SMS Anwendungen

Stefan Frings, April 2007

Ein Buch über den Aufbau kommerzieller SMS Anwendungen im Stil eines Lehrganges:

- Vorstellung geeigneter Hardware
- Vorstellung geeigneter Betriebssysteme und Software
- Anwendung von GSM Modems zum zum Senden und Empfangen
- Einsteiger Anleitung und Anwendungsbeispiele für:
  - Shell Scripte, PHP, Awk
  - Apache Webserver
  - MySQL
  - Email Server
- Tips zur Verbesserung und Erweiterung der SMS Server Tools

Downgeloaded von http://stefanfrings.de

Inhaltsverzeichnis

1. Einleitung....9
1.1. Über dieses Buch....9
1.2. Über den Autor....10
1.3. Was ist SMS?....11
1.4. Was ist MMS?....12
2. Hardware und Software....13
2.1. SUN Server....13
2.2. PC Server....14
2.3. Serielle Schnittstellen....16
2.3.1. So funktioniert eine serielle Schnittstelle....16
2.3.2. Anschlussbelegungen....17
2.3.3. Serielle Ports erweitern....19
2.3.4. Digi Etherlite....19
2.4. GSM Modems und Mobiltelefone....20
2.4.1. Siemens GSM Modems....20
2.4.2. Falcom GSM Modems....20
2.4.3. Mobiltelefone....22
2.5. GSM Antennen....23
2.6. Betriebssysteme....24
2.6.1. Microsoft Windows....24
2.6.2. SUN Solaris....25
2.6.3. Linux....26
2.6.4. Vergleichstabelle....27
2.7. Programme....28
2.7.1. PDU Spy....28
2.7.2. CygWin....29
2.7.3. Apache Webserver....29
2.7.4. War FTP Daemon....29
2.7.5. MySQL Datenbank....30
2.7.6. PHP Programmiersprache....30
2.7.7. nnCron Lite....30
2.7.8. SMS Server Tools....31
2.7.9. EditPad Lite....31
2.7.10. Fetchmail, ssmtp und ncftp....32
3. Systemdesign....33
3.1. Stabilität....33
3.2. Mechanischer Aufbau....34
3.3. Stromversorgung....35
3.4. Datensicherheit....36
3.5. Redundanz....37

3.6. Firewall....38
3.7. Konkretes Beispielsystem....39
4. Basis KnowHow....41
4.1. SMS Befehle für GSM Modems....41
4.1.1. Terminalprogramm starten....41
4.1.2. GSM Modembefehle....42
4.1.3. SMS Modembefehle....43
4.1.4. PDU Datenformat (Empfangen)....45
4.1.5. PDU Datenformat (Status Report)....46
4.1.6. PDU Datenformat (Senden)....48
4.1.7. Zeichensatz....50
4.1.8. Praktische Beispiele....53
4.2. Software Installation....55
4.2.1. Software Installation unter Solaris....55
4.2.1.1. Solaris konfigurieren....55
4.2.1.2. Packages Installieren....56
4.2.1.3. MySQL konfigurieren....57
4.2.1.4. Apache and PHP compilieren....58
4.2.1.5. Apache und PHP konfigurieren....59
4.2.1.6. Apache, PHP und MySQL Testen....60
4.2.1.7. Sourcen installieren....61
4.2.1.8. Merkliste für Solaris....62
4.2.2. Softwareinstallation unter Windows....63
4.2.2.1. Windows Programme Installieren....63
4.2.2.2. CygWin Installieren....64
4.2.2.3. Apache und PHP Installieren....66
4.2.2.4. Apache, PHP und MySQL Testen....67
4.2.2.5. SMS Server Tools Installieren....68
4.2.2.6. Merkliste für Windows....68
4.2.3. Softwareinstallation unter Linux....69
4.2.3.1. Vorbereitung....70
4.2.3.2. SMS Server Tools Installieren....70
4.2.3.3. MySQL Konfigurieren....71
4.2.3.4. Apache, PHP und MySQL Testen....72
4.2.3.5. Merkliste für Linux....73
4.3. CygWin Schnelleinstieg....74
4.3.1. Die Shell....74
4.3.2. Dateinamen....75
4.3.3. Verzeichnisnamen....77
4.3.4. Zugriffsrechte....78
4.3.5. Vergleich Windows/Unix Befehle....79
4.3.6. Suchpfade....80

4.4. Arbeiten mit Cron Jobs....81
4.5. Arbeiten mit eMails....83
4.5.1. Sendmail Konfigurieren....83
4.5.1.1. Sendmail konfigurieren unter SuSE Linux....83
4.5.1.2. Sendmail konfigurieren unter RedHat Linux....84
4.5.1.3. Sendmail Konfgurieren unter Solaris....84
4.5.2. ssmtp Konfigurieren....85
4.5.3. eMail senden....86
4.5.4. eMail empfangen....87
4.5.5. eMail Verarbeiten....88
4.5.6. Formail anwenden....89
4.6. Arbeiten mit FTP....90
4.6.1. FTP Client benutzen....90
4.6.2. Automatische FTP Downloads....92
4.6.3. Automatische FTP Uploads....93
4.7. MySQL Anwenden....94
4.7.1. Das Funktionsprinzip von MySQL....94
4.7.2. Datentypen....95
4.7.3. Optionen zu Datentypen....96
4.7.4. Eine Datenbank mit zwei Tabellen anlegen....97
4.7.5. Tabellen mit Daten Füllen....98
4.7.6. Tabellen Auslesen....99
4.7.7. Tabellen Verknüpfen....100
4.7.8. NULL Werte....101
4.7.9. Weitere Schlüsselwörter für Select....102
4.7.10. Spalten Ändern....103
4.7.11. Daten Löschen....104
4.7.12. SQL Abfragen in Shell Scripten....105
4.7.13. Datensicherung....107
4.8. Dynamische Webseiten mit PHP....109
4.8.1. Die erste Webseite....109
4.8.2. Die erste dynamische Webseite....110
4.8.3. print(), printf() und sprintf()....111
4.8.4. Variablen und Datentypen....112
4.8.5. Operatoren....114
4.8.6. Kontrollstrukturen....116
4.8.6.1. if...elseif...else....116
4.8.6.2. switch...case....117
4.8.6.3. while und do...while....118
4.8.6.4. for....118
4.8.6.5. foreach....119
4.8.6.6. break, continue und exit....119

4.8.7. include....120
4.8.8. Funktionen....121
4.8.9. Formular Variablen und Datei-Uploads....122
4.8.10. SQL Abfragen....123
4.8.11. Lokale und globale Variablen....125
4.8.12. Variable Variablen....127
4.8.13. PHP Scripte ausserhalb von Apache....127
4.8.14. Wie geht es weiter?....128
4.9. Shell Script Programmierung....129
4.9.1. Das erste Shell Script....129
4.9.2. Der printf Befehl....130
4.9.3. Zeichenketten....131
4.9.4. Variablen....132
4.9.5. Ein/Ausgabe Umleitung in Dateien....134
4.9.6. Ein/Ausgabe Umleitung in Programme....134
4.9.7. Ein/Ausgabeumleitung in Variablen....134
4.9.8. Mehrere Befehle in einer Zeile....135
4.9.9. Kontrollstrukturen....136
4.9.9.1. if...then...else...fi....136
4.9.9.2. Vergleiche....137
4.9.9.3. Mehrere Vergleiche verknüpfen....138
4.9.9.4. case...esac....138
4.9.9.5. while...do...done....140
4.9.9.6. for...in...do...done....140
4.9.9.7. break, continue und exit....141
4.9.10. Funktionen....142
4.9.11. Spezielle Variablen....143
4.9.12. Mehr als neun Argumente....144
4.9.13. Arithmetische Ausdrücke mit expr....145
4.9.14. Fehlermeldungen....146
4.9.15. Nützliche Hilfsprogramme....147
4.9.15.1. cat....147
4.9.15.2. cut....148
4.9.15.3. sed....149
4.9.15.4. grep....150
4.9.15.5. Reguläre Ausdrücke....151
4.9.15.6. Leerzeichen wegschneiden....152
4.10. Awk Programmierung....153
4.10.1. Wie man awk startet....153
4.10.2. Erste awk Beispiele....153
4.10.3. Grundsätzliche Syntax von awk Scripten....154
4.10.4. Feldtrenner....154

4.10.5. Eingebaute Variablen....156
4.10.6. Variablen an der Kommandozeile übergeben....156
4.10.7. Operatoren....157
4.10.8. Bedingungen mit if....158
4.10.9. Schleifen mit while....159
4.10.10. Schleifen mit for....159
4.10.11. Break und continue....160
4.10.12. Next und exit....160
4.10.13. Sonderzeichen in print....160
4.10.14. Printf und sprintf....161
4.10.15. Arrays....162
4.10.16. Eingebaute Funktionen....163
4.10.17. Eigene Funktionen....164
4.10.18. Awk Beispiel für SMS Anwendungen....165
5. Die SMS Server Tools....167
5.1. Überblick....167
5.2. Installation und Konfiguration....172
5.2.1. Einfache Konfigurationsdatei....173
5.2.2. Mehrere Modems und Netzanbieter....174
5.2.3. Alle Konfigurationseinstellungen....175
5.2.3.1. Haupteinstellungen....176
5.2.3.2. Warteschlangen....178
5.2.3.3. Netzanbieter....178
5.2.3.4. Modemeinstellungen....180
5.3. SMS Server Tools starten/stoppen....182
5.4. Kurzmitteilungen senden....183
5.5. Kurzmitteilungen empfangen....184
5.6. SMS Dateiformat....185
5.6.1. Texte senden....186
5.6.2. Binärdateien senden....187
5.6.3. Unicode Texte senden....188
5.6.4. Empfangene Texte....189
5.6.5. Empfangene Binärdateien....190
5.6.6. Empfangene Statusberichte....191
5.6.7. Zeitstempel....192
5.7. Modemauswahl....193
5.8. Status Monitor....194
5.9. Ereignisprogramme....195
5.10. Alarm Programme....196
5.11. Schwarze und weiße Listen....197
5.12. Statistik Dateien....198
5.13. Protokollierung in SQL Datenbank....199

5.14. Mysmsd erklärt....201
6. Sinnvolle Erweiterungen....205
6.1. Wartungsfreier Dauerbetrieb....205
6.1.1. Protokolldateien pflegen....206
6.1.2. Alte SMS Dateien löschen....209
6.1.3. Alte SQL Daten löschen....209
6.1.4. Selbst-Test....210
6.2. Statusmonitor verbessern....213
6.2.1. Statusmonitor als Webseite....214
6.2.2. Alarme als bunte Webseite....215
6.3. Mehr als 32 Modems....216
6.4. Eine SMS an mehrere Empfänger....217
7. Praktische Anwendungsbeispiele....218
7.1. Zentrales Alarmsystem....218
7.1.1. Senderseite....218
7.1.2. Empfängerseite....219
7.2. Fahrzeugortung....220
7.2.1. Senderseite....220
7.2.2. Empfängerseite....220
7.3. eMail zu SMS Gateway....222
7.3.1. eMail zu SMS Gateway mit lokalem Mailserver....222
7.3.2. eMail zu SMS Gateway mit entferntem Mailserver....223
7.4. Absender mit SQL prüfen....224
7.5. Eisprung Erinnerung....225
7.5.1. Das Anmeldeformular....225
7.5.2. Das Anmelde-Script....226
7.5.3. Die Tabelle mit Terminen....228
7.5.4. Der Cronjob....229
7.6. Klingeltöne, Logos und Sprüche....230
7.6.1. Content bereitstellen....230
7.6.2. Das Script....231
7.7. Web-Formular sendet Nachrichten....231
7.7.1. Das Web-Formular....232
7.7.2. Das sende-Script....232

1. Einleitung

In diesem Kapitel beschreibe ich die Absichten dieses Buches.

Ich stelle mich selbst vor und beschreibe die Grundprinzipien der SMS und MMS Dienste.

1.1. Über dieses Buch

SMS ist eines von vielen Phänomenen des 20. Jahrhunderts.

Ursprünglich völlig ignoriert, erfreuen sich die kurzen Texte inzwischen einer erstaunlichen Beliebtheit. Die Netzbetreiber und zahlreiche weitere Dienstleister erzielen durch SMS Dienste einen signifikaten Umsatz.

Viele Handybesitzer investieren mehr Geld in SMS Dienste, als in Telefongespräche. In Deutschland nutzen etwa 62 Mio. Bürger das Mobiltelefon, davon 85% SMS Dienste.

