Unix I – Einführung in das Unix Betriebssystem

Stephan Thesing

Netzwerke

- WWW
- Mail
- SSH

WWW

Wahrscheinlich *der* Grund, warum das Internet so enorm gewachsen ist.

Idee: Informationen werden global verfügbar gemacht, durch *Links* verknüpft.

Viele Datentypen können eingebunden werden (über MIME Schema).

Zugriff erfolgt über das HTTP Protokoll (RFCs 1945, 2068).

Benennung von Komponenten erfolgt über URIs (URLs, RFCs 1738, 2396)

Durch ansprechende graphische Darstellung sehr PR fördernd.

WWW = Gesamtheit aller verzeigerten Informationen.

Hauptinformation auf *HTML Seiten* (`text/html`)

Diese Seiten beinhalten

- Header-Information (Titel, Meta-Informationen)
- Body: Text und *Tag*s. Tags dienen zur Auszeichnung bestimmter Strukturen

Durch Tags werden etwa Attribute (Fettschrift, etc.) als auch Strukturen (Tabellen, etc.) und Links ausgezeichnet.

Format von HTML vom W3C festgelegt (nette Fiktion)

In HTML Seiten können andere Objekte direkt eingebettet werden.

Der Standort dieser Objekte wird ebenfalls über URIs angegeben.

Am wichtigsten aber sind die Links. Diese führen zu anderen Seiten oder Objekten.

Der *Browser* ist für die Darstellung der HTML Seiten und das Handling der Objekte zuständig.

Der Server liefert nur Daten anhand von HTTP Anfragen zurück.

Wichtig: HTTP ist *state less*: Schränkt Interaktion ein.

Wie sehen URIs aus?

(Wir betrachten nur URLs):

Protokoll:*Zusatzdaten*

Die wichtigste Form von URLs sind die Internet URLs:

Protokoll://*User*:*Passwort*@*Host*:*Port*/*Pfad*

Beispiele:

```
http://www.cs.uni-sb.de
ftp://ftp:thesing%40cs.uni-sb.de@ftp.cs.uni-sb.de/rfc/rfc-index.gz
mailto:thesing@cs.uni-sb.de
https://commerce.acme.com
telnet://thesing@kap01.studcs.uni-sb.de
```


Der Browser interpretiert URLs und holt die Daten dann mit dem richtigen Protokoll.

Wie weiss er aber, was er mit der Antwort machen soll?

Die Antwort enthält Information über den Typ der Daten.

Diese ist über MIME (Multipurpose Internet Mail Extensions, RFC 2045-2049) kodiert (HTTP-Protokoll).

Der Typ gibt dabei eine Kategorie und einen Untertyp an, z.B.

```
text/html
image/gif
application/postscript
video/quicktime
```

Einige dieser Typen kann der Browser selbst verarbeiten.

Für andere kann man ihm *Helper Applications* angeben.

Das sind Programme, die mit diesen Daten als Argument aufgerufen werden (z.B. Postscript Viewer).

In den Dateien `.mime.types` kann man auch eine automatische Assoziation von Dateiendungen zu MIME Typen angeben (etwa für FTP Protokoll).

In der Datei `.mailcap` kann man dann MIME Types mit Helper Applications verknüpfen.

Beispiele:

`.mime.types`:

```
type=application/postscript; exts=".ps,.eps"
```

`.mailcap`:

```
application/postscript; gv %s
video/quicktime; xanim %s
```

Auch Programme ausser Netscape sollten sich dieser Definitionen bedienen.

Da HTTP *state less* ist, kann man nur sehr eingeschränkt interagieren.

Ursprünglich war nur eine Form der Interaktion vorgesehen: FORMs

Das sind Formulare, die durch Tags im HTML Text ausgezeichnet werden und dann an den Server zurückgesendet werden.

Dort führen sie meist zur Ausführung eines Scriptes (CGI), welches wieder Daten produziert, die der Browser anzeigen kann.

Seit geraumer Zeit kann man auch noch sog. *Cookies* verwenden:

Das sind kleine Token, die bei jeder Abfrage an den Server geschickt werden.

In diesen kann dann state information gespeichert werden.

Ein Server kann auch neue Cookies zurückschicken.

Die Verwendung von Cookies hat Sicherheitsimplikationen.

Um die Zugriffe auf das Web schneller zu machen, verwendet man *Caches*.

Um durch Sicherheitsbarrieren (Firewalls) zu kommen, verwendet man *Proxies*.

Oft sind beide auch noch zusammengelegt.

Der Browser kann einen lokalen Cache im Speicher und/oder auf Platte halten.

