Hochschule RheinMain
Studiengang Medieninformatik
Sommersemester 2018
Prof. Dr. Wolfgang Weitz

Betriebssysteme und Rechnerarchitektur

Übungsblatt[-1] (ergänzt, siehe blaues Ende)

18.06.2018
(Tag der *Lord-Robert-Stephenson-Smyth-Baden-Powell*-Durchführung)

Aufgabe:
Zum Verschicken bzw. Empfangen von E-Mails wird weithin das *Simple Mail Transfer Protocol* eingesetzt. Der Client (der eine Mail loswerden möchte) öffnet eine TCP-Verbindung auf den entsprechenden Port des Mailservers und die beiden spielen SMTP miteinander. Rückantworten des Servers beginnen immer mit einem dreistelligen Statuscode, gefolgt von einem Leerzeichen und einem frei wählbaren, erläuternden Text. Wie bei POP3 enden Zeilen immer mit „CRLF“ („`\r\n`“). Hier ein Vorbild für solch einen Dialog nach Verbindungsaufbau durch den Client:

Server	*220 meinmailserver.de*	220 = Begrüßung des Clients durch Server
Client	**HELO** clientname.de	Client meldet sich mit seinem Rechnernamen
Server	*250 Ok*	250 = Aktion ok
Client	**MAIL FROM:**<joghurta@brautrauwem.de>	Mail-Adresse des Absenders in < >
Server	*250 Ok*	Absenderadresse akzeptiert (oft nur grobe Prüfung)
Client	**RCPT TO:**<joendhard@localhost>	Adresse des Empfängers
Server	*250 Ok*	Server ist bereit, Mail diese Adresse anzunehmen
Client	**DATA**	Client will Mail übertragen
Server	*354 End data with <CR><LF>.<CR><LF>*	*354* = „start mail input“
Client:		Client überträgt Mailinhalt (mit Headern)

```
From: "Brauerei Taucha" <joghurta@brautrauwem.de>
To: "Joendhard Biffel" <joendhard@localhost>
Subject: Wie geht's
Date: Thu, 7 Jun 2018 14:39:00 +0200

Hallo Joendhard, wie geht's?

Viele Gruesse, Joghurta
```

.		Einzelner Punkt = Ende der Maildaten
Server	*250 Ok*	250 = Aktion ok
Client	**QUIT**	Client will Schluss machen
Server	*221 Bye*	221 = Server schließt Verbindung

Wenn der Client ein **unbekanntes Kommando** schickt, antwortet der Server mit einem 5xx-Code, z.B. 500 für unerkanntes oder syntaktisch falsches Kommando oder 550 für ein nicht ausführbares Kommando (z.B. unbekannte Empfängeradresse). Codes der Form 2xx sind immer ein gutes Zeichen. Eine vollständige Liste finden Sie in RFC5321 in Abschnitt 5.2.2

Sie können vereinfachend davon ausgehen, dass die Kommandos in dieser Reihenfolge eingehen, und können Ihren „Dialog-Automaten“ leicht so konfigurieren, dass er `HELO`, `MAIL FROM`, `RCPT TO` annimmt und nach `DATA` in einen „Maileinlesen-Folgezustand“ übergeht. Es kann sein, dass zu Beginn oder zwischendurch unbekannte Kommandos geschickt werden, was Sie einfach mit einem Fehlercode 502 (command not implemented) oder 550 (nicht ausführbar) beantworten, oder dass notwendige Schritte (z.B. RCPT TO) fehlen.

Vereinfachend nehmen wir auch an, dass nach einer Maileinlieferung ein `QUIT`-Kommando folgt, das Sie zwar verarbeiten müssen (also eine Antwort schicken und die Verbindung schließen), aber Sie können davon ausgehen, dass nicht anstelle des `QUIT` weitere Mails folgen (was normalerweise möglich wäre). Wir beschränken uns also auf eine Mail je SMTP-Verbindung, weil das nicht zuletzt aufgrund des Umstands, dass es weniger kompliziert ist, einfacher ist, wenn auch nicht viel, so dass man es auch anders machen könnte, was wir aber hier nicht müssen.

Bei **Eingang einer neuen Mail** identifizieren Sie also zunächst die **zugehörige Mailbox** (anhand des

RCPT TO Kommandos, siehe Abschnitt „Konfiguration“ weiter unten) und hängen Sie dann zuerst a) eine neue **Trennzeile** der bekannten Form „From / Absender (aus MAIL FROM) / Zeitstempel, z.B.

From joghurta@brautrauwem.de *Sun May 27 19:05:07 2018*

und danach einfach b) den nach „DATA“ empfangenen **Mailinhalt** an die Mailboxdatei an, gefolgt von c) einer den neuen Mailbox-Eintrag abschließenden **Leerzeile**. Danach sollten Sie über Ihren POP3-Server die Mailbox erneut öffnen und Ihre neue Mail sehen / abrufen können. Den aktuellen Zeitstempel als String erhalten Sie leicht durch die Kombination der Bibliotheksfunktionen `time()` (liefert Sekunden seit 1.1.1970) und `ctime()` (wandelt Sekunden in Datumsstring um).