Wissenschaftler sammeln Messdaten per SMS zur zentralen Auswertung und Autobesitzer lokalisieren gestohlene Fahrzeuge, die ihre Position per SMS melden. Alarmanlagen und Maschinen melden Störfälle per SMS.

SMS gibt es schon seit rund 10 Jahren, doch kaum jemand ist sich bewusst, was man mit den Kurzmitteilungen alles anfangen kann. Ich möchte Ihnen in diesem Buch einige Anwendungsbeispiele vorführen, von der Idee bis zum physikalischen Aufbau und dem Betrieb.

Ich habe dieses Buch bewusst in kurzen und bündigen Sätzen geschrieben, weil ich Sie nicht unnötig lange mit dem Lesen inhaltloser Worte aufhalten möchte. Die Branche ist schnellebig und wer nicht entsprechend schnell arbeitet, kommt nicht weit.

Lesen Sie das Buch von vorne nach hinten durch. Der Schwierigkeitsgrad steigt stetig und die Kapitel bauen jeweils auf die vorherigen Kapitel auf.

Mit SMS Diensten wird viel Geld verdient.

Schneiden Sie sich ein Stück von diesem Kuchen ab!

In diesem Buch tauchen zahlreiche registrierte Firmennamen, Produktnamen, Logos und Fotos auf. Diese sind Eigentum der entsprechenden Firmen.

1.2. Über den Autor

Mein Name ist Stefan Frings. Ich wurde 1974 in Deutschland geboren und arbeitete mich bereits als Kind in die Funktionsweise von Elektronik und Computer ein. In der Schule entwickelte ich zusammen mit dem Technik-Lehrer Bausätze für die nachfolgenden Jahrgänge und reparierte elektronische Geräte der Schule.

Nach der Ausbildung zum Kommunikationselektroniker Fachrichtung Telekommunikation, arbeitete ich zunächst als EDV Techniker, dann als Programmierer für SMS Dienste und Datenbanken und dann als Hardware-Entwickler.

Die dort erlernten Qualifikationen ermöglichten mir einen zentralen Arbeitsplatz bei dem grössten Mobilfunknetz-Betreiber Europas, wo ich für den Betrieb von rund 250 Computern verantwortlich bin und Schulungen durchführe.

Mein kostenloses Programm SMS Server Tools erfreut sich großer Beliebtheit. Es wird von zahlreichen Firmen für die unterschiedlichsten Zwecke kommerziell eingesetzt.

Wenn Sie Rückfragen oder Kritik zum Buch haben, können Sie mich per eMail unter der Adresse smstools@meinemullemaus.de erreichen.

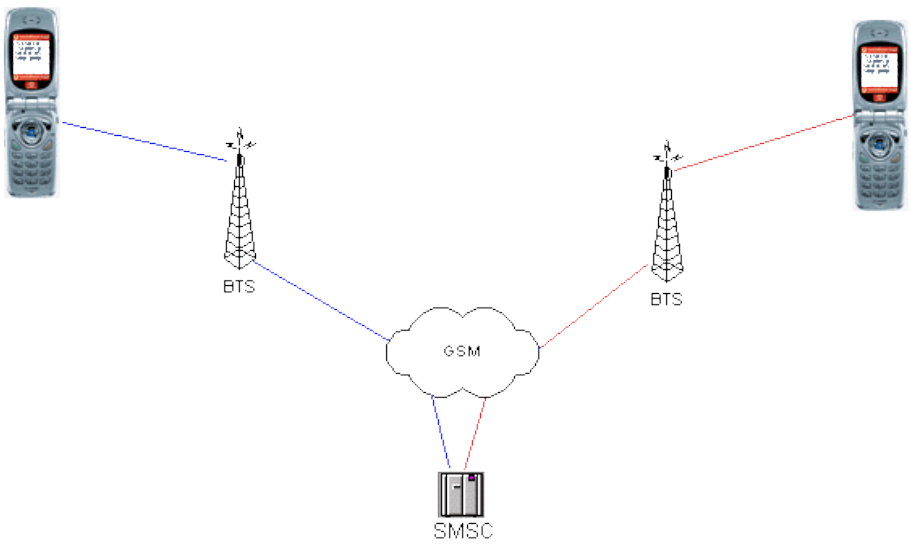
1.3. Was ist SMS?

SMS ist die Abkürzung für

- Short
- Message
- Service

Dabei handelt es sich um eine kurze Textmitteilung mit maximal 160 Zeichen (7 Bit), die von Mobiltelefon zu Mobiltelefon gesendet werden können.

Manche Geräte unterstützen auch binäre Kurzmitteilungen mit 8 Bit zur Übertragung kleiner Bilder, Klingeltöne oder Konfigurationsdaten für das Mobiltelefon.



Die Kurzmitteilung wird normalerweise über die Tastatur des Mobiltelefons eingegeben und zunächst auf der SIM-Karte gespeichert. Dann wird sie durch den Signalisierungskanal, der auch für den Rufaufbau und Rufabbau zuständig ist, zum SMSC gesendet.

Da es mehrere SMSC gibt, kann man im Mobiltelefon konfigurieren, welches man verwenden möchte.

Das SMSC speichert die Kurzmitteilung bis zu zwei Tage lang und sendet sie ebenfalls durch den Signalisierungskanal zum Empfänger. Je nach Wunsch sendet es ausserdem eine Zustellbenachrichtigung an den Absender.

1.4. Was ist MMS?

MMS ist die Abkürzung für

- Mulitmedia
- Message
- Service

Multimedia Messages sind Mitteilungen, die neben Text auch Bilder, Videos und Audio enthalten können. Die neueste Generation von Mobiltelefonen, wie z.B. das abgebildete Sharp GX-10 unterstützen diese Funktion.

MMS ist nicht Thema dieses Buches, doch da der neue Dienst gerade sehr aktuell und mit SMS stark verwandt ist, möchte ich ihn hier kurz vorstellen.

Eine MMS besteht immer aus einem Text, dem man optional Dateien anhängen kann. Diese Vorgehensweise kennt man von eMails. Das MMSC überträgt diese Nachricht ohne Veränderung zum Empfänger.

Wenn man eine MMS sendet, baut das Mobiltelefon eine Internetverbindung auf (genauer: WAP) und sendet den Text und die Anhänge an das MMSC.

Das MMSC speichert diese und prüft zunächst, ob der Empfänger als MMS Kunde registriert ist. Wenn er nicht registriert ist, bekommt er eine gewöhnliche SMS mit einer Internetadresse (URL), unter der er die MMS im Internet mit einem WebBrowser betrachten kann.

Wenn der Empfänger als MMS Kunde registriert ist, bekommt er eine Benachrichtigungs SMS. Diese veranlasst das Mobiltelefon, eine Internetverbindung (genauer: WAP) aufzubauen, um die MMS downzuloaden. Nach erfolgreichem Download, wird die Nachricht im Mobiltelefon gespeichert und angezeigt.

Derzeit können Sie MMS Anwendungen nur in direkter Zusammenarbeit mit den GSM Netzbetreibern entwickeln.

2. Hardware und Software

In diesem Kapitel stelle ich Ihnen exemplarisch geeignete Hardwarekomponenten vor, und die in diesem Buch verwendete Software. Da die Hardware-Hersteller sehr schnell neue Geräte auf den Markt bringen, kann dieses Kapitel nie ganz aktuell sein. Wenn Sie die Hersteller ansprechen, werden diese Sie gerne über die jeweils aktuellen Nachfolger der hier vorgestellten Geräte informieren.

2.1. SUN Server

Die Firma SUN Microsystems liefert Hardware mit höchster Qualität. SUN Server sind nicht nur für ihre hohe Performance bekannt, sondern auch für geringe Ausfallzeiten. Verglichen mit Intel basierten Computern (PC's) läuft das Multitasking herausragend flüssig und sauber. SUN Server können ohne Mühe weit über 1000 Programme gleichzeitig ausführen, ohne dass der Betrieb des Servers ins Stocken gerät.

Der Technische Support von SUN arbeitet vorbildlich, sowohl was Reaktionszeiten angeht, als auch die fachliche Kompetenz des Personals.

Sun Server werden üblicherweise mit dem Betriebssystem Solaris ausgeliefert. Sun bietet jedoch auch Windows- und Linux kompatible Server an.

SUN Fire V120

SUN Netra T1000

SUN Fire V480

2.2. PC Server

PC Server laufen mit dem Betriebssystem Linux oder Windows. Es werden sehr viele unterschiedliche Geräte angeboten.

Achten Sie beim Kauf besonders auf die Qualität der Hardware. Die gerade neuesten Rechner sind oft eine schlechte Wahl, da sie noch unzuverlässig laufen. Für SMS Anwendungen spielt die Geschwindigkeit des Prozessor keine Rolle.

Stellen Sie sich folgende Fragen:

- Hat der Server alle benötigen Anschlüsse?
- Ist der mechanische Aufbau stabil und ordentlich?
- Leistet der Lieferant ausreichenden Support?
- Sind Ersatzteile ausreichend verfügbar?
- Passen nur Originalteile dieses Herstellers oder Modells in das Gehäuse, so dass Sie von deren Verfügbarkeit abhängen?
- Ist die Hardware zum gewünschten Betriebssystem (Windows/Linux) kompatibel?

Einige Beispiele:

Dell Poweredge 850

HP BL20P

IBM Blade Center HS20

SUN Fire B200x

2.3. Serielle Schnittstellen

In diesem Kapitel beschreibe ich die Funktion einer seriellen RS232 Schnittstelle. Sie finden hier die Anschlussbelegung unterschiedlicher Stecker, sowie ein Gerät zur Erweiterung der Anschlüsse.

2.3.1. So funktioniert eine serielle Schnittstelle

Eine serielle Schnittstelle heisst so, weil sie die Zeichen seriell überträgt. Ein Zeichen besteht aus 8 Bits, welche nacheinander durch eine einzige Leitung gesendet werden.

Beispiel:

Der Buchstabe A hat den Code 65, denn er ist das 65. Zeichen in der Zeichensatz-Tabelle. Binär ist das die Zahl 01000001.

Zur Trennung der Zeichen wird ein Startbit mit dem Wert 0 vorangestellt und ein oder zwei Stopbits mit dem Wert 1 hinten angehängt. Die acht Datenbits selbst werden von rechts nach links übertragen, also das niederwertigste Bit zuerst.

Die Übertragung eines einzelnen A sieht so aus:

```
.......0 1 0 0 0 0 0 1 0 1.......
```

Und zwei A direkt hintereinander sehen so aus:

```
.......0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1.......
```

Die 0 wird als +12V übertragen und die 1 als -12V. Wenn die Schnittstelle gerade keine Zeichen überträgt (hier durch Punkte dargestellt), steht die Leitung auf -12V.

Die Übertragung eines einzelnen A sieht dann so aus:

```
+ + + + + + + + + - + + + + + - + - + + + + + + + + +
```

Zur Übertragung der Zeichen gibt es an jedem seriellen Anschluss zwei Leitungen. Die Leitung TxD sendet Daten und die Leitung RxD empfängt Daten.

Bei einer solchen zwei-Draht Verbindung fehlt allerdings die Möglichkeit, den Datenfluss (Handshake) zu steuern. Der sendende Computer weiss gar nicht, wann der Empfänger überhaupt empfangsbereit ist.

Zu diesem Zweck hat man zwei Zeichen reserviert, die zur Signalisierung verwendet werden. Sie heissen Xon und Xoff. Zur Steuerung von Modems hat man sich allerdings entschieden, noch ein paar Steuerleitungen hinzuzufügen, damit die Datenleitungen jedes beliebige Zeichen übertragen können und die Programmierung einfach bleibt. Diese Leitungen nennt man Hardware Handshake Leitungen.

- Mit CTS zeigt das Modem an, dass es empfangsbereit ist.
- Mit DTR zeigt das Modem an, dass es eingeschaltet ist.
- Mit DCD zeigt das Modem an, dass der es eine Verbindung zu einer Gegenstelle aufgebaut hat. Nicht alle SUN Computer haben eine DCD Leitung, aber das ist für SMS Anwendungen egal, weil wir diese Leitung ohnehin nicht brauchen.
- Mit DSR zeigt der Computer an, dass der serielle Port belegt ist.
- Mit RTS bittet der Computer das Modem, sich bereit zu machen und zeigt gleichzeitig an, dass der Computer seinerseits empfangsbereit ist.

