Versionsneutrale Socket-Programmierung

In dieser Aufgabe lernen Sie Anwendungen zu programmieren, die über das Internet kommunizieren. Hier steht praktisches Erarbeiten der Netzwerkprogrammierung in C im Vordergrund. Mit einem dialogorientierten Protokoll ("Request-Response") als Anwendung.

Aufgabe

Implementieren Sie in der Sprache C eine Anwendung zwischen Clients und einem multiclientfähigen Server unter Verwendung von TCP. Diese sollen das folgende vereinfachte Anwendungsprotokoll im http-Stil realisieren. Als Beispieldaten können Sie einfache Textdateien benutzen.

Client Kommando	Server Antwort
`List`	Liste aller verbundenen Clients der Form: `<Clienthostname>:<Clientport>`
`Files`	Liste aller Dateien im Server-Verzeichnis in der Form: `<Dateiname> <Datei-Attribute: last modified, size>`

`Get <dateiname>`	`<Datei-Attribute: last modified, size>`
	Inhalt von `<Dateiname>`
	`Übertragungszeit: <Zeit in Millisekunden>ms`

`Put <dateiname>`	Nach vollständigem Empfang und Speicherung der Datei:
	`OK <Serverhostname>`
	`<Benutzte Server-IP-Adresse>`
	`<Datum + Uhrzeit>`

`Quit`	Clientseitige Beendigung der Verbindung.

Die Anwendung soll **protokollunabhängig**, also sowohl für IPv4 als auch IPv6, lauffähig sein.

Entwicklungsumgebung

Zur Bearbeitung der Aufgabe können Sie sich auf `internet.link-lab.net` über SSH einloggen. Ihr Benutzername folgt dem Schema `rn-asX`, wobei X ihre AS Nummer aus der zweiten Aufgabe ist. Der Port ist der SSH Standardport 22. Nehmen Sie das Passwort aus der zweiten Aufgabe. Auf dem Server finden Sie ein Ubuntu Linux als Betriebssystem.

Auf dem Interface `bond0` sind eine öffentliche IPv4-Adresse (`141.22.28.244`) und einen öffentliche IPv6-Adresse (`2001:67c:254:b0b0::abcd:18`) konfiguriert, die Sie zum Testen nutzen können.

Als Texteditor gibt es `vim` (und `neovim`). Sie können das Dateisystem auch lokal mounten, um einen Editor Ihrer Wahl zu verwenden (sshfs). Mache Tools bringen sowas bereits mit, beispielsweise hat Visual Studio Code Support für Remote Development über SSH eingebaut.

Achten Sie darauf, dass sich die APIs auf Linux und Windows unterscheiden! Wenn Sie ihr Programm basierend auf der Windows API entwickeln, funktioniert es daher nicht zwangsweise auch auf dem Linux-Server. Obwohl macOS und Linux deutlich näher beieinander sind bei vielen Netzwerk spezifischen APIs, kann es auch hier Unterschiede in den Details geben.

Weiterhin gilt auch hier: *Do no harm.* Der Server wird von vielen Studenten genutzt und ist in erste Linie zum Absolvieren der Praktikumsaufgaben gedacht.

Tools: Linux, C Compiler (gcc, clang), man, gdb, tshark, Handbuch[1]

Hinweise zur Implementierung

Client-Details

Dem Client soll beim Aufruf der Server-Kontakt mitgegeben werden, also der DNS-Name oder die IPv4/6 Adressen sowie der Port des Servers.

Server-Details

Der Server bleibt single-threaded, multiplexed aber eingehende Verbindungen im Reactive Pattern mithilfe des `select` Posix-Calls. Initialisieren Sie hierzu im Zustand 'Listen' ein `fd_set`, rufen hierauf `select` auf und steuern das Annehmen eingehender Verbindungen (`accept`) über aktive Descriptoren im `fd_set`.

Versionsneutrale Programmierung

- Starten Sie zunächst mit den "Simple"-Socket-Programmen: `socket_client.c` und `server_src.c` aus der Vorlesung.
- Erweitern Sie diese um eine DNS-Namensabfrage mithilfe des Aufrufs `getaddrinfo` (s. MANPAGE). Dieser Call erzeugt die für den `socket`-Aufruf notwendigen Adressstrukturen in folgender Gestalt:
  - Er liefert einen Pointer auf eine verkettete Listen von `addrinfo` Adressstrukturen (s. MANPAGE).
  - Mithilfe einer Indirektion liefert `addrinfo` transparenten Zugriff auf die protokollabhängigen `sockaddr*`-Strukturen.
- Für die Socketerzeugung iterieren Sie über die Ergebnisliste und benutzen diejenige Adressstruktur, welche als erste funktioniert.
- Ergänzen Sie nun Ihr Programm für den Zugriff mithilfe von Namen und IP-Adressen aus der Kommandozeile. IP-Adressen können Sie mit `inet_pton` (s. MANPAGE)in die Netzwerkdarstellung, `getaddrinfo` kann mit der Eingabe einer IP-Adresse ebenfalls benutzt werden.

[1] Als praktisches Handbuch sei Beej's Guide to Network Programming empfohlen.

Testen

Testen Sie, dass Ihr Server sowohl über IPv4 als auch über IPv6 erreichbar ist. Stellen Sie außerdem sicher, dass Ihr Client einen Server sowohl über den DNS Namen als auch über die direkte Adresse findet.

Da das Protokoll rein textbasiert ist, sollte es problemlos mit den Programmen anderer Gruppen zusammen funktionieren.

Abgabge

Reichen sie wohldokumentierten C-Code zusammen mit einem Protokoll der Praxistests ein.