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

Betriebssysteme und Rechnerarchitektur

Übungsblatt sechs

04.06.2018
(Hessischer Tag der Durchführung)

Aufgabe:
Sie haben im Rahmen der Bearbeitung des letztwöchigen Übungsblatts eine funktionierende Funktion „`int process_pop3(int infd, int outfd)`“ erschaffen, welche auf einer vorgegebenen mbox-Datei die POP3-Operationen zum Einloggen und zur Inhaltsabfrage (Übersicht und Komplettabruf einzelner Nachrichten) durchführt.

Der Kern-Kommandoumfang von POP3 sieht noch wenige weitere Operationen vor, die Sie nun noch hinzufügen sollten. Referenz ist noch immer RFC1939 (https://tools.ietf.org/html/rfc1939). Auch diese Kommandos sollen schreibweisenunabhängig (Groß-/Kleinschrift egal) akzeptiert werden.

`NOOP`
„no operation“ - tut nichts (und bestätigt das trotzdem positiv mit +OK)

`DELE` *n*
merkt Nachricht *n* zum Löschen vor (löscht nicht direkt) - Sie erinnern sich an das dafür vorgesehene Feld `del_flag` in der `FileIndexEntry`-Struktur. Für alle nachfolgenden Kommandos wird jedoch so getan, als wäre Nachricht n schon weg. Jeder nachfolgende Zugriff-Versuch auf diese Nachricht wäre ein Fehler - mehrfaches „`dele`“ für dieselbe Nachricht wäre also ein Fehler (und wäre mit Rückantwort -ERR zu ahnden), auch taucht Nachricht „n“ nicht mehr in der Ausgabe von `list` auf (die anderen Nachrichten behalten jedoch während dieser POP3-Sitzung noch ihre Nummern - kein „Aufrücken“ - die `list`-Ausgabe hat dann hinsichtlich Numerierung halt eine „Lücke“), zählt bei `stat` nicht mehr zur Mailbox-Größe (Anzahl/Octet-Gesamtgröße) hinzu usw.

`RSET`
„reset“ setzt das Lösch-Flag *aller* Einträge wieder auf zurück. Zuvor mit „`dele`“ zum Löschen vorgemerkte Einträge sind also wieder mit allen Rechten und Pflichten zurück, zählen also z.B. wieder zum `stat`-Umfang, treten in `list` auf und sind mit `retr` abrufbar.

`QUIT`
beendet die Sitzung (und wird später die Netzwerk-Verbindung schließen, sobald wir eine haben). Zudem werden alle zum Löschen vorgemerkten Einträge tatsächlich aus der Mailbox-Datei entfernt (vgl. dazu Ihre Funktion `fi_compactify()` aus Ihrem `fileindex`-Modul).

Aufgabe:
Sie haben's schon erwartet - jetzt wird's netzig. Bitte schreiben Sie sich ein Modul „`mailserver.c`“, welches unter Verwendung Ihres vorhandenen Baukastens auf eine eingehende TCP-Verbindung lauert und, sobald eine eingeht, den zugehörigen Socket-Deskriptor in Ihr process_pop3() hineinreicht, und zwar zweimal (als Input- und als Output-Deskriptor, denn Sockets funktionieren ja bidirektional). Sobald die Verbindung vom Client abgebaut wird (z.B. unhöflich durch Schließen der Verbindung) oder der Server das „`QUIT`“-Kommando bekommen hat (woraufhin er sich mit einem letzten +0K verabschiedet und selbst die Verbindung schließt),

warten Sie auf die nächste Verbindung.

Natürlich verwenden wir auch hier unsere coole Konfigurationsdatenbank. Legen Sie bitte (als String) die Portnummer, auf der Ihr Server läuft unter `key="port"` und `cat="pop3"`, und die IP-Adresse, an die Sie sich binden möchten (Ihr Rechner könnte ja mehrere Interfaces mit unterschiedlichen Adresse haben), unter `key="host"` und `cat="pop3"`. Falls kein passender Eintrag in der Datenbank vorhanden ist, soll Ihr Server als Default-Interfaces das Loopback-Interface Ihres Rechners und als Default-Port 8110 (da nicht privilegiert) verwenden. Nützlich sein könnte dazu auch die besprochene Umwandlungs-Funktion `int inet_aton(const char *cp, struct in_addr *inp)`.