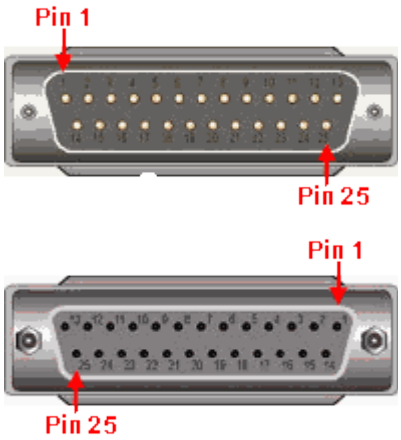
Die Handshake-Leitungen werden vom Betriebssystem gesteuert. Ausserdem richtet dass Betriebssystem einen Pufferspeicher ein, in dem empfangene Zeichen zwischengelagert werden, bis das laufende Programm diese ausliest. Wenn die Pufferspeicher des Computers oder Modems voll sind, zeigen die Geräte dies durch die CTS und RTS Leitung an.

Die Spezifikation der RS232 Schnittstelle schreibt vor, dass die Signale 1 und 0 durch -12V und +12V dargestellt werden, wobei aber eine Toleranz im Bereich 3-15V zugelassen wird. In der Praxis liegen die Spannungen der seriellen Schnittstellen meistens zwischen 9 und 12 Volt.

2.3.2. Anschlussbelegungen

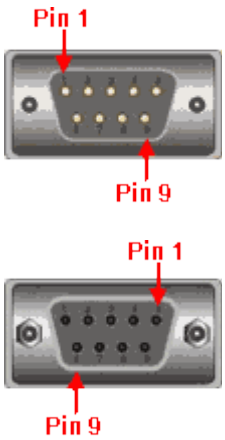
Achtung: SUN Computer haben oft eine eigentümliche Pinbelegung!

25 poliger SUB-D Anschluss (Modem und PC)



2 TxD
3 RxD
4 RTS
5 CTS
6 DSR
7 GND
8 DCD
20 DTR

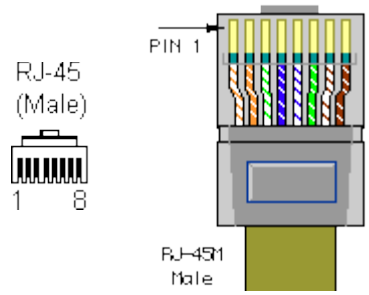
9 poliger SUB-D Anschluss (Modem und PC)



1 DCD
2 RxD
3 TxD
4 DTR
5 GND
6 DSR
7 RTS
8 CTS

RJ45 Anschluss (Sun Fire V120 und V480)

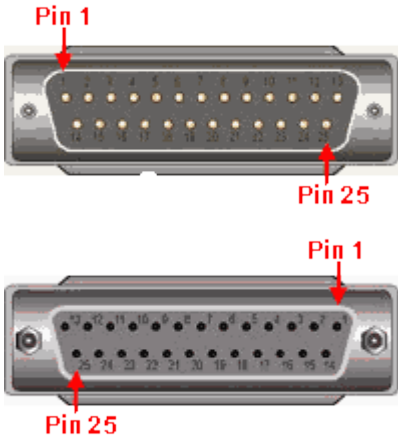
Diese Server haben keine DCD Leitung am seriellen Port!



1 RTS
2 DTR
3 TxD
4 GND
5 GND
6 RXD
7 DSR
8 CTS

25 poliger SUB-D Anschluss (SUN Fire V880)

Bei diesem Server befinden sich beide seriellen Ports in einem Stecker! Da es hier keine DSR Leitung gibt, muss man das Modem so einstellen, dass es dieses Signal ignoriert, oder man verbindet auf Modem Seite DSR mit RTS.

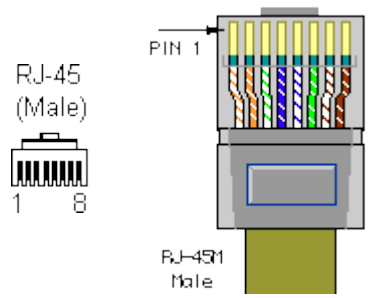


2 TxD Port A
3 RxD Port A
4 RTS Port A
5 CTS Port A
8 DCD Port A
20 DTR Port A

7 GND

11 DTR Port B
12 DCD Port B
13 CTS Port B
14 TxD Port B
16 RxD Port B
19 RTS Port B

RJ45 Anschluss (Digi Etherlite)



1 RTS
2 DSR
3 DCD
4 RxD
5 TxD
6 GND
7 DTR
8 CTS

2.3.3. Serielle Ports erweitern

Wenn der Server nicht genug serielle Schnittstellen hat, kann man diese auf unterschiedliche Weise erweitern:

- mehrere Server verwenden
- manuelle Umschalter
- serielle Ports über USB
- zusätzliche PCI Karten
- serielle Ports über Ethernet

Die erste Variante scheidet aus Kostengründen aus, die zweite ist auch nicht praktikabel, denn der Server soll automatisch funktionieren. Serielle Ports kann man elegant über die USB Schnittstelle erweitern, doch diese Geräte werden derzeit leider ausschliesslich mit Windows Treibern geliefert.

Zusätzliche PCI Karten lassen sich oft nur sehr bedingt einbauen, weil die Anzahl der verfügbaren PCI Slots begrenzt ist. Durch das Öffnen des Gerätes erlischt ausserdem bei vielen Herstellern die Garantie.

Die letzte Möglichkeit, serielle Ports über Ethernet zu erweitern, halte ich für die beste. Es gibt zahlreiche Geräte auf dem Markt, die serielle Ports über Ethernet erweitern. Sie schliessen ein kleines Kästchen an das Netzwerk des Servers an und erhalten damit bis zu 32 neue serielle Ports. Diese funktionieren exakt so, als wären es lokale Ports im Server selbst. Serielle Ports über Ethernet funktionieren unter Windows, Solaris und Linux, sowie alle anderen gängigen Betriebssystemen.

Im folgenden Kapitel stelle ich Ihnen eine entsprechende Geräteserie der Firma Digi vor.

2.3.4. Digi Etherlite

Die Digi Etherlite Geräte erweitern die seriellen Ports eines Servers um zwei, acht, sechzehn oder zweiunddreissig Schnittstellen. Und wem das nicht genug sind, der kann ohne weiteres mehrere dieser Geräte zusammen einsetzen.

Bei SMS Anwendungen werden Sie allerdings niemals mehr als 32 serielle Ports brauchen.

Digi Etherlite Geräte funktionieren unter Linux, Windows und Solaris einwandfrei. Aus Sicht der Software funktionieren diese Schnittstellen exakt so, als wären es lokale Schnittstellen innerhalb des Servers.

Durch den Ethernet Anschluss zum Server können Sie die Modems weit vom Server entfernt aufstellen, ohne dazu zahlreiche teure Kabel verlegen zu müssen.

Digi Etherlite Geräte besitzen eine interne Elektronik zur Unterdrückung von Masseschleifen.

Masseschleifen kennen Sie vieleicht von Ihrem Heim-PC, der mit der Stereoanlage verbunden ist. Häufig hat man dort ständig ein lästiges Brummen in der Audio-Leitung, dass man durch den Einbau von Übertragern in die Audio-Leitungen oder in das Antennenkabel unterdrücken kann. Bei Modems, die über andere Steckdosen mit Strom versorgt werden, als der Server, kann eine Masseschleife die Signale der seriellen Schnittstelle so weit verzerren, dass die Modems ausfallen. Wie gesagt, kann Ihnen das beim Einsatz eines Digi Etherlite nicht passieren.

Bitte beachten Sie, dass sie einen Windows PC benötigen, um ein Digi Etherlite zu konfigurieren. Mit einem kleinen Windows Programm stellen Sie die IP-Adresse des Gerätes ein. Danach können Sie das Gerät ans Netzwerk anschliessen und auch unter Linux und Solaris betreiben.

Auf der Webseite http://www.digi.com finden Sie Lieferanten für diese Geräte.

2.4. GSM Modems und Mobiltelefone

In diesem Kapitel stelle ich einige GSM Modems vor, die sich für SMS Anwendungen bewährt haben. Sie erfahren zudem auch, welche Kriterien bei der Auswahl eines Mobiltelefons wichtig sind und warum diese für professionelle Anwendungen nicht geeignet sind.

2.4.1. Siemens GSM Modems

Für fast alle Siemens GSM Modems benötigen sie zum Betrieb noch ein 12V Netzteil mit mindestens 500mA Belastbarkeit, sowie eine GSM Antenne und eine SIM Karte. Wenn das Netzteil nicht stabilisiert ist, sollten Sie einen Kondensator mit 2200yF 25V parallel schalten.

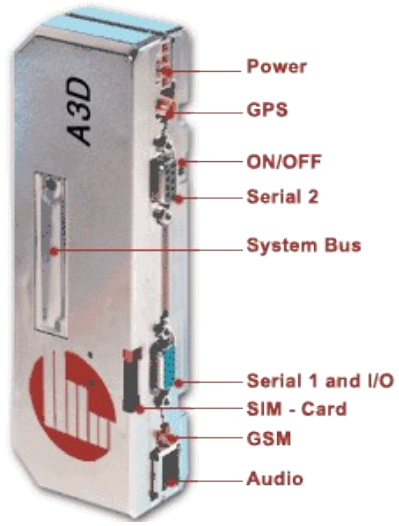
Das Siemens TC35 Terminal ist das derzeit handelsübliche GSM Modem von Siemens.

Um mit Siemens Modems Status-Reports empfangen zu können, müssen Sie diese mit **AT+CNMI=2,0,0,2,1** initialisieren.

2.4.2. Falcom GSM Modems

Falcom Modems zeichnen sich durch beste Zuverlässigkeit aus. So wundert es nicht, dass ich alle meine eigenen GSM Anwendungen bereits vor Jahren rasch auf Falcom Geräte umgestellt hatte, sobald das erste GSM Modem dieses Herstellers auf den Markt kam.

Falcom bietet für jeden Anwendungsfall genau das richtige GSM Modem oder GSM Modul an. Es gibt sogar Geräte mit erweiterbarer Firmware und GPS Antenne, die bei Anruf oder per SMS die Position melden.



Falcom hat Techniker, mit denen man gut Probleme und Aufgabenstellungen besprechen kann, um so das geeignete Produkt auszuwählen.

Die Geräte werden mit einem praktischen Halteclip geliefert, den man zur Montage des Gerätes verwenden kann. So hat das Gerät seinen festen Platz, kann aber bei Bedarf aus dem Clip herausgenommen werden.

Das Falcom A3D ist derzeit das flexibelste GSM Modem, dass ich für SMS Anwendungen empfehlen kann.

Neben normalen Datenverbindungen, SMS, Voice Calls und Fax kann dieses Modem selbst mit einer eigenen Firmware programmiert werden. Die entsprechenden Anleitungen kann man auf der Webseite des Herstellers http://www.falcom.de downloaden.

Das Alarm-Programm sendet eine Alarm-SMS, wenn einer von drei unterschiedlichen Eingängen ein Signal erhält. Auf diese Anwendung gehe ich in Kapitel 7 näher ein.

Optional kann das Modem mit einem GPS Empfänger ausgestattet werden, welcher die Position per SMS und Datencall meldet.

Das Falcom A3D funktioniert zuverlässig und ohne Einschränkungen für alle SMS Anwendungen dieses Buches.

Für dieses Falcom GSM Modem benötigen sie zum Betrieb noch ein 12V Netzteil mit mindestens 800mA Belastbarkeit, sowie eine GSM Antenne und eine SIM Karte. Wenn Sie die GPS Funktion nutzen wollen, brauchen Sie ausserdem eine GPS Antenne. Wenn das Netzteil nicht stabilisiert ist, sollten Sie einen Kondensator mit 2200yF 25V parallel schalten.

Das Falcom A3D ist mit einer Notstrombatterie (Li Ion) ausgestatt, die die Grundfunktionen (GSM, ohne Alarmeingänge) 4 Stunden lang versorgt, wenn die externe Stromversorgung ausgefallen ist.

In dem Modem befindet sich ein interessanter Chip, mit dem Sie auf AT-Befehlsebene Dateien senden und empfangen können. Der Chip unterstützt unter anderem die folgenden Internetprotokolle: POP3, SMTP, HTTP und FTP. Der Chip ist besonders nützlich, wenn Sie das Modem ohne Computer mit selbst entwickelter Firmware betreiben.