Häufig benutzte Seiten können daraus geladen werden.

HTTP unterstützt Attribute, wie 'Cacheable' und 'Modified since': Cache Verwaltung möglich.

Ein Proxy dient zur Umsetzung interner HTTP (und anderer) Anfragen an die Aussenwelt.

Das ist nötig, wenn die Browser nicht direkt nach aussen zugreifen können.

Am RZ etwa ist ein Proxy eingerichtet, der auch als Cache dient (Einwahl von daheim).

Mail

Immer noch *der wichtigste* Dienst im Internet.

Prinzip: Ähnelt der Briefpost.

Nachrichten werden zwischen Sender und Empfänger(n) ausgetauscht.

Adressierung erfolgt dabei über eine eindeutige Adresse:

User@Domain

Aus den Anfangszeiten der Netzwerke gibt es auch heute noch Relikte (UUCP Mail, Bitnet,...)

Wichtig ist: Die *Domain* ist i.A. kein Rechnername.

Auslieferung geschieht also nicht zwischen zwei fixen Rechnern.

Die Mail ist eng mit dem DNS verknüpft.

Dort wird für jeden Domainnamen gepeichert, an welchen Rechner Post ausgeliefert werden soll (MX).

Wenn man also ein Mail verschickt, wird für jeden Adressaten der Postrechner gesucht.

An diesen geht dann die Nachricht.

Er kann sie dann weiterleiten oder auf Wunsch des Empfängers umleiten.

(Spezielle Ausnahmen: UUCP, Bitnet)

Eine Nachricht besteht immer aus zwei Teilen:

- Dem Header. Er enthält Sender, Empfänger, Subject, etc.
- Dem Body. Dieser enthält nur den reinen Nachrichtentext.

Der Header besteht aus Zeilen, die wiederum aus Feld und Text aufgebaut sind.

Der Body besteht ausschliesslich aus dem Text der Nachricht.

Vorsicht: Bei der Übermittlung wandeln viele Stationen den Body nach 7 Bit um.

Man kann daher z.B. keine Programme direkt übertragen.


```
Received: from sol.cs.uni-sb.de (IDENT:hYjl0o989FQ9AiAMuzmCH5ifXmo8tNCY@sol.cs.uni-sb.de [134.96.252.20]) by gigasun.cs.uni-sb.de
(8.9.3/1999020800) with ESMTP id NAA04803; Tue, 25 Jan 2000 13:36:37 +0100 (CET)
Received: from cs.uni-sb.de (cs.uni-sb.de [134.96.252.31]) by sol.cs.uni-sb.de (8.9.3/1999020800) with ESMTP id NAA03144 for
<fb-mit@sol.cs.uni-sb.de>; Tue, 25 Jan 2000 13:36:22 +0100 (CET)
Received: from gigasun.cs.uni-sb.de (gigasun.cs.uni-sb.de [134.96.242.3]) by cs.uni-sb.de (8.9.3/1999031900) with ESMTP id NAA22456;
Tue, 25 Jan 2000 13:36:18 +0100 (CET)
Received: from francoise.cs.uni-sb.de (francoise.cs.uni-sb.de [134.96.252.51]) by gigasun.cs.uni-sb.de (8.9.3/1999020800) with ESMTP
id NAA04787; Tue, 25 Jan 2000 13:36:17 +0100 (CET)
Received: (from franz@localhost) by francoise.cs.uni-sb.de (8.9.0/1998061600) id NAA03334; Tue, 25 Jan 2000 13:36:15 +0100 (CET)
Date: Tue, 25 Jan 2000 13:36:15 +0100 (CET)
Message-Id: <200001251236.NAA03334@francoise.cs.uni-sb.de>
From: franz@cs.uni-sb.de
To: profs@cs.uni-sb.de, fbmit@cs.uni-sb.de, sek@cs.uni-sb.de
Subject: neue Hausmeister-Rufe
Cc: franz@cs.uni-sb.de

An alle Mitglieder des Fachbereichs Informatik

Hausmeister Hoffmann ist ab sofort wie folgt zu erreichen:

Tel 2999 - 5610

Piepser - Sprachruf 71510
```

Man kann Mail an mehrere Adressaten schicken.

Man kann auch Kopien an zusätzliche Leute verschicken (CC)

Es gibt auch sog. *Mailing-Listen*.

Diese verteilen eingehende Mail an eine Reihe von Leuten weiter.

Eingehende Mail wird lokal im sog. *Spool Directory* gespeichert.

In diesem (`/var/mail` oder `/var/spool/mail`) gibt es für jeden Benutzer ein File, seine Mailbox.

