Dr. Andreas Borchert
Christian Ehrhardt

Systemnahe Software (SS 2003)

Abgabetermin 8.7.2003

Mails

In diesem Blatt soll ein Programm geschrieben werden, mit dem eine Mail an alle Teilnehmer der eigenen Tutoriumsgruppe geschickt werden kann.

Verbindung

Dazu ist es notwendig, sich mit dem Mailserver zu unterhalten. In späteren Blättern wird das direkt über Sockets funktionieren, für dieses Blatt beauftragen wir das Programm `nc` (auf manchen Systemen auch `netcat`) mit der Verbindungsaufnahme. `nc` bekommt zwei Argumente, den Namen des Rechners, auf dem der Server läuft und den zu verwendenden Port. Für dieses Blatt kann als Servername `theseus.mathematik.uni-ulm.de` verwendet werden. Der Port für Mail ist `25`.
Wenn man `nc` mit diesen Parametern aufruft, dann schickt `nc` seine Standardeingabe an den Mailserver weiter und die Antworten des Mailservers erscheinen auf der Standardausgabe. Auf diese Weise ist es also (mit Hilfe von pipes) für ein Programm möglich, sich mit dem Mailserver zu unterhalten.

Unterhalten mit dem Mailserver

Prinzipiell läuft die Unterhaltung mit dem Mailserver Zeilenorientiert ab. Jede Zeile muß durch die Zeichenfolge \r\n abgeschlossen werden. Es ist Aufgabe Eures Programms das sicherzustellen, selbst dann, wenn eine eingegebene Zeile nur mit \n beendet wurde.
Die Unterhaltung mit dem Mailserver beginnt damit, daß der Mailserver eine Begrüßung verschickt. Diese Begrüßung ist genauso aufgebaut, wie die Antwort auf ein Kommando (siehe unten). Anschließend läuft die Kommunikation so ab, daß der Client ein Kommando sendet. Der Server beantwortet dieses Kommando mit einer Antwort. Je nach Kommando und Antwort kann es außerdem noch sein, daß der Server oder der Client nach der Antwort zusätzliche Daten verschickt.

Wichtige Kommandos

Ein Kommando besteht immer aus einem oder mehreren Worten (unten fett gedruckt). Manche Kommandos haben auch Argumente, die durch Leerzeichen getrennt hinter dem Kommando kommen (unten kursiv gedruckt). Beendet wird das Kommando durch \r \n.

- **HELO** *rechnername*: Mit diesem Kommando begrüßt der Client den Server. *rechnername* muß der Name des lokalen Rechners sein. In dieser Aufgabe kann dieser String noch als Konstante im Programm definiert sein.
- **MAIL From:** *user@host*: Dieses Kommando gibt an, daß *user@host* die Mailadresse des Absenders ist. Sollte es später bei der Zustellung der Mail zu Problemen kommen, wird eine Fehlermeldung an diese Adresse geschickt. Dieses Kommando sollte für jede Mail nur ein einziges Mal im Dialog mit dem Server auftauchen.
- **RCPT To:** *user@host*: Dieses Kommando gibt an, daß *user@host* einer der Empfänger der Mail ist. Dieses Kommando kann mehrere Male für eine einzige Mail auftauchen.
- **DATA** Dieses Kommando hat keine Argumente. Es sagt dem Server, daß anschließend der Inhalt der Mail folgt. Nach der Antwort des Server sollte der Client den Inhalt der Mail an den Server schicken. Der Client zeigt dem Server durch eine Zeile, die nur einen Punkt enthält an, daß die Mail zu Ende ist. Sowohl das **DATA** Kommando alsauch der Punkt wird vom Server durch eine einzelne Zeile beantwortet (siehe unten).
- **QUIT** Dieses Kommando teilt dem Server mit, daß die Kommunikation beendet werden soll.

Diese Kommandos sollten in der Regel in dieser Reihenfolge vom Client geschickt werden. Einzige Ausnahme: **QUIT** kann immer geschickt werden. Wenn **QUIT** vorzeitig verschickt wird, wird die Verbindung beendet ohne daß die Mail verschickt wird.