Wenn Sie einfach nur SMS Senden oder Empfangen möchten, gefallen Ihnen vielleicht auch die folgenden Geräte von Falcom:

Falcom Twist

Falcom Tango

Falcom Samba

Initialisieren Sie das Falcom Samba mit dem Befehl **AT^SSMSS=1**, damit Sie empfangene Kurzmitteilungen auslesen können. Dieser kleine USB Stecker hat keine externe Antenne und ist daher nicht an jedem Standort nutzbar.

Vodafone Connect Card GPRS/UMTS (Hergestellt von Option)

Diese Modemkarte hat einen internen USB Controller und einen USB2Serial Adapter. Das Modem darin ist also ein serielles Modem. Die Karte kann auch Unter Linux verwendet werden. Laden Sie den Treiber wie fogt:

modprobe ohci
modprobe usbserial vendor=0x0af0 product=0x5000

Das Device heisst dann /dev/usb/ttyUSB0

2.4.3. Mobiltelefone

Generell sind auch Mobiltelefone für SMS Anwendungen geeignet. Jedoch hat sich herausgestellt, dass sie für professionelle Anwendungen nicht stabil genug funktionieren. Selbst Handies, die im normalen Tagesgebrauch absolut zuverlässig funktionieren, machen oft plötzlich Probleme, sobald sie mit einem Computer verbunden werden.

Mobiltelefone sind nicht für ständigen Gebrauch gebaut. Die Möglichkeit, einen Computer anzuschliessen, ist für Notebooks gedacht und nur für temporären Gebrauch. 90% aller Support Anfragen mit Problemberichten zum Thema SMS beziehen sich auf Mobiltelefone!

Deshalb rate ich dringend davon ab, Mobiltelefone an Computer anzuschliessen. Für Testzwecke mag das eine feine Sache sein, aber nicht für professionelle Anwendungen.

Da die angebotene Produktpalette beinahe monatlich wechselt, kann ich in diesem Buch keine Mobiltelefone empfehlen.

Bitte beachten Sie, dass nahezu alle Mobiltelefone zwar Status-Reports empfangen können, diese aber für den Computer unerreichbar sind und nicht ausgelesen werden können. Auch der Empfang und das Senden von binären Nachrichten mit 8 Bit ist mit Mobiltelefonen nicht oder nur eingeschränkt möglich.

Wenn Sie ein serielles Kabel benutzen, dann muss dieses Hardware-Handshake und RS232 Protokoll unterstützen. FBUS und MBUS Kabel sind nicht geeignet. Viele nicht originale Anschlusskabel unterstützen kein Hardware-Handshake, während die originalen Kabel immer vollständig beschaltet sind.

Die Infrarot-Schnittstelle hat sich (unter Linux) als weitaus unproblematischer herausgestellt. Denken Sie bitte auch daran, dass eine Infrarot-Schnittstelle wesentlich flexibler einsetzbar ist: Sie brauchen nicht für jedes neue Mobiltelefon ein neues Kabel kaufen.

Unter Windows können Sie die Infrarotschnittstelle nicht mit den SMS Server Tools benutzen, weil CygWin diese nicht unterstützt.

Nicht alle Mobiltelefone können SMS über Infrarot oder Serielle Schnittstelle SMS senden. Achten Sie bei der Geräteauswahl auf das Stichwort GSM 07.05 oder internes Modem in der Auflistung der Funktionen.

2.5. GSM Antennen

Sie können selbstverständlich jede beliebige GSM 900/1800 Antenne an die Modems anschliessen.

Als besonders praktisch haben sich die abgebildeten Magnetfussantennen herausgestellt, die für KFZ Dachmontage gedacht sind. Sie können Antennen dieser Bauform leicht auf 19 Zoll Schränke oder Computergehäuse stellen.

Wenn Sie Probleme mit schwachem Empfang haben, wählen sie eine möglichst große Antenne und stellen Sie diese möglichst weit von anderen elektrischen Geräten entfernt auf.

Mehrere GSM Antennen sollten wenigstens 10cm Abstand zueinander haben und sie dürfen (ausser am Sockel) nichts berühren.

Bevor sie ein GSM Modem ohne Antenne einschalten, sollten Sie sich wenigstens eine provisorische Drahtantenne (33 cm lang) basteln. Wenn Sie ein GSM Modem ohne Antenne einschalten, riskieren Sie einen Geräteschaden. Benutzen Sie auch keine Antennen mit beschädigten oder eingeknickten Anschlusskabeln.

2.6. Betriebssysteme

In diesem Kapitel stelle ich Betriebssysteme vor, die ich für SMS Anwendungen benutze.

Nach dem Lesen dieses Kapitels werden Sie leichter entscheiden können, welches Betriebssystem für Ihre Anwendung am besten geeignet ist.

Ich vergleiche die Betriebssysteme Linux, Solaris und Windows unter dem Hintergrund, darauf SMS Anwendungen laufen zu lassen.

2.6.1. Microsoft Windows

Windows gibt es in zahlreichen Varianten. Für SMS Anwendungen sind derzeit die folgenden Versionen geeignet:

- Windows NT 4
- Windows 2000 Server
- Windows XP

, wobei Microsoft den Support für Windows NT inzwischen eingestellt hat.

Bitte sehen Sie davon ab, einen Server mit einer Workstation Version von Windows zu betreiben, wie Windows 95, 98 oder ME. Obwohl alle Anwendungen dieses Buches auch unter diesen Versionen laufen, ist der rund-um-die-Uhr Betrieb dieser Windows Versionen nicht ratsam.

Viele Leute klagen darüber, dass die Workstation Versionen im Laufe der Zeit immer langsamer werden und irgendwann aus unerklärlichen Gründen stehen bleiben. Doch dieses Problem kann leicht erklärt werden:

Die Workstation Versionen haben eine einfachere Speicherverwaltung. Sie sind darauf getrimmt, möglichst viel Speicher und CPU Leistung für die Anwendungsprogramme (primär Spiele) zur Verfügung zu stellen. Dazu verzichten sie unter anderem auf eine Defragmentierung des Arbeitsspeichers, was früher oder später zur bekannten Verlangsamung des Systems und zu mangelnder Verfügbarkeit großer zusammenhängender Speicherblöcke führt.

Bei Windows 95, 98 und ME habe ich mehrmals beobachtet, dass die Uhr nach einigen Tagen Betriebszeit ungenau wurde. Der Uhrenchip auf dem Mainboard lief aber genau und nach einem Neustart stimmte die Zeitanzeige wieder.

Windows ist eine feine Sache, wenn Sie lediglich einen Fileserver aufbauen wollen. In allen anderen Fällen sind Sie jedoch in der Regel darauf angewiesen zusätzliche Software zu kaufen (z.B. für Webserver, eMail, Fax, Statistische Auswertung, zeitgesteuerte Programme).

Da die Softwareentwicklung für Windows im allgemeinen nicht Open-Source ist, können Sie die erworbenen Programme nicht verändern. Die Verknüpfung unterschiedlicher Programme unterschiedlicher Hersteller ist auf Windows Systemen typischerweise nicht möglich. Windows-Softwarehersteller konkurrieren so stark miteinander, dass Sie in der Regel jegliche Zusammenarbeit mit anderen Firmen verweigern.

Ein großer Teil der Server-Software kann natürlich direkt von Microsoft bezogen werden, und die Programme lassen sich gut miteinander verknüpfen. Doch dann hängen Sie von einem einzigen Hersteller ab.

Bitte bedenken Sie auch, dass Windows und Microsoft Programme immer wieder das Haupt-Angriffsziel von Viren sind.

2.6.2. SUN Solaris

Sun bietet das Betriebssystem Solaris sowohl für die eigenen Sun Computer, als auch für PC an. Zur Zeit ist die Versionen 9 aktuell (auch 2.9 genannt).

Da die Hersteller von PC Hardware aber nur sehr selten Solaris Treiber entwickeln und Sun sich auch nicht ernsthaft darum kümmert, können Sie Solaris auf einem PC nur bedingt einsetzen.

Ein SUN Server mit Solaris läuft sehr zuverlässig und effizient. Da SUN einen guten (allerdings auch teuren) Support anbietet, ist dieser Lieferant empfehlenswert, wenn Sie ein zuverlässiges System mit möglichst geringen Ausfallzeiten benötigen.

Solaris hat etwas weniger Werkzeuge, als Linux. Dennoch ist eigentlich alles drin, was man so braucht. Und die wenigen Programme, die man zusätzlich benötigt, bekommt man grösstenteils kostenlos aus dem Internet.

Ein umfangreiches Verzeichnis mit Solaris Freeware finden sie z.B. unter http://www.sunfreeware.com.

Die meisten Anwendungsbeispiele in diesem Buch beziehen sich auf Linux, aber Solaris ist wie Linux ein Unix-Betriebssystem und daher weitgehend mit Linux identisch. Solange es alle für Ihre Anwendung nötigen Programme auch für Solaris gibt, spricht nichts dagegen, sich für einen SUN Computer mit SUN Solaris zu entscheiden. Die Anwendungen dieses Buchen laufen unter Solaris ab Version 6.

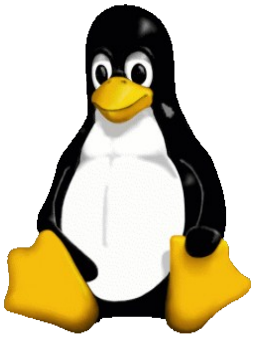
Wer an einem SUN Computer arbeitet, sollte möglichst die folgenden Schulungen bei SUN besuchen:

- Grundlagen von Solaris
- Solaris Administration 1
- Solaris Administration 2
- Solaris Netzwerkadministration
- Shell Programmierung für Systemadministratoren

In diesen Schulungen lernt der EDV Techniker alle wesentlichen Befehle und Konfigurationsdateien kennen, mit denen man das Betriebssystem Solaris administriert.

Mein Fazit: SUN ist teuer, aber richtig gut. Entscheiden Sie gemäss Ihren Prioritäten.

2.6.3. Linux



Linux ist ein Betriebssystem von Linus Torvalds und unzähligen weiteren Entwicklern. Der Name ist eine Kombination aus Unix und Linus, dem Entwickler des Betriebssystem-Kerns.

Linux ist ein Unix ähnliches Betriebssystem für PC. Die Entwicklung begann zu DOS Zeiten und geht seit dem rasch voran. Da Linux von Anfang an als Multi-Tasking Betriebssystem mit Netzwerk ausgelegt war, ist es in vielen Punkten etwas Leistungsstärker, als Windows.

Der Name Linux bezeichnet eigentlich nur den Betriebssystem-Kern. Eine Unzahl von professionellen- und Hobbyprogrammierern entwickeln für Linux immer mehr Werkzeuge, Treiber und Anwendungsprogramme. Einige Firmen sammeln diese Programme zusammen mit dem Linux Kern und fügen ein Installationsprogramm hinzu.

Jede Linux Installation basiert auf einer solchen Linux Distribution, und daher hat sich die Bedeutung des Wortes "Linux" schleichend verändert. Die Distributionen unterscheiden sich im Umfang und Support-Angebot. Ich empfehle Distributionen von RedHat und SuSE.

RedHat ist das Vorbild aller Linux Distributionen. RedHat ist weniger umfangreich, als SuSE Linux. Ein weiterer Unterschied ist, dass SuSE in Deutschland sehr aktiv ist und Hardware, Software sowie Schulungen anbietet. RedHat belässt die Programme auf der Distribution normalerweise im Originalzustand, während SuSE viele Programme leicht modifiziert, um den Installations- und Administrationsaufwand zu reduzieren.

Entscheiden Sie sich für RedHat, wenn Sie programmieren und ihre Anwendung möglichst problemlos auch unter anderen Linux Versionen laufen sollen. Wählen Sie SuSE, wenn Sie eher Wert auf eine einfache installation und Konfiguration legen.

Linux ist das ideale Betriebssystem für Programmierer, da es kostenlos mit allen erdenklichen Programmiersprachen und Werkzeugen ausgestattet wurde. Nahezu alle Programme werden als Open-Source veröffentlicht, also mit Quelltext. Dies ermöglicht Programmierern, die Programme nach eigenen Wünschen zu verändern, sie durch Untersuchen des Quelltextes zu prüfen oder auf deren Basis eigene Programme zu schreiben.

Im Gegensatz zur Firma Microsoft, die zunehmend eigene Ideen als Weltstandard forciert, arbeiten die Linux Programmierer gemeinsam am großprojekt "Linux". Das führt dazu, dass die Programme gut zusammen passen, obwohl sie von unterschiedlichen Entwicklern kommen.