Mit Mail Clients kann nun Nachrichten aus dieser Box in anderer verschieben, sie Lesen, Beantworten, etc.

Praktisch überall vorhanden ist das `mail` Programm.

Es zeigt beim Aufruf den Inhalt der Mailbox an und man kann dann Kommandos geben:

- Mit Return zeigt man die nächste Nachricht an.
- Mit `d` kann man eine Nachricht, oder mehrere Löschen.
- Mit `q` oder `x` verlässt man `mail`
- Durch Eingabe einer Nummer springt man zu der Nachricht.
- Durch `-`, `+` kann man navigieren.
- Durch `?` erhält man eine Hilfestellung.
- `r` oder `R` beantwortet eine Mail.
- `s` speichert eine Mail in ein File.

Zum Senden einer Mail gibt man etwa

`mail thesing@cs.uni-sb.de schellen@cs.uni-sb.de`

Danach wird man nach dem Subject gefragt und editiert die Mail.

Im Edit-Modus kann man mit `~v` den `vi` aufrufen.

Anderer Edit-Kommandos stehen in der Man-Page *mail(1)*.

Zum Abschluss gibt man einen Punkt ein, oder EOF.

Bequemer geht es natürlich mit anderen Mail-Clients:

- pine
- Netscape
- xfmail
- …

Wenn man Mail aus einem Account automatisch weiterleiten will, kann man dies durch das File `.forward` tun.

Dieses enthält Adressen, an welche die Mail automatisch weitergeleitet wird:

`\thesing, thesing@cs.uni-sb.de`

Auf manchen Systemen kann man auch Programme angeben, durch die die Mail gefiltert wird:

`|/usr/bin/foobarmail`

Es kann nützlich sein, *Aliase* für (lange) Adressen einzuführen.

Mit `mail` geschieht dies in dem File `.mailrc`:

`alias sn foobar@baz.com nixda@schnurz.org`

(In diesem File kann man auch Konfigurationen für `mail` setzen)

Mail Text wird meistens durch die Durchgangsserver auf 7 Bit gekürzt.

Daher kann man keine Binärdaten einfach per Mail verschicken.

Diese müssen vorher in ASCII kodiert werden.

Früher oft benutzt: `uuencode` und `uudecode` Sender:

```
> uuencode nix.ps.gz nix.ps.gz >baz.uu
> (include baz.uu in mail)
```

Empfänger:

```
> (save mail as e.g. xx)
> uudecode xx
> gv nix.ps.gz
```

`xx` sieht dann etwa so aus:

```
begin 644 rfc-index.gz
M'XL("/XABS@``W)F8RUI;F1E>`"L7%ESVTB2?A[^BHKHA[8C2!J%&YJ8B-5M
```


Heute mehr benutzt: MIME

Damit kann man mehrere Files einpacken und auch noch deren Typ angeben.

Dann könnte ein Mail Client etwa direkt eine Helper Application aufrufen.

Von Hand Packen/Entpacken geht auch: `mimencode`

Etwa `mimencode <xx >xx.mime`

oder `mimencode -u xx.mime`

Siehe *mimencode(1)*.

SSH

Durch das Wachstum des Internet gibt es jetzt auch mehr 'böse Buben'.

Sicherheitsrisiken ergeben sich u.a.:

- Durch das Einlogen. Verbindungen (telnet) können abgehört werden.
- Daten können von anderen gelesen werden.
- Man hat keine Authentifizierung anderer Rechner.

Einen Teil dieser Probleme löst SSH (Secure SHell).

Sie dient dazu, verschlüsselte Verbindungen zwischen zwei Rechnern aufzubauen.

Darüber kann man sich dann per Unix-Passwort oder durch Public-Key Verfahren am Zielrechner anmelden.

Dabei gibt es zwei Abstufungen:

- Die Verbindung wird verschlüsselt, die Anmeldung erfolgt aber weiter über das Unix-Passwort.
- Die Verbindung wird verschlüsselt und auch die Anmeldung erfolgt über ein Public-Key Verfahren.