Integration in den POP3-Server
Wir wollen *ein* Serverprogramm haben, das sowohl POP3- als auch SMTP-Verbindungen akzeptiert. Verwenden Sie bitte `select()`, um „gleichzeitig“ auf Verbindungaufbau-Wünsche auf dem POP3- und dem SMTP-Socket zu lauern. Es soll also nur eine Haupt“`accept`“-Schleife geben, die sowohl für POP3- als auch für SMTP-Verbindungen zuständig ist. Es soll keine getrennten POP3- und SMTP-Hauptschleifen in separaten Prozessen/Threads geben - die Abarbeitung einer bereits mit `accept()` entgegengenommenen Verbindung dagegen schon, POP3 in einem eigenen Subprozess wie gehabt, SMTP in einem pThread. Achten Sie bitte dabei darauf, dass immer nur einer gleichzeitig auf dieselbe Mailbox schreiben darf, sonst gibt's Durcheinander und (daraus folgend) Unmus.

Wie bei POP3 können Sie auch hier Ihren Server zunächst einmal interaktiv mit „`telnet localhost` *`smtpport`*“ ansteuern und den SMTP-Dialog von Hand führen, um zu sehen, ob der Server korrekt auf den Normalablauf, aber auch auf Fehler (z.B. Syntaxfehler, unbekannte Mailadresse bei `MAIL FROM` usw.) reagiert.

Konfiguration:
In Ihrer Konfigurationsdatenbank haben Sie bereits Einträge für POP3-User nach dem Schema „`key=`*username*, `cat=`mailbox, `value=`*pfad_zur_mailboxdatei*“ abgelegt. Nun verwenden wir einfach die Konfigurationsdatenbank, um eingehende Mails anhand der Ziel-Mailadresse (SMTP-Kommando „RCPT TO“) auf den zugehörigen Nutzer und damit seine Mailboxdatei abzubilden. Auf diese Weise können wir sogar Mails für verschiedene Mail-Adressen derselben Mailbox zuordnen.

Legen Sie bitte zu jeder **Mailadresse**, die Sie unterstützen möchten, einen Konfigurationseintrag „`key=`*mailadresse*, `cat=`smtp, `value=`*username*“ an, wobei „*username*“ dem `key` des zugehörigen POP3-mailbox-Eintrags entspricht (s.o.), also z.B. „key=j.biffel@mymaildings.de, cat=smtp, value=“joendhard“). Sie können also leicht zu einer erhaltenen Mailadresse feststellen, ob Sie diese „kennen (also die Mail annehmen wollen), und dann über den zugeordneten *usernamen* die Mailbox-Datei uplookieren. In der bisherigen Headerdatei „`database.h`“ ist die Konstante `DB_KEYLEN` für die maximale `key`-Länge auf 20 gesetzt, was für phantasievolle Mailadressen etwas kurz ist. Sie sollten sie daher auf 80 hochsetzen (und sich freuen, dass Sie in Ihrem Programm konsistent mit den Konstanten gearbeitet haben, so dass keine weiteren Anpassungen nötig sind).

Ebenso wie bei POP3 sollte auch hier die Netzwerkschnittstelle und der Port, auf dem der SMTP-Service läuft, konfigurierbar sein. Default sei „alle Schnittstellen“ und „Port 8025“, falls der jeweilige Eintrag nicht in der Datenbank vorhanden ist. Verwenden Sie bitte „`key=`port, `cat=`smtp, `value=`*portnummer*“ für die Port- und „`key=`host,`cat=`smtp,`value=`hostaddr“ für die Adresse hostaddr des zu bedienenden Interfaces in der üblichen, gepunkteten IPv4-Adressnotation.

Wer noch Zeit und Lust hat, kann den SMTP-Server so erweitern, dass er auf Konfigurationseinträge „`key=`*domainname*,`cat=`forward,`value=`*zielserver:portnr*“ reagiert, indem er eingehende Mails an die Domäne *domainname* automatisch an den SMTP-Server *zielserver* (mit Zielport *portnr*) weiterleitet. So können Sie miteinander eine eigene Mail-Routing-Infrastruktur aufbauen, bei der Nachrichten konfigurierbar ggf. über mehrere Stationen weitergeleitet werden.

The End.
Abgabe aller Sourcefiles des POP3/SMTP-Servers incl. Makefile im read.MI mit nachfolgender Vorführung des abgegebenen Codes bis Vorlesungsende (bis 06.07.18, gerne früher), Termin-vereinbarung bis 04.07. möglich. Die erfolgreiche Vorführung mit Erläuterung des Codes ist Voraussetzung zum Bestehen des Praktikums.