Vorlesung Betriebssystemarchitektur WS 2004/05

Aufgabenblatt 6 vom 07. Januar 2005

(Vorstellung der Lösungen bei den Tutoren bis zum 1. Februar 2005 möglich)

Aufgabe 6: (100 Punkte)

Schreiben Sie in der Programmiersprache C ein FTP-Server (RFC 959) Programm, das einen eingeschränkten Funktionsumfang realisiert. Aus dem FTP Protokoll sollen die Funktionalitäten folgender Kommandos implementiert werden: USER, PASS, TYPE, CWD, PWD, PORT, LIST, NLST, QUIT. Das FTP-Protokoll unterscheidet die Kontrollverbindung (standardmäßig Port 21) und Datenverbindungen (werden im aktiven Modus mit PORT ausgehandelt). Über die Kontrollverbindung werden Kommandos an den Server gesendet und die Antworten (Replies) vom Klienten empfangen. Neben Anworten auf Kommandos kann ein Server auch unaufgefordert Replies senden (z.B. Timeout, Server wird beendet). Für die Aufgabe sind nur die Replies zu den angegebenen Kommandos zu implementieren, sowie Fehlermeldungen für nicht unterstützte Kommandos. Ein FTP Kommando (siehe RFC 959, 5.3.1.) besteht aus einer Zeichenkette und von einem Leerzeichen getrennt optionale Parameter. Es wird nicht zwischen Groß- und Kleinschreibung unterschieden. Ein FTP Reply beginnt mit einem 3 stelligem Statuskode und mit einem Bindestrich (weitere Nachrichten mit dem selben Statuskode folgen) oder einem Leerzeichen (Ende der Nachricht) getrennt eine textuelle Beschreibung. Alle Kommandos und Replies sind mit CRLF (\r\n) abgeschlossen. Nach einem Verbindungsaufbau durch einen Klienten sendet ein FTP Server eine Begrüßung: z.B.

```
220-=========================================
220-  Welcome to the BSA practice 6 FTP server
220-=========================================
220 FTP Server lite ready
```

Im nächsten Schritt folgt das Authentifizieren mit USER und PASS. z.B.

```
USER ichbins
331 User name okay, need password.
PASS <password>
230 login ok
```

Ab diesem Punkt, Empfang von *230*, sind andere FTP Kommands erlaubt. Hat sich ein User nicht authentifiziert erhält er in der Regel *530*. Im dieser Aufgabe können sie auf jedes gültige USER-Kommando mit *331* und auf jedes gültige PASS Kommando mit *230* antworten. Auf das PWD Kommando liefert ein FTP Server die Informationen im Reply mit:

```
PWD
257 "/" is current directory
```

Das aktuelle Verzeichnis steht zwischen den Anführungszeichen. Es werden UNIX Pfad Konventionen benutzt. Das Server-Verzeichnis läßt sich mit CWD <Pfad> wechseln.

```
CWD /subfolder
250 CWD command successful.
```

Für ein Verzeichnislisting wie aus Aufgabe 5.2 wird zwischen Server und Klient eine Datenverbindung aufgebaut. Das FTP Protokoll unterscheidet 2 Verbindungsaufbauarten: 1. aktiv; Der Klient öffnet einen Server-Socket an den der FTP-Server connect aufruft. 2. passiv; Der Klient sendet das PASV Kommando und erhält als Antwort die Socket-Adresse an der der FTP-Server auf eine Verbindung wartet. In dieser Aufgabe sollen Sie mindestens den aktiven Verbindungsaufbau implementieren.

Um ein Verzeichnislisting zu erhalten, sind mehrere Schritte notwendig. Der FTP Klient kann sich das Datenformat mit dem TYPE Komando wünschen. Hier sollen Sie sich auf das ASCII-Format (`TYPE A`) beschränken (alle Zeilen werden mit CRLF beendet).

```
TYPE A
200 Type set to A.
```

Im nächsten Schritt muss der Klient eine Socket-Adresse für die Datenübertragung angeben. Dazu dient das PORT Kommando, welches als Parameter eine mit Kommas getrennte IPv4 Adressanteile und einen Port als 2 8 Bit Zahlen enthält. z.B.

```
PORT 127,0,0,1,16,51
200 Command okay.
```

Der Klient erwartet die nächste Datenübertragung an der Adresse 127.0.0.1 auf Port 4147. Der FTP-Server gibt mit 200 an das er diese Adresse für die nächste Datenübertragung wenden wird. Datenübertragungen werden z.B. mit `LIST, NLST, RETR` oder `STOR` eingeleitet. Der FTP-Server anwortet bei einem solchen Kommando mit dem Öffnen der Datenverbindung und nach erfolgter Übertragung mit dem schließen der Datenverbindung (close(sock)): z.B.

```
LIST
150 Opening ASCII mode data connection for /bin/ls.
226 Closing data connection.
```

Das im FTP Protokoll keine Größenangabe für das Empfangen von Daten vorgesehen ist, bedeutet das Schließen des Datensockets bzw. Reply 226 das Ende der Übertragung. Auf nicht unterstützte Kommandos antwortet man z.B. mit `502 Command not implemented.` Zum Beenden eine FTP-Sitzung sendet ein Klient QUIT, worauf ein FTP-Server mit 221 antwortet und die Verbindung beendet.

```
QUIT
221 Goodbye.
```

Das FTP-Server Programm soll 2 Übergabe-Parameter akzeptieren 1. Kontrollportnummer des Servers und 2. Das Root-Verzeichnis des Servers (CWD /)

```
aufg6.exe 5021  .
```

Startet den FTP-Server auf Port 5021 im lokalen Verzeichnis. Ports unterhalb von 1024, so auch 21 sind nur mit Administratorrechten verfügbar. Für die Implementierung hilfreiche Funktionen sind z.B. *strtok*, *strncasecmp(3)* bzw. *strnicmp, inet_addr, inet_ntoa, htons, htonl, chdir(2)/getcwd(2)* bzw. *Set/GetCurren tDirectory.*

Sie können das Programm unter Linux oder Windows entwickeln. Mit Linux ist die Implementierung einfacher, daher wird Linux empfohlen. Verpacken Sie alle Source-Code-Dateien inklusive eines Makefiles in eine ZIP-Datei und schicken Sie diese ZIP-Datei als Attachment an `bs@hpi.uni-potsdam.de` mit dem Betreff `AUFGABE=6 GRUPPE=`<Ihre Übungsgruppennummer> `OS=WIN32` oder `AUFGABE=6 GRUPPE=`<Ihre Übungsgruppennummer> `OS=LINUX` je nach Betriebssystem.
Die ausführbaren Dateien sollen `aufg6.exe` (Windows) bzw. `aufg6` (Linux) heißen.

Testen Sie Ihr Programm mit einem FTP-Klienten z.B. *ftp*. Das Programm *ftp* unter Windows XP benutzt statt LIST das Kommando NLST, auf welches Sie jedoch gleich reagieren können.