Die erste Variante kann man immer einsetzen, die zweite nur, nachdem man ein eigenes Schlüssel-Paar generiert hat.

```
> ssh kap04.studcs.uni-sb.de
kap04.studcs.uni-sb.de
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)? yes
Host 'kap04.studcs.uni-sb.de' added to the list of known hosts.
Last login: Thu Apr  8 14:12:00 1999 from 134.96.242.163
NetBSD 1.4.1 (KAP) #0: Mon Sep  6 10:09:24 CEST 1999
```

Für die wirklich sichere Methode muss man ein Schlüsselpaar generieren.

```
> mkdir $HOME/.ssh
> ssh-keygen
Initializing random number generator...
Generating p:  .................++ (distance 244)
Generating q:  ..................................................++ (distance 666)
Computing the keys...
Testing the keys...
Key generation complete.
Enter file in which to save the key (/home/stud/thesing/.ssh/identity):
Enter passphrase:
Enter the same passphrase again:
Your identification has been saved in identity.
Your public key is:
1024 35 1361337724711057186432959439815842132753651593245804644834968326666861 62...
Your public key has been saved in identity.pub.
```

In den Files `.ssh/identity` und `.ssh/identity.pub` liegen dann der private und öffentliche Schlüssel.

Den öffentlichen Schlüssel muss man auf allen Rechnern, auf denen man sich einlogen will, hinterlegen.

Dazu in die Datei `.ssh/authorized_keys` schreiben (eine Zeile pro Schlüssel)

Nun wird bei jeder Verbindung durch den privaten Schlüssel der Benutzer authentifiziert.

Die Passphrase dient dazu, die Datei mit dem privaten Schlüssel nochmals zu verschlüsseln.

Damit kann nur der Benutzer selbst diesen verwenden. Auch eine Kopie der Datei nützt einem Angreifer nichts.

Allerdings ist es lästig, für jede Verbindung die Passphrase einzugeben.

Daher kennt SSH einen *Schlüssel-Agenten*.

Diesem teilt man einmal die Passphrase mit.

Man sagt allen anderen SSH Programmen, dass sie diesen Agenten nach dem privaten Schlüssel fragen sollen.

SSH Programme finden den Agenten über den Wert von Umgebungsvariablen.

Man muss also den Agenten starten und zwei Umgebungsvariablen setzen.

Dazu dient das `ssh-agent` Programm. Es gibt ein kurzes Shell Script aus, dass die Umgebungsvariablen setzt.

Man kann es etwa an den Anfang von `.xsession` stellen:

```
eval `/usr/bin/ssh-agent -s`
```

Danach sind die Umgebungsvariablen für alle folgenden Programme in `.xsession` gesetzt.

Am Ende (nach dem Windowmanageraufruf) sollte man diese Zeile setzen:

```
/usr/bin/ssh-agent -k
```


Nun läuft der Agent zwar, aber man muss ihm noch den privaten Schlüssel mitteilen.

Dazu dient das `ssh-add` Programm. Man setzt etwa in `.xsession` (nach `ssh-agent`!):

```
/usr/bin/ssh-add </dev/null
```

Beim Einlogen erscheint nun ein Fenster, das einen zur Eingabe der Passphrase für den privaten Schlüssel auffordert.

Nun benutzen alle SSH Programme den Agenten und haben automatisch den privaten Schlüssel.

SSH besteht aus einer Reihe von Programmen:

- `ssh` dient zum Einlogen in einen anderen Rechner. Man kann sich auch als jemand anders einlogen, siehe *ssh(1)*. Dazu muss derjenige in seinem `.ssh/authorized_keys` Ihren öffentlichen Schlüssel aufnehmen.
- Dateien kopieren: `scp` (s.u.)
- `slogin` ist ein Alias für `ssh`

Beispiel:

```
ssh -l foo baz
ssh baz.cs.uni-sb.de
scp wichtige_Datei kap02.studcs.uni-sb.de:remote_Datei
scp kap02:Unsinn lokaler_Unsinn
scp Unsinn foo@kap02:remote_foo_Unsinn
```


Man kann die Verbindung zwischen zwei Rechnern auch noch komprimieren.

Dazu gibt man in der Datei `.ssh/config` etwa folgendes:

```
Host *
Compression yes
CompressionLevel 6
```

Man kann auch XWindow Verbindungen durch SSH tunneln. Dann kann man auf dem entfernten Rechner einen Client starten, dessen Ausgabe auf dem lokalen Server erscheint.

Dazu setzt SSH per Default auf dem Zielrechner die Umgebungsvariable `DISPLAY` auf so was wie `:10.0`

Diese Verbindung wird dann von SSH durch die SSH Verbindung weitergeleitet.

Vorsicht: Wenn man `DISPLAY` auf dem Zielrechner in den Login-Scripten selbst setzt, geht die Setzung von SSH verloren.

Nachteile von SSH:

- Bietet keine perfekte Sicherheit.
- Verlangsamt den Zugriff.
- Verlust der Datei mit dem privaten Schlüssel nicht reversibel.
- Passphrase ist auch Schwachstelle (und nicht wiederherstellbar).