Testen Sie Ihren POP3-Server bitte, indem Sie ihn in einer Shell starten und von einer andern aus mithilfe des „`telnet`"-Kommandos eine TCP-Verbindung aufbauen, so dass Sie sich direkt via Netzwerk mit Ihrem Server unterhalten können (z.B. „`telnet localhost 8110`").

Wenn das grundsätzlich geht, können Sie auch gegen die Python3-`poplib` automatisiert testen. Schreiben Sie sich dazu bitte ein klitzekleines Python3-Skript, um Ihren Server zu prüfen:

```
import poplib
pop = poplib.POP3("localhost", 8110)
pop.set_debuglevel(1)
pop.user("joendhard")
pop.pass_("biffel")  # man beachte den "_", denn "pass" allein ist ein Python-Keyword
print(pop.stat(), pop.list(), pop.retr(1))
pop.close()
```

Aufgabe:
Momentan kann unser Server nur eine POP3-Verbindung auf einmal bedienen. Die nächsten Usenden müsste also warten, bis der Servierende wieder „frei" ist, egal, wie lange die Session dauert. Das wird als uncool wahrgenommen und ist natürlich nicht verkaufsförderlich.

Bitte erweitern Sie Ihren Server zu einem „*forking server*". Bei Entgegennahme einer neuen Verbindung soll also ein Subprozess exklusiv zum Handling dieser Verbindung erzeugt werden. Dadurch wird Ihre Haupt-"`accept()`"-Schleife sofort wieder frei und kann auf die nächste Verbindung warten. Denken Sie bitte daran, nach dem `fork()` im Kindprozess den „accept"-Socket des Vaters und im Vaterprozess den von accept gelieferten „Kind-Socket" zu schließen.

Nun könnte irgendeine Pappnase auf die Idee kommen, sich parallel mehrfach für dieselbe Mailbox einzuloggen. Das ergäbe Ummus (denken Sie allein an abweichend zum Löschen vorgemerkte Nachrichten usw.). Bitte erinnern Sie sich des simplen „Sperrdatei"-Tricks, den wir im Zusammenhang mit „`open()`" im Datei-Kapitel angesprochen haben, und sehen Sie eine Lösung vor, so dass immer nur max ein Prozess auf gleichzeitig je Mailbox zugreifen kann (Fehler bei „`pass`", wenn mbox „im Gebrauch" ist). Nutzen Sie dazu eine Sperrdatei, deren Name sich von dem der Mailbox ableitet (also mbox-Datei „`joendhard`" -> Sperrdatei „`joendhard.lock`").

Nun könnte es sein, dass Ihr Server abstürzt und die Lock-Datei stehen bleibt, wodurch die Mailbox ggf. unzugreifbar wird. Das wäre schade. Eine gängige Lösung ist, die Prozessnummer der sperrenden Prozesses als einzigen Inhalt in die Sperrdatei zu schreiben. Wenn ein anderer Prozess die Sperrdatei antrifft, gibt er nicht sofort auf, sondern liest die Sperrdatei aus und prüft, ob der Prozess mit der enthaltenden PID überhaupt läuft. Falls nein, ist die Sperrdatei veraltet und der Prozess kann sie überschreiben, um sich selbst die Rechte an der Mailbox zu sichern. Das macht's deutlich robuster.

Bitte installieren Sie nun einen geeigneten Signalhandler, so dass eine eventuell gerade bestehende Netzwerkverbindung (und deren Handler-Prozesse) bei Abbruch des Servers mit strg-c (also `SIGINT`) geordnet beendet werden, bevor der Server sich beendet (der Vaterprozess sollte also einen Überblick über seine Kindprozesse behalten). Laufende Subprozesse sollen geordnet herunterfahren (Verbindung schließen, Sperrdatei entfernen etc), bevor der Vaterprozess endet. Natürlich dürfen in dieser Phase keine neuen Verbindungen angenommen werden.