Linux ist hauptsächlich kostenlos. Daher gibt es auch in den meisten Fällen keinen kommerziellen Support von den Entwicklern. Doch Firmen wie SuSE bieten kommerziellen Support an und dieser wird nur selten in Anspruch genommen, weil die Autoren der Programme meist direkt per eMail ansprechbar sind und gerne helfen.

Da die meisten Softwareentwickler von Linux Hobbyprogrammierer, Schüler und Studenten sind, finden Sie im Internet kostenlose Hilfe für viele Probleme und Aufsätze über alles, was man als Linux Administrator lernen sollte.

Ein umfangreiches Verzeichnis mit Linux Software finden sie z.B. unter http://www.freshmeat.net.

Linux ist in der Regel erheblich performanter, als Windows. Sie können dies nachvollziehen, indem Sie den Apache Webserver unter beiden Betriebssystemen installieren und die Zeiten vergleichen.

Wenn Sie eine SMS Anwendung unter Linux entwickeln wollen, brauchen sie unbedingt Personal mit entsprechenden Linux Kenntnissen. Bei Windows kann man sich notfalls noch "durchklicken", wenn man nicht Bescheid weiss und fachlichen Rat bekommt man für Windows in jedem Computershop. Aber um Linux zu konfigurieren, muss man sich mit der Befehlszeile und langen Anleitungen vertraut machen. Mit Linux kennen sich nur wenige Händler aus und die verkaufen ihr KnowHow teuer. Kostenlose Hilfe gibt es im Internet, aber ohne Grundkenntnisse hat man es auch dort schwer.

2.6.4. Vergleichstabelle

Wenn Sie sich nun für ein Betriebssystementscheiden müssen, dann entscheiden Sie zunächst danach, für welches System es alle nötige Software gibt. Wenn Sie ein Programm benötigen, dass es z.B. für Windows nicht gibt, dann können Sie Ihre Anwendung auch auf gar keinen Fall unter Windows entwickeln. Darüberhinaus sind folgende Kriterien nach Aufwand und Nutzen abzuwägen:

	Linux	**Solaris**	**Windows**
Support für Betriebssystem	Installations-Support, keine Gewährleistung	voller Support	voller Support
Funktionsumfang	sehr umfangreich	mittel	gering
Integration in Netzwerke mit anderen Betriebssystemen	sehr leicht	schwierig	fast unmöglich
Kompatibilität zu Standards	fast vollständig	fast vollständig	Microsoft definiert eigene Standards
Kosten für Betriebssystem	minimal	gering	hoch
Kosten für Betrieb und Software *	gering	hoch	hoch
Verfügbarkeit von Programmier-Tools	hoch	gering	mittel
Verfügbarkeit von Server-Software	hoch	mittel	mittel
Allgemeine Stabilität	mittel	hoch	gering
Support im Internet	umfangreich	gering	mittel
Erreichbarkeit der Autoren	mittel	gar nicht	gar nicht
Transparenz des Codes	sehr gut	gar nicht	gar nicht
Verknüpfung OS typischer Software mit selbst entwickelter Software	leicht möglich	möglich	kaum möglich
Anfälligkeit für Viren/Hacker	gering	gering	hoch
Unabhängigkeit von Hardware Lieferant	ja	nein	ja
Unabhängigkeit von Software Lieferant	ja	nein	nein
Allgemeine Performance	mittel	hoch	niedrig

* Unter der Annahme, dass Fachpersonal bereits vorhanden ist.

2.7. Programme

In diesem Kapitel stelle ich Programme vor, die ich für SMS Anwendungen benutze.

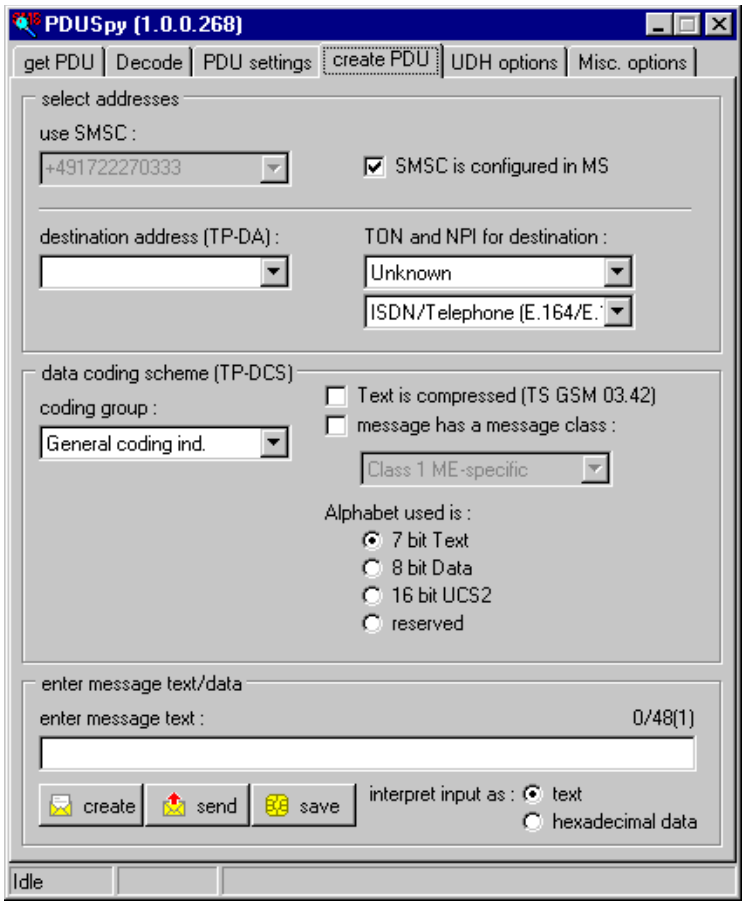
Sie werden fest stellen, dass die Programme alle primär unter Linux entwickelt wurden und anschliessend für andere Betriebssysteme angepasst wurden. Daher werden besonders die Windows Administratoren grafische Oberflächen mit bunten Buttons vermissen.

Trotzdem sind die Programme unter Windows und Solaris einwandfrei brauchbar. Sie sollten sich nicht davon abschrecken lassen, dass sich die meiste Dokumentation im Internet auf Linux bezieht.

Ich gehe davon aus, dass Sie sich mit den jeweiligen originalen Dokumentationen vertraut machen, wenn Sie mit den Programmen arbeiten.

2.7.1. PDU Spy

PDU Spy ist ein Werkzeug für Programmierer. Damit kann man SMS Codieren und Decodieren. Ausserdem kann das Programm einzelne SMS per GSM Modem senden und empfangen.



Programmierer können mit PDU Spy Fehler in ihren Programmen aufspüren.

Das Programm gibt es auf der Homepage des Autors unter http://www.nobbi.com.

Unter Linux können Sie es mit Hilfe des Windows Emulators wine starten. Für Solaris gibt es leider keine Version.

Tip: Wenn Sie das Programm mit der Option -d starten, erzeugt es ein Logfile, in dem die Modembefehle protokolliert werden.

2.7.2. CygWin

CygWin ist ein Set von leistungsstarken Werkzeugen, die Programmierern die Entwicklung und Migration von Unix Programmen nach Windows erleichtern.

CygWin ist kein Emulator. Die Programme, die man mit CygWin entwickelt sind echte Windows Programme.

CygWin besteht hauptsächlich aus dem GNU C Compiler, der cygwin1.dll Library und einem Satz von Programmen, die zum Unix Standard gehören. Das Paket wird von RedHat entwickelt. Es wird kostenlos unter http://www.cygwin.com angeboten.

Ich verwende CygWin als Basis aller selbst geschriebener Windows Programme. So kann ich die von Unix gewohnten Befehle gleichermassen unter Windows benutzen - Programme laufen einfach unter beiden Betriebssystemen.

Nach der Installation von CygWin werden Sie neben der "MS-Dos Eingabeaufforderung" auch eine "CygWin Bash Shell" finden, in der sie alle Unix Befehle eingeben können.

Die Installation von CygWin erfordert ca 150 MB freien Festplatten-Speicherplatz. Sie ist kinderleicht. Downloaden Sie zuerst den CygWin Installer und starten Sie ihn. Dieser bietet Ihnen dann eine Liste von optionalen Komponenten an.

Der Installer downloaded alle Dateien aus dem Internet und installiert sie.

2.7.3. Apache Webserver

Der Apache Webserver ist mit großem Abstand der meisst genutzte Webserver der Welt. 80% aller Webseiten werden mit diesem Server betrieben. Er unterstützt CGI Scripte in Perl, PHP und TCL, sowie ssl Verschlüsselung. Webseiten können mit Passwörtern versehen werden.

Sie können den Apache Webserver von der Apache Webseite http://www.apache.org kostenlos downloaden.

Eine sehr umfangreiche Anleitung -auch für den Schnellstart- finden Sie ebenfalls auf der Apache Webseite. Für die ersten Versuche sind Beispiel-Webseiten mitgeliefert.

2.7.4. War FTP Daemon

Der War FTP Daemon ist für alle Windows Computer mit Internet Anbindung eine optimale Ergänzung.

Während der Apache Webserver nur das HTTP Protokoll zur Verfügung stellt, können Sie mit dem War FTP Daemon einen vollwertigen FTP Server aufbauen.

Die Zugriffsrechte lassen sich sehr komfortabel per Mausklick einstellen und sie können beliebig viele User anlegen.

Installieren Sie den War FTP Daemon, wenn Sie Ihren Kunden Downloads oder Uploads per FTP ermöglichen möchten.

Sie erhalten das Programm kostenlos unter http://www.jgaa.com.

2.7.5. MySQL Datenbank

MySQL ist -wie der Name schon sagt- eine SQL Datenbank.

Ich benutze MySQL immer, wenn eine Anwendung mittlere bis große Datenmengen speichern und wieder laden muss. Die Verwendung einer SQL Datenbank bringt im Vergleich zu einer selbst entwickelten Datenbank meistens Vorteile.

MySQL ist schnell. Sie können mit Hilfe zahlreicher grafischen Programme (z.B. Microsoft Access, StarOffice, MySQL Navigator) auf die Daten zugreifen und sie verändern.

Natürlich gibt es auch ein entsprechendes Programm für die MS-Dos Eingabeaufforderung bzw. Unix Shell. Dieser sogenannte SQL Client ist bereits im Lieferumfang von MySQL enthalten. Mit Hilfe des SQL Clients können Sie sogar aus einfachen Batch-Dateien oder Shell Scripten heraus auf die Datenbank zugreifen.

Wenn Sie eine MySQL Datenbank einsetzen, dann empfehle ich Ihnen als Nachschlagewerk das gedruckte deutsche Handbuch "MySQL Das offizielle Handbuch" ISBN 3-8266-0888-7.

MySQL speichert wie alle SQL Datenbanken Ihre Daten in tabellarischer Form ab. Sie können beliebig viele Datenbanken anlegen, die jeweils beliebig viele Tabelle speichern. Jede Tabelle kann wiederum beliebig viele Reihen und Spalten haben. Das Konzept ist ähnlich, wie in Microsoft Access, nur dass bei SQL eine klare Trennung zwischen Datenbank Server und Datenbank Client besteht.

MySQL Datenbanken können Sie in allen gängigen Programmiersprachen benutzen. In diesem Buch greifen wir mit Shell-Scripten und PHP Scripten auf MySQL Datenbanken zu.

MySQL bietet kommerziellen Support, sowie Schulungen an. Sie bekommen MySQL kostenlos auf der Webseite http://www.mysql.com. Beachten Sie die Lizenzbestimmungen, denn MySQL ist nicht für alle Anwendungsfälle kostenlos.

2.7.6. PHP Programmiersprache

PHP ist eine Script Programmiersprache für Webseiten mit dynamischen Inhalten.

Wenn Sie z.B. einen Online Shop anbieten möchten, dann verwenden Sie Apache als Webserver und MySQL als Datenbank. Mit PHP schaffen Sie die Verbindung zwischen dem Webserver und der Datenbank.

PHP ähnelt der C++ Programmiersprache. Besonders praktisch ist, dass Sie PHP Code mitten in HTML Dateien einfügen können, genau wie Sie es vieleicht von JavaScript her gewohnt sind.

Der Vorteil von PHP ist, dass die Scripte vom Webserver ausgeführt werden und nicht vom Webbrowser. PHP Scripte können besonders einfach auf SQL Datenbanken zugreifen.

In vielen Fällen können Sie JavaScripte durch PHP Schripte ersetzten und so die lästigen Kompatibilitätsprobleme der Webbrowser umgehen.

Es werden regelmässig weltweit Schulungen angeboten. Das umfangreiche Handbuch wurde in viele Sprachen (auch deutsch) übersetzt. Sie bekommen alles rund um PHP kostenlos auf der Webseite http://www.php.net.

2.7.7. nnCron Lite

Das Programm nnCron Lite ermöglicht das zeitgesteuerte Starten von Programmen.

nnCron Lite ist nur für Windows erhältlich. Es läuft unter allen Windows Versionen.

Unter Unix gibt es das gleichwertige Programm cron, welches zum Standard-Umfang jeder Unix Installation gehört.

Nach der Installation tragen Sie in die Konfigurationsdatei cron.tab ein, welches Programm sie wann starten wollen. Übrigens ist das Dateiformat praktischerweise mit Unix identisch.

Wer nicht nur Programme starten möchte, sondern auch noch steuern möchte (also Benutzereingaben simulieren), sollte sich die grössere Version nnCron (ohne Lite) ansehen.

Sie können das Programm kostenlos unter http://www.nncron.ru/ downloaden.

Falls Sie jetzt feststellen, dass es auch in CygWin ein cron Programm gibt, dann haben Sie Recht. Zu dem Zeitpunkt, als das Buch geschrieben wurde, funktionierte das Programm jedoch nicht. Daher habe ich Ihnen alternativ das Programm nnCron Lite empfohlen.

2.7.8. SMS Server Tools

Die SMS Server Tools sind eine Sammlung ausgereifter Programme zum Senden und Empfangen von Kurzmitteilungen.

Sie senden eine Kurzmitteilung, indem Sie einfach eine Textdatei mit der Zielrufnummer und dem Text erzeugen und in ein bestimmtes Verzeichnis abspeichern. Das Hauptprogramm smsd überwacht dieses Verzeichnis ständig und sendet die Nachricht mit Hilfe von GSM Modems.

Empfangene Kurzmitteilungen werden jeweils als eigene Textdatei in einem anderen Verzeichnis gespeichert und stehen so zur weiteren Verarbeitung zur Verfügung.

Mit Hilfe von Shell Scripten und Webserver schaffen Sie schnell und flexibel Verbindungen zu anderen Programmen sowie zum Internet.

Dieses Buch stützt sich hauptsächlich auf die SMS Server Tools. Das Programmpaket wird das Kernstück Ihrer SMS Anwendungen sein.

Sie können die originalen SMS Server Tools kostenlos von meiner Homepage http://www.meinemullemaus.de downloaden, auf die sich dieses Buch bezieht.

Unter http://smstools3.kekekasvi.com/ finden Sie neueste Version der SMS Server Tools.

2.7.9. EditPad Lite

EditPad Lite ist ein Texteditor für Windows. Im Vergleich zu Notepad hat er folgende Zusatzfunktionen:

- Öffnen von mehr als einer Datei gleichzeitig
- Leistungsstarke Suchen/Ersetzen Funktion
- Öffnet auch Dateien von Unix und Macintosh (keine Probleme mit Zeilenumbrüchen)
- Unbegrenzte Dateigrösse
- Unbegrenzte Rückgängig-Funktion
- Markieren von rechteckigen Blöcken (mit Alt-Taste)
- Bessere Druckfunktion
- Umwandlung von OEM (Dos) Zeichensatz in Ansi (Windows) Zeichensatz und umgekehrt
- Anzeige der Zeilen und Spaltennummer des Cursors

Wenn Ihnen EditPad Lite gefällt, dann möchten Sie sich vielleicht auch EditPad Pro ansehen. Diese grössere Version hat unter anderem folgende Zusatzmerkmale:

- bunte Syntax Hervorhebung für alle gängigen Programmiersprachen und HTML
- Hex-Editor
- Dateivergleich
- Zusammenfassung mehrerer Dateien in eine Projekt
- Lesezeichen
- Sonderzeichen-Tabelle
- Rechtschreibkontrolle
- Sortieren
- Reguläre Ausdrücke
- Sechs Zwischenablagen

Sie erhalten Editpad auf der Webseite http://www.editpadlite.com/editpadlite.html. Bei nicht-kommerzieller Anwendung ist das Programm kostenlos.

2.7.10. Fetchmail, ssmtp und ncftp

Fetchmail und ssmtp sind zwei kleine eMail Programme, die in jeder Linux Distribution enthalten sein sollten.

Mit fetchmail rufen Sie eMails von externen Mailservern ab, z.B. mit POP3 Protokoll.

Mit ssmtp versenden sie eMails über externe Mailserver mit smtp Protokoll. ssmtp simuliert das unter Unix übliche sendmail, indem es alle eMails an einen anderen Mailserver weiter leitet.

Diese beiden Programme können nützlich sein, wenn Sie keinen eigenen Mailserver auf Ihren Computern betreiben. Sie ermöglichen den eMail Transfer an der Kommandozeile oder innerhalb von Shell-Scripten bzw. Batch-Dateien.

Ncftp ist ein FTP Client, der besonders gut für Shell-Scripte bzw. Batch-Dateien geeignet ist. Im Gegensatz zum normalen FTP Client ist dieser nicht interaktiv, sondern Sie bestimmen anhand von Argumenten, welche Dateien sie up- oder downloaden wollen.

3. Systemdesign

In diesem Kapitel beschreibe ich wichtige Aspekte für das Design einer kommerziellen SMS Anwendung.

Nachdem Sie es gelesen haben, werden Sie alle nötigen und sinnvollen Komponenten zusammenstellen können. Sie werden ausserdem wissen, auf welche Details man bereits beim Aufbau eines entsprechenden Systems achten sollte.

Das Kapitel unterstützt Sie bei der Auswahl der Hardware nach den Anforderungen höchster Zuverlässigkeit.

3.1. Stabilität

Für kommerzielle Anwendungen ist die Stabilität eines Systems sehr wichtig, vielleicht sogar wichtiger, als die Kosten.

Bei der Software haben Sie leider nicht viele Wahlmöglichkeiten. Solange Sie sich auf die in den vorangegangenen Kapiteln vorgestellten Programme stützen, brauchen Sie jedoch keine bösen Überraschungen befürchten. Die Programme haben alle eine lange Entwicklungszeit hinter sich und laufen daher zuverlässig. Und sollten Sie dennoch einen Fehler entdecken, stehen Ihnen die Autoren der Programme meist gerne mit Rat und Tat zur Seite.

Im Gegensatz zur Software haben Sie auf die Hardware einen großen Einfluss. Für SMS Anwendungen spielt die Geschwindigkeit eines Server kaum eine Rolle, prinzipiell können Sie jeden beliebigen PC oder SUN Rechner verwenden. Sie können also sehr frei den für Sie richtigen Rechner auswählen.

SUN Computer sind generell sehr zuverlässig, aber sie laufen nur mit dem Betriebssystem Solaris. Wenn Sie gerne einen SUN Computer verwenden möchten, dann müssen Sie unbedingt auch einen geschulten Solaris Administrator haben und Sie müssen prüfen, ob alle gewünschten Programme auf für Solaris erhältlich sind. Wenn diese Bedingungen nicht sicher erfüllt sind, dann können Sie keinen SUN Server für Ihre SMS Anwendung benutzen.

Für PC's können Sie auf jeden Fall praktisch jedes Programm bekommen entweder für Windows oder für Linux. Hier geht es erst einmal um die Auswahl der Hardware. Auf dem Markt werden sowohl hochwertige PC's, als auch Billigmodelle angeboten. Der Preis sagt nicht unbedingt etwas über die Qualität der Hardware aus, das wissen Sie sicher.

Kaufen Sie daher PC's nur bei Händlern, die Ihnen auch einen ausreichenden Reparatur- und Ersatzteilservice anbieten. Es wäre sehr unangenehm, wenn Ihr SMS Dienst - und damit Ihr Gewinn - tagelang ausfällt, weil ein Ersatzteil nicht mehr verfügbar ist.

Erfahrene EDV Händler können Ihnen aus dem stehgreif sagen, welche Hardwarekomponenten zuverlässig sind, und welche eher problematisch sind.

Im Zweifelsfall ist der Kauf eines Marken-Servers angebracht, wie zum Beispiel die zuvor vorgestellten Geräte von Dell. Mit großen Anbietern können Sie nämlich Wartungsverträge und finanzielle Ausgleiche bei Überschreitung gewisser Reparaturzeiten vereinbaren.

Und noch ein kleiner Tip zur Ausstattung von PC-Servern: Je weniger drin ist, um so besser. Bemühen Sie sich, Ihre Anwendung mit möglichst wenig Hardwarekomponenten zu realisieren. Je weniger Bauteile im Computer stecken, desto stabiler läuft er und desto schneller können defekte Teile lokalisiert werden. Wenn Ihr Rechner mit unnötigen Gerätschaften bestückt ist (z.B. Soundkarte, TV-Karte, internes Modem, usw.), dann entfernen Sie diese.

Neben der Hard- und Softwareauswahl kann der mechanische Aufbau einen erheblichen Einfluss auf die Systemstabilität haben. Damit befasst sich das nächste Kapitel.

3.2. Mechanischer Aufbau

Der mechanische Aufbau kann einen erheblichen Einfluss auf die Stabilität eines Systems haben.

Stellen Sie sich eine wilden Haufen von Gerätschaften vor, die mitten im Raum auf dem Boden liegen. Sie haben jetzt vermutlich schon ein Bild im Kopf, wo gerade jemand darüber fällt und zahlreiche Kabel beschädigt. Natürlich würden Sie niemals auf die Idee kommen, Computer so aufzustellen.

Ein Professionelles System befindet sich in einem 19 Zoll Schrank. Die Massangabe bezieht sich auf den Abstand der linken und rechten Tragschienen im Schrank, an die Sie alle Geräte schrauben. Sie können Server, Switche, Aufstellböden und so weiter für diese Schränke bekommen. 19 Zoll Schränke gibt es mit durchsichtigen Türen und in abgeschirmter Variante mit Metall- oder Gittertüren. Sie ermöglichen es, Ihre Hardware mit handelsüblichem Montagematerial stabil und geschützt zu montieren.

Erden Sie den Schrank. Nur so ist ihre Hardware vor statischen Ladungen geschützt. Heutzutage führen statische Ladungen zwar nur sehr selten zu einem Defekt, aber sie können Ihr System sehr wohl kurzzeitig lahm legen.

Verlegen Sie Kabel grundsätzlich in geordneten Bahnen im Doppelboden oder in Kabelkanälen.

Stellen Sie keine Geräte lose innerhalb des Schrankes auf, sondern befestigen Sie alles gut - auch die Kabel. Auf diese Weise verhindern Sie , dass Erschütterungen oder Bewegungen die Kabel nicht beschädigen oder zu Wackelkontakten führen.

Achten Sie auf ausreichende Lüftung. Die Lufttemperatur innerhalb des Schrankes sollte 40°C nicht überschreiten. Um dies sicherzustellen sind die meisten 19 Zoll Schränke mit entsprechenden Ventilatoren ausgestattet. Wenn Ihr Schrank keine gelochten Türen besitzt, sondern nur Lüftungsschlitze, dann halten Sie die Türen geschlossen, damit die Belüftung korrekt funktioniert. Manche Schränke heizen sich bei geöffneten Türen mehr auf, als bei geschlossenen.

Stellen Sie Antennen niemals im Innern des Schrankes auf. Sie würden damit die Modems zu erhöhter Sendeleistung zwingen und zugleich die Empfangsqualität verschlechtern. Die hohe Sendeleistung wiederum könnte die Hardware im Schrank stören.

Stellen Sie den Schrank so auf, dass sie die vordere und hintere Türe öffnen können. Ansonsten werden Sie bei Montagearbeiten Schwierigkeiten bekommen.

Wenn Sie einen Monitor und eine Tastatur in den Schrank einbauen wollen, dann verwenden Sie einen TFT Monitor (z.B. 15 Zoll) und eine Spezialtastatur für 19 Zoll Schränke. Diese sind etwas schmaler, als normale Tastaturen und haben wahlweise sogar einen eingebauten Trackball. Montieren Sie die Tastatur in eine herausziehbare Schublade.

Zum stabilen Betrieb gehört auch eine stabile Stromversorgung. Darauf geht das nächste Kapitel ein.

3.3. Stromversorgung

Eine stabile Stromversorgung ist Grundvoraussetzung für den zuverlässigen Betrieb.

