Dr. M. Grabert
Johannes Mayer

Systemnahe Software (SS 2001)

Abgabetermin: 9. Juli 2001

12 Jetzt aber mit Sockets! (5 Punkte)

Nachdem Sie nun schon in der Vorlesung kennengelernt haben, wie man mit Sockets umgeht, können Sie auch das letzte Übungsblatt ohne den Umweg über nc oder telnet lösen. Verändern Sie also Ihre Lösung oder die Beispiellösung zum letzten Übungsblatt so, dass statt des „Forkens“ mit darauffolgendem exec-Aufruf und der Kommunikation über zwei Pipes eine Socket-Verbindung zu dem POP3-Server hergestellt wird.

13 Hol' die Mails, Browser! (15 Punkte)

Wow, jetzt haben Sie also einen eigenen POP3-Client, mit dem Sie die Header (Absender, Betreff und Datum) der letzten Mails abholen können. Das ist doch was, oder?! Naja, aber besonders schön sieht die Ausgabe noch nicht aus. Wie wär's denn, wenn die Ausgabe im Browser erfolgen würde. Das ist doch 'mal eine gute Idee. Und das geht zudem noch verblüffend einfach – wie Sie ja in der Vorlesung gesehen haben.
Also nichts wie ans Werk. Schreiben Sie also einen Server, der auf Port ...[1] lauscht. Sobald sich ein Client mit Ihm verbindet, liest er eine POST-Anfrage gemäß HTTP-Protokoll. Eine solche Anfrage besteht aus einem Header, dessen letzte Zeile eine Leerzeile ist (nur mit „`\r`“ bzw. „`\n`“ am Ende). Danach kommen genau so viele Zeichen, wie im Header angegeben ist (in der Header-Zeile, die mit `Content-Length:` beginnt). Nach diesen Zeichen kommt kein Zeilentrenner mehr! Diese zuletzt gelesenen Zeichen enthalten die Variablen. Die einzelnen Variablen sind voneinander durch `&` getrennt. Zwischen dem Name und dem Wert einer Variablen steht ein `=`-Zeichen. Die Variable `host` enthält den Namen des POP3-Servers. In den Variablen `login` und `pass` sind die Zugangsdaten des POP3-Servers enthalten. (Die Reihenfolge der Variablen kann beliebig sein!)
Beispiel für eine POST-Anfrage nach dem HTTP-Protokoll:

```
POST / HTTP/1.0
User-Agent: Mozilla/4.7 [en] (X11; U; SunOS 5.8 sun4u; Nav)
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
```

[1] Portnummer > 1024; bedenken Sie bei einem evtl. Fehler, dass dieser bereits verwendet wird und versuchen Sie es mit einem anderen.

```
Accept-Encoding: gzip
Accept-Language: de, en
Accept-Charset: iso-8859-1,*,utf-8
Content-Type: application/x-www-form-urlencoded
Content-Length: 32
Via: 1.0 suckit.rz.uni-ulm.de:3128 (Squid/2.3.STABLE2)
X-Forwarded-For: 134.60.66.5
Host: thales.mathematik.uni-ulm.de:11023
Cache-Control: max-age=259200

host=turing&login=test&pass=test
```

Nun sollen von Ihrem POP3-Server die Header der neuesten Mails von dem angegebenen POP3-Account abgeholt und als HTML-Dokument ausgegeben werden. Hierzu können Sie Ihre Lösung für die vorige Aufgabe als Unterprogramm verwenden. Sie müssen es „nur“ so abändern, dass es ein HTML-Dokument produziert. Der Client, welcher Ihrem Server die POST-Anfrage geschickt hat, erwartet nach dem HTTP-Protokoll aber nicht nur das HTML-Dokument, sondern auch eine kurze Antwort, welche aus einem Header und – getrennt durch eine Leerzeile – einem HTML-Dokument besteht.
Beispiel für eine (minimale) Antwort auf eine POST-Anfrage:

```
HTTP/1.1 200 OK
Content-Type: text/html

<HTML>
   <BODY>
   ... die Header der neuesten Mails ...
   </BODY>
</HTML>
```

Nachdem Ihr Server die POST-Anfrage mit seiner Antwort quittiert hat, soll er die nächste Anfrage entgegennehmen.
Tipps: Um komfortabel eine POST-Anfrage an Ihren Server schicken zu können, benötigen sie ein *HTML-Formular*. Bei diesem müssen sie nur noch
`action="http://thales.mathematik.uni-ulm.de:11023/"`
durch den Rechner, auf dem Ihr Server läuft, und den Port, den Ihr Server verwendet, ersetzen. *Hier* finden Sie außerdem ein kleines HTML-Dokument, das sie als Grundlage für die Ausgabe der Mail-Header verwenden können. Wenn Sie die Ausgabe des Servers aufwendiger gestalten wollen, können Sie den *SELFHTML-Kurs* ja 'mal anschauen.
Die Werte der Variablen, welche Sie via POST erhalten, sind „URL-codiert“. Zu deren Decodierung und zur Ausgabe von Text als HTML erhalten Sie zwei *C-Funktionen*. Zum Zerlegen des Strings, der die Variablen (der POST-Anfrage) enthält, können Sei die Funktion `strtok` einsetzen. Bei Verwendung von File-Pointern sollten sie nach jedem Schreiben den Puffer mittels `fflush()` leeren. Es ist empfehlenswert, die File-Deskriptoren mittels `fdopen()` in File-Pointer zu konvertieren, und dann mit diesen zu arbeiten.

Viel Spass und Erfolg!