Anfrage zu Server – Sendet einen Dateinamen zum löschen

Server Antwort zu Client – Sendet Status

200: Datei wurde erfolgreich gelöscht

404: Datei konnte nicht gelöscht werden

cdr:

Client Anfrage zu Server – Sendet einen Verzeichnisnamen

Server Antwort zu Client - Sendet Status

200: Erfolgreich bewegt

400: Konnte nicht bewegen

idr:

Client Anfrage zu Server – Sendet Anfrage auf Rücksetzung des Verzeichnispfades.

200: Erfolgreich zurückgesetzt

400: Unerwarteter Fehler

Fazit:

Die Verwendung von Sockets ermöglicht es eine einfache Verbindung zu einem Server mit dem UDP-Protokoll zu erstellen. Am Anfang war es schwer sich mit Sockets und UDP auseinanderzusetzen, aber nach den ersten paar Tagen wurde es angenehmer.

Grundsätzlich sollte die Anwendung als Notizserver dienen, aber das Auslesen und zurücksenden von Dateiinhalten hat sich als äußerst schwer erwiesen. Es war uns nicht möglich die Auslesefunktionen “dir“ und “get“ zu implementieren. Wir haben es mit Streams versucht, aber hatten leider keinen Erfolg.

Wir sind leider nicht ganz zufrieden mit dem Endergebnis unserer Anwendung, da es uns nicht möglich war alle Anfragen/Responses zu implementieren die Ursprünglich geplant waren.