Heutige Computer stellen keine hohen Ansprüche an die Stromversorgung, doch sie können äusserst empfindlich auf hochfrequente Störimpulse aus der Steckdose reagieren - nämlich mit einem Absturz. Diese Störimpulse werden beispielsweise von Leuchtstofflampen erzeugt und treten daher praktisch in jedem Gebäude auf.

Installieren Sie einen guten Netzfilter gegen hochfrequente Störungen. Sie bekommen solche Geräte in jedem Elektrohandel. Normalerweise sind diese mit einem Überspannungsschutz kombiniert, den ich Ihnen ebenfalls nur empfehlen kann.

Überspannungen entstehen vor allem beim Abschalten von Motoren und Leuchtstofflampen, wenn diese nicht korrekt entstört sind, sowie durch Blitzeinschlag. Für verhältnismässig kleines Geld schützen Sie Ihre teure Hardware Überspannungen.

Bitte beachten Sie, dass Filter und Überspannungsableiter in unterschiedlichster Qualität angeboten werden. Viele Geräte haben keine nennenswerte Schutzwirkung. Lassen Sie sich im Elektro-Fachhandel Ihres Vertrauens vor dem Kauf beraten.

USV

ist die Abkürzung für "Ununterbrochene Stromversorgung" und bezeichnet Geräte, die mit Hilfe von Batterien Stromausfälle überbrücken. Wenn Sie Ihrem System eine USV vorschalten, sind Sie vor kurzzeitigen Stromausfällen geschützt. Diese Geräte gibt es in den unterschiedlichsten Grössen.

Achten Sie beim Kauf nicht nur darauf, dass sie die erforderliche Leistung (z.B. 1000VA) bringt, sondern auch auf ausreichende Batteriekapazität. Diese können Sie so berechnen, falls konkrete Zeitangaben in der Gerätebeschreibung fehlen:

Multiplizieren Sie die Batteriespannung mit der Kapazität und teilen Sie das Ergebnis durch zwei. Beispiel:

Auf der Batterie steht 12V 110 AH. 12V * 110AH / 2 = 660VAH. Das bedeutet, dass eine USV mit dieser Batterie ein System mit 660VA Stromverbrauch eine Stunde lang versorgen kann. Würde Ihr System 1320VA Strom verbrauchen, dann würde die Batterie entsprechen für eine halbe Stunde ausreichen.

Die Batterieleistung wird in dieser Formel halbiert, weil die Elektronik der USV, die aus der Batteriespannung 230V AC macht, deutlich weniger als 100% Wirkungsgrad hat und die Leistung der Batterie schnell nachlässt. Wenn Sie die Leistung durch zwei teilen, erhalten Sie einen praxisnahen Anhaltswert.

Batterien sind nur begrenzt haltbar. Nach Ihrer normalen Lebensdauer sinkt die Kapazität rasch ab, und damit auch die maximale Zeit, die Ihre USV überbrücken kann. Sie müssen die Batterien also regelmässig prüfen und austauschen (z.B. jährlich). Das ist aber nicht möglich, wenn Sie Ihren Server 24 Stunden täglich betreiben wollen.

Es gibt gegen dieses Problem natürlich auch eine Lösung, und zwar USV, die die Batterie automatisch testen und deren Batterie Sie im laufenden Betrieb auswechseln können. Achten Sie beim Kauf auf ein solches Leistungsmerkmal, wenn für Sie ein 24 Stunden Betrieb wichtig ist.

Es gibt USV Geräte, die neben einem akustischen Alarm Ihren Server auch automatisch herunterfahren können, wenn die Batterieladung knapp wird. Diese Funktion ist sehr empfehlenswert, denn sie schützt Ihre Festplatten vor Datenverlust durch Stromausfall.

Damit sind wir auch gleich beim Thema Datensicherheit, dass ich in den nächsten Kapiteln erörtere.

3.4. Datensicherheit

Datensicherheit ist wichtig, damit Ihr System zuverlässig funktioniert und Ihre Arbeit dauerhaft Früchte tragen kann.

Beim Aufbau einer SMS Anwendung investieren Sie Zeit in die Installation von Software, sowie deren Konfiguration. Diese Arbeiten wollen Sie nicht freiwillig wiederholen. Ausserdem speichert Ihre Anwendung möglicherweise Daten, die Sie später zur Erstellung von Rechnungen benötigen. Diese Daten dürfen auf keinen Fall verloren gehen.

Sie schützen Ihre Daten, indem Sie qualitativ einwandfreie Hardware einsetzen. Doch dieser Schutz reicht nicht aus. Auch die besten Systeme können abstürzen und dabei Daten verlieren. Ausserdem haben Festplatten nur eine begrenzte, nicht genau festlegbare, Lebensdauer. Selbst Bedienungsfehler können zu Datenverlust führen.

Erstellen Sie daher regelmäßig Backups!

In der Regel werden Sie dazu in Ihre Server jeweils ein Bandlaufwerk einbauen. Ich empfehle die Verwendung von 4mm DAT Laufwerken mit DDS3 Format. Diese Laufwerke speichern 12-24GB pro Band und haben ein brauchbares Preis- Leistungsverhältnis.

Wenn Sie viele Server betreiben, bietet sich die Verwendung eines zentralen Backup-Servers an. Sie müssen dann nicht so viele Bänder manuell auswechseln.

Da auch die DAT Bänder nicht 100% verlässlich sind, verwenden Sie immer wenigstens drei Bänder im Wechsel. Sollten Sie ein Backup zurücklesen müssen, können Sie so notfalls auf ein älteres Band zurück greifen, wenn das aktuelle Band defekt ist.

Benutzen Sie die Bänder nicht zu oft. Ich führe auf jeder Kassette eine Strichliste. Wenn ein Band 20 mal benutzt wurde, ersetze ich es gegen ein neues.

Lagern Sie Datensicherungen in einem feuerfesten Tresor, oder an einem anderen Ort. Denken Sie daran, dass unerwartete Dinge passieren können, wie z.B. ein Brand, eine Explosion oder Diebstahl. Ein Band, das zusammen mit Ihrem Server vernichtet wird, nützt Ihnen nichts.

Aus Kostengründen kann auch eine Datensicherung auf CD-R oder CD-RW interessant sein. Entsprechende Software können Sie für Windows kaufen, in einer Linux Distribution ist sie bereits enthalten.

Auch bei CD's ist es angebracht, mehrere ältere Scheiben aufzubewahren, denn die aktuellste CD könnte beschädigt werden.

3.5. Redundanz

Ein Redundantes System ist durch doppelte Hardware weit gehend gegen Defekte geschützt.

Der grösste Schwachpunkt in jedem Server sind seine Festplatten. Sie fallen am häufigsten aus und deren Austausch macht zugleich am meisten Arbeit, weil alle Daten wiederhergestellt werden müssen.

Dagegen können Sie ganz einfach mehrere Festplatten in Ihren Server einbauen.

Die Klassische redundante Festplattenspeicherung ist ein RAID-5 System. Darin befinden sich meistens fünf Festplatten (jedenfalls mindestens 3), auf denen Ihre Daten verteilt gespeichert werden, und zwar zusammen mit CRC Prüfbytes. Die Prüfbytes ermöglichen eine vollständige Datenrestauration, solange nicht mehr als eine der fünf Festplatten ausfallen. RAID-5 Systeme mit 5 Festplatten nutzen 1/5 des Speicherplatzes für die Prüfbytes und 4/5 für die Nutzdaten.

Heutzutage sind die Kapazitäten gewöhnlicher Festplatten derart groß, dass RAID-5 Systeme nur noch selten Sinn machen. Bei einer SMS Anwendung ist es ganz bestimmt so.

Ich empfehle Ihnen daher eine einfache Festplattenspiegelung. Dabei speichert Ihr Server einfach alle Daten auf zwei Festplatten gleichzeitig. Sollte eine davon ausfallen, kann der Server mit der verbleibenden Festplatte weiter laufen.

Unter Solaris können Sie diese Aufgabe mit der Veritas Software lösen und bei Linux ist eine entsprechende Funktion bereits im Kernel enthalten. Für Windows sind entsprechende Controller und Software erhältlich.

Neben doppelten Festplatten ist es auch sinnvoll, Server mit doppelten Netzteilen und Lüftern einzusetzen, weil diese Teile auch weit oben in der Ausfallstatistik stehen.

Redundante Festplatten sind gut, doch sie schützen nicht vor Systemabstürzen oder Fehlbedienungen.

Wenn Sie die Zuverlässigkeit Ihrer SMS Anwendung noch weiter erhöhen möchten, können Sie einfach zwei Server gleichzeitig laufen lassen. Wenn einer davon ausfällt, läuft der andere noch weiter und sie haben keinen Service-Ausfall.

Die SMS Server Tools sind so entwickelt worden, dass Sie auf mehreren Servern gleichzeitig laufen können. Was Sie dazu zusätzlich brauchen, ist lediglich ein Fileserver, auf den alle SMS Server gleichzeitig zugreifen können. Ich nehme an, dass Sie bereits einen solchen Server besitzen.

Benutzen Sie den Fileserver nicht für andere Zwecke. Ein Fileserver, der keine andere Aufgabe hat, läuft sehr zuverlässig und ist weitgehend Wartungsfrei. Ausfälle sind an dieser Stelle kaum zu befürchten.

Wenn Sie beabsichtigen, zwei Fileserver zur Redundanz zu verwenden, dann schlage ich Ihnen vor, die Webseite http://www.linux-ha.org zu lesen. Diese Seite befasst sich mit dem Aufbau redundanter Fileserver.

3.6. Firewall

Sobald Ihre Server in einem grösseren Netzwerk eingebunden sind, oder gar ins Internet, ist die Installation einer Firewall notwendig.

Die Firewall schützt Ihre Server vor unbefugtem Zugriff, z.B. durch Hacker.

Firewalls blockieren den Zugriff durch das Netzwerk und erlauben nur bestimmte Verbindungen von gewünschten Quellen. Beispielsweise können Sie die Firewall so konfigurieren, dass sich nur der Administrator von seinem Arbeitsplatz-PC aus die Server einloggen kann, während die Internet Kunden nur Zugriff auf den Webserver haben.

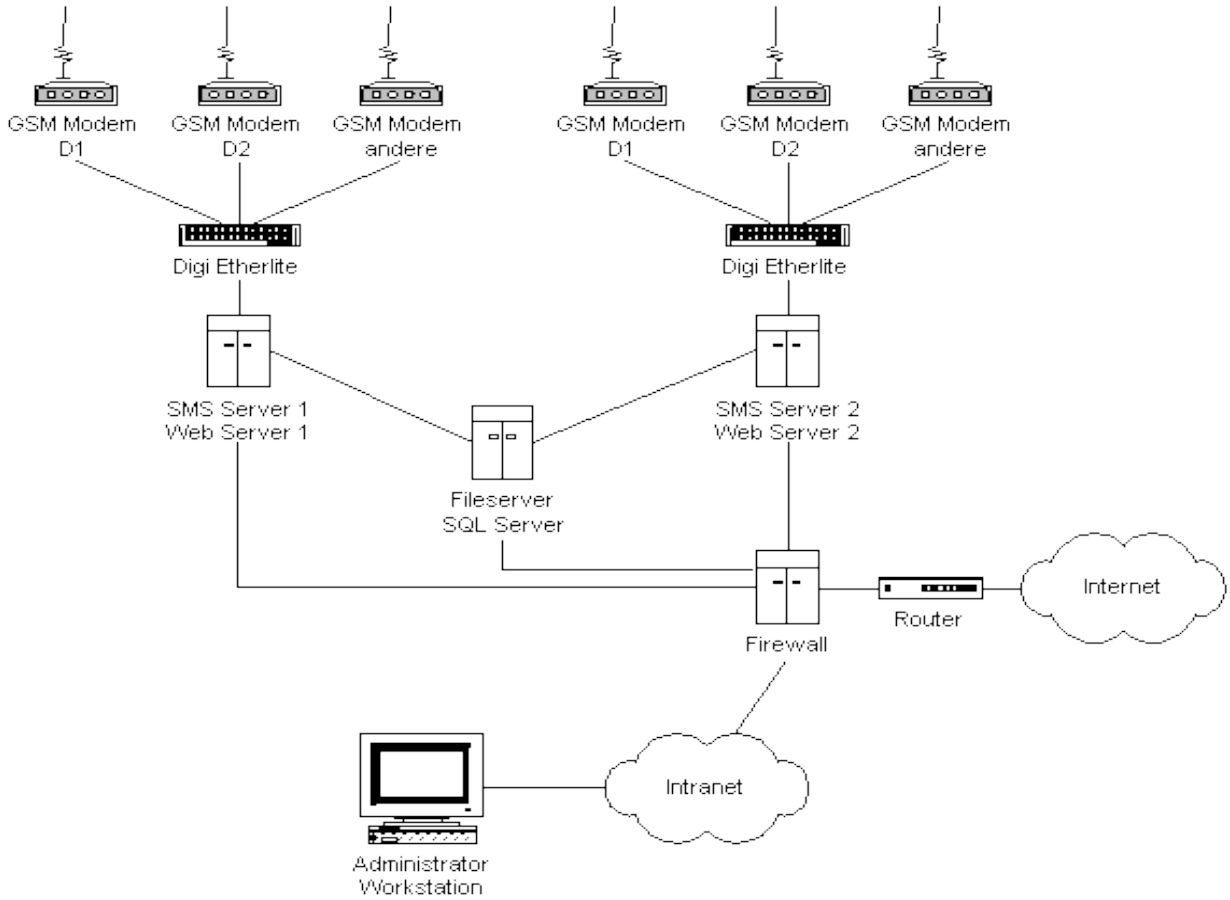
Eine einfache typische Firewall hat z.B. drei Netzwerkanschlüsse. Einer führt über einen Router ins Internet, einer führt in Ihr Firmen-Netzwerk und der dritte Anschluss führt zu Ihren Servern.