Antworten des Servers

Die Antwort des Servers besteht immer aus einer Zeile. Diese Zeile wiederum besteht aus drei Ziffern gefolgt von einem kurzen Text. Die Ziffern geben an, ob das Kommando erfolgreich war oder was für ein Fehler aufgetreten sein könnte. Der Text beschreibt die Antwort in einer für Menschen verständlichen Form. Für dieses Blatt ist nur die erste Ziffer des Antwortcodes interessant. Folgende Klassen von Antwortcodes gibt es:

- **1xy** Sollte in unserer Andwendung nicht vorkommen. Falls es doch passiert, sollte diese Fehlermeldung mit dem **QUIT** Kommando beantwortet werden.
- **2xy** Diese Antwortcodes zeigen an, daß das Kommando erfolgreich war.
- **3xy** Diese Antwortcodes zeigen ebenfalls an, daß das Kommando akzeptiert wurde, es fehlen aber noch weitere Daten. Für den Client heißt das, daß er weitermachen kann wie geplant. Ein solcher Antwortcode folgt zum Beispiel direkt auf ein erfolgreiches **DATA** Kommando.

- **4xy** Diese Antwortcodes zeigen einen temporären Fehler an. Temporär bedeutet hier, daß das Kommando zwar fehlschlug, daß es aber begründete Hoffnung gibt, daß das Kommando zu einer späteren Zeitpunkt erfolgreich sein könnten. Ein solcher Fehler kann zum Beispiel auftreten, wenn der Server im Moment nicht genügend Resourcen hat, um das Kommando auszuführen.
- **5xy** Diese Antwortcodes zeigen einen permanenten Fehler an. Ein möglicher solcher Fehler wäre zum Beispiel, wenn ein Kommando zu diesem Zeitpunkt noch nicht zulässig ist, oder wenn der Server für die Domain einer Empfängeradresse nicht zuständig ist.

In der Regel wird der Server nur ganz zu Beginn und nach einem Kommando eine solche Antwort schicken (siehe aber das **DATA** Kommando). Zusätzlich sind manche **5xy** und **4xy** Antworten auch ohne ein Kommando (zum Beispiel mitten während dem Senden des Mailinhalts) möglich.

Der Inhalt einer Mail

Eine Mail besteht aus zwei Teilen, dem sogenannten Header und dem eigentlichen Text. Der Header wird durch eine leere Zeile beendet. Der Header einer von Euch verschickten Mail sollte mindestens die folgenden Zeilen enthalten (fettgedruckter Text ist wörtlich zu nehmen, kursiver Text beschreibt durch was der Text ersetzt werden soll):

- **Subject: Wichtige Nachricht an die Tutoriumsgruppe**
- **From:** *Mailadresse des Absenders*
- **To: mein.tutorium@mathematik.uni-ulm.de**
- *Die Leerzeile, die den Header vom Text trennt*

Zu beachten ist, daß die Einträge in den From:- und To:-Zeilen nicht unbedingt etwas mit den beim **MAIL From:** und **RCPT To:** Kommando angegebenen Adressen zu tun haben müssen. Es ist Aufgabe Eures Programms sicherzustellen, daß die Mail diese Headerzeilen enthält und daß Ihr Inhalt richtig ist.

Zusammenfassung

Euer Programm soll den Text einer Mail von der Standardeingabe einlesen und diesen Text als Mail mit Header an sämtliche Mitglieder der eigenen Tutoriumsgruppe verschicken. Dazu wird eine Verbindung mit dem Mailserver aufgebaut. Euer Programm ist dafür zuständig, sich mit dem Mailserver zu unterhalten, einen richtigen Header einzufügen und eventuell ein `\r` am Zeilenende hinzuzufügen, wenn die Zeile durch ein alleinstehendes `\n` beendet wird. Außerdem soll das Programm die Antwortcodes des Server überprüfen und zum Schluß angeben, ob die Mail wirklich erfolgreich beim Server abgeliefert wurde.

Hinweise

- Wie wir oben bereits gesehen haben, muß der Text, der in der From:- oder To:-Zeile einer Mail steht nichts mit dem tatsächlichen Absender oder dem tatsächlichen Empfänger zu tun haben. Dies könnte verwendet werden, um recht viel Verwirrung zu stiften. Allerdings verstößt das gegen unsere Nutzungsbedingungen und es gibt durchaus Möglichkeiten, auch in so einem Fall den echten Absender einer Mail ausfindig zu machen.
- Wenn auf den pipes mit `FILE *` Pointern gearbeitet wird, muß mit Hilfe von fflush sichergestellt werden, daß der Server das Kommando auch tatsächlich erhält nachdem es in die pipe geschrieben wurde.
- Im Zusammenhang mit Netzwerken kann es im Prinzip zu beliebig langen Zeilen kommen. Euer Programm sollte das berücksichtigen.
- User Mailserver reagiert unter Umständen mit einer Fehlermeldung, wenn der Rechnername beim **HELO**-Kommando nicht zu dem Rechner paßt, auf dem der Client läuft. Die Rechner in unseren Pools sind davon aber ausgenommen.