Ich rate davon ab, Firewalls unter Windows zu betreiben, weil Windows das Ziel von fast allen Angriffen ist und ausserdem immer wieder Sicherheitslücken in Windows entdeckt werden.

In den meisten Linux Distributionen ist Firewall-Software bereits enthalten, bei Solaris müssen Sie diese dazu kaufen. Die Firma SuSE bietet eine spezielle Linux Firewall-Distribution an, in der sich nichts weiter befindet, als die für Firewalls nötige Software sowie sehr komfortable Programme zur Konfiguration derselben.

3.7. Konkretes Beispielsystem

Nun möchte ich Ihnen beispielhaft einen Hardwareaufbau zeigen, mit dem Sie alle in diesem Buch beschriebenen Anwendungen zuverlässig realisieren können.



Das System ist teilweise redundant aufgebaut. Ich habe mich hier für einen Kompromiss zwischen Aufwand und Nutzen entschieden.

Die beiden SMS Server sind Dell Poweredge 350 mit Linux und einer zusätzlichen Netzwerkkarte. Die Firewall ist ebenfalls ein Dell Poweredge mit Linux und einer zusätzlichen 4-Fach Netzwerkkarte. Zur Datensicherheit sind alle drei Rechner mit zwei gespiegelten Festplatten und einem Bandlaufwerk ausgestattet.

Der Fileserver ist ein SUN V880 mit Solaris. Auch hier ist eine zusätzliche Netzwerkkarte nötig. Die Daten werden auf zwei gespiegelten Festplatten gespeichert und mit einem Bandlaufwerk gesichert.

Der Fileserver dient zugleich auch als SQL Server zum Aufbau dynamischer Webseiten und zum Speichern von Abrechnungsdaten.

Der Router führt zum Internet. Ihr Internet-Anbieter legt fest, welchen Router Sie genau verwenden werden.

Der Systemadministrator kann sich von seiner Workstation aus auf alle Server einloggen. Daher braucht man an den Servern keine eigenen Monitore und Tastaturen, nachdem die Netzwerkkarten und Betriebssysteme installiert sind.

Die Firewall schützt alle Server vor unbefugtem Zugriff aus dem Internet.

Aus Redundanzgründen sind zwei SMS Server vorhanden, die Ihre Warteschlangen, sowie Statistiken auf dem gemeinsam genutzten Fileserver speichern. Normalerweise teilen sich die beiden SMS Server die Last. Wenn einer ausfällt, werden die SMS entsprechend langsamer durch den verbliebenen Server gesendet.

Beide SMS Server haben jeweils mindestens drei Modems, jeweils eins für D1, eins für D2 und eins für alle anderen Netze. Durch diese Aufteilung können die SMS kostengünstiger gesendet werden.

Der Fileserver stellt einen Schwachpunkt im System dar, daher sollte er mit grösster Sorgfalt administriert werden und auch nicht für andere Zwecke missbraucht werden. Je weniger man am Fileserver ändert, und je weniger Funktionen er hat, umso zuverlässiger läuft er.

Das Gesamte System ist absichtlich ohne Hubs oder Switche aufgebaut, weil ein defekter Switch zu einem Totalausfall führen würde. Stattdessen werden viele einzelne Netzwerk-Kabel und Netzwerk-Anschlüsse verwendet. Bei obigen Aufbau darf - bis auf den Fileserver - jede beliebige Komponente ausfallen, ohne dass Sie einen Totalausfall bekommen.

Beide SMS Server dienen nebenbei auch als Webserver. Damit betreiben Sie Webseiten und FTP Zugänge für Ihre Kunden. Sie sollten identisch konfiguriert werden (bis auf die IP-Adresse), damit Ihre Kunden wahlweise den ersten oder zweiten verwenden können.

Es gibt Loadbalancer und Proxy Server, die alle Web-Anfragen automatisch zwischen den beiden Webservern verteilen, so dass Sie den Kunden nur eine IP-Adresse und URL nennen müssen. In diesem Beispiel wird davon kein Gebrauch gemacht.

Wenn Sie Ihre Webseiten umgestalten oder die SMS Anwendung verändern, können Sie dies in Ruhe auf einem der beiden SMS Server tun, während Ihre Kunden den anderen SMS Server für den Live-Betrieb nutzen. Der SMS Server, den Sie für Tests benutzen, kann in dieser Zeit z.B. eine andere IP-Adresse und eigene Warteschlangen-Verzeichnisse bekommen, so dass der Live-Betrieb nicht beeinträchtigt wird.

4. Basis KnowHow

In diesem Kapitel lernen Sie, wie der Versand und Empfang von SMS mit GSM Modems realisiert wird. Sie lernen die nötigen Modembefehle kennen, sowie das Datenformat der Kurzmitteilungen. Anschliessend erfahren Sie, wie die benötigten Programme installiert werden und wie man sie konfiguriert.

4.1. SMS Befehle für GSM Modems

In diesem Kapitel lernen Sie, wie der Versand und Empfang von SMS mit GSM Modems funktioniert. Das Kapitel umfasst die nötigen Modembefehle und das Datenformat der SMS.

Wenn Sie die SMS Server Tools zum Senden und Empfangen von Kurzmitteilungen verwenden, brauchen Sie sich nicht allzu tief mit der Materie dieses Kapitels zu befassen, da das Programm alles für sie erledigt.

Aber zum Troubleshooting und zum Verständnis des Programmes hilft es, die folgenden Seiten wenigstens flüchtig durchzulesen.

4.1.1. Terminalprogramm starten

Starten Sie ein beliebiges Terminalprogramm und stellen es auf folgende Parameter ein: 9600 bps, 8 Bits, 1 Stopbit, keine Parität, Hardware Handshake (RTS/CTS).

Unter Windows können zie z.B. das Programm Hyperterminal benutzen, und unter Linux das Programm Minicom. Beide sind Menügeführt.

Solaris:

Unter Solaris können sie das Programm tip benutzen. Dazu müssen Sie vorher jedoch die Datei /etc/remote um diese Zeilen ergänzen:

```
porta:dv=/dev/cua/a:br#9600
portb:dv=/dev/cua/b:br#9600
```

Wenn Sie noch mehr serielle Ports an Ihrem SUN Rechner haben, tragen Sie auch diese entsprechend ein. Der Name vor dem Doppelpunkt ist frei erfunden. Dahinter kommt der Device-Name und die Bitrate.

Nachdem Sie die Konfigurationsdatei angepasst haben, können Sie das Programm durch den Befehl

tip porta

starten, um den ersten seriellen Port des Rechners zu öffnen.

Sie beenden tip durch Eingabe der Zeichenfolge **~.**

4.1.2. GSM Modembefehle

Für SMS Anwendungen brauchen Sie keine Verbindung zur SMSC aufzubauen. GSM Modems senden und empfangen Kurzmitteilungen durch den Signalisierungskanal, der nach dem Einbuchen ins Netz ständig verfügbar ist. Sie werden also vergeblich nach ATD-Befehlen suchen.

Allgemeine Regeln zu Modembefehlen:

Alle Modembefehle beginnen mit AT. Nach Eingabe des Befehls müssen Sie die Enter Taste drücken.

Man kann mehrere Befehle auf einmal in eine einzige Zeile schreiben, dann schreibt man das AT aber nur einmal an den Beginn der Zeile.

Zwischen AT und zwischen mehreren Befehlen dürfen Leerzeichen eingefügt werden.

Wenn Sie ein gebrauchtes Modem benutzen, sorgen Sie mit dem Befehl AT &F &W dafür, dass das Modem beim Reset auf Werkseinstellungen steht. Den Befehl brauchen Sie nur ein einziges mal eingeben.

Für SMS Anwendungen brauchen Sie nur einen kleinen Bruchteil aller Modembefehle, und zwar folgende:

AT Z

führt einen Reset durch. Das Modem initialisiert sich so, als ob es aus- und wieder eingeschaltet worden wäre.

AT E0

schaltet das Echo aus. Sie sehen danach nicht mehr, was sie eingeben. Dieser Befehl mag zunächst sinnlos erscheinen, aber für Programmierer kann er das Programmdesign deutlich vereinfachen. Das Gegenteil ist übrigens AT E1.

AT +CPIN?

zeigt an, ob das Modem nach der Eingabe der PIN verlangt. Die PIN wird wie beim Handy benötigt, um die SIM Karte benutzen zu können und ist somit Voraussetzung für alle anderen GSM Befehle.

Das Modem Antwortet auf eine der folgenden Weisen:

- READY bedeutet, dass die PIN bereits eingegeben wurde.
- SIM PIN bedeutet, dass Sie die PIN eingeben müssen (siehe nächster Befehl).
- SIM PUK bedeutet, dass die PIN wegen mehrfacher Fehleingabe gesperrt wurde.
- ERROR bedeutet, dass es ein Problem mit der SIM Karte gibt. Die Bedienungsanleitung Ihres Modems erklären genau, was die Zahl hinter ERROR bedeutet.
- Es gibt noch andere mögliche Antworten, doch die habe ich in der Praxis noch nie zu sehen bekommen. Daher beschreibe ich sie hier auch nicht. Schauen Sie bei Bedarf in das Handbuch des Modems.

AT +CPIN=xxxx

Mit diesem Befehl geben Sie die PIN ein. Unmittelbar danach bucht sich das Modem ins GSM Netz ein. Den Befehl brauchen Sie nach dem Einschalten der Stromversorgung nur einmal einzugeben.

AT +CPIN=yyyyyyyy,xxxx

Mit diesem Befehl ändern Sie die PIN. Das geht auch, wenn die alte PIN wegen mehrfacher Fehleingabe gesperrt wurde. Anstelle von yyyyyyyy geben Sie bitte die PUK ein, und anstelle von xxxx die neue gewünschte PIN. Zur Info: Die PUK ist ein längerer Sicherheitscode, den Sie zusammen mit der SIM Karte auf Papier erhalten haben.

AT +CSQ

zeigt die Empfangsqualität an. Als Antwort zeigt das Modem zwei Zahlen an. Die erste bezieht sich auf die Signalstärke des Empfangs und sollte immer zwischen 2 und 31 liegen. 0 bedeutet, dass kein Empfang möglich ist, während 31 den maximal zulässigen Pegel darstellt. Die zweite Zahl bezieht sich auf die Bit-Fehlerrate. Sie ist bei SMS Anwendungen nutzlos, weil sie sich nur auf Verbindungen bezieht. Aber zum Versand und Empfang von SMS bauen wir keine Verbindungen auf.

AT +CREG?

zeigt an, ob das Modem ins GSM Netz eingebucht ist. Die Antwort 0,1 ist positiv. Alle anderen Antworten zeigen an, dass das Modem noch nicht eingebucht ist. Solange das Modem nicht mit 0,1 antwortet, können Sie zwangsläufig keine SMS übertragen.

Im nächsten Kapitel zeige ich Ihnen Befehle, die Sie zum Senden und Empfangen von SMS brauchen.

4.1.3. SMS Modembefehle

Nachdem Sie ein paar Befehle kennengelernt haben, mit denen Sie das Modem initialisieren und prüfen, ob es ins Netz eingebucht ist, sollen Sie nun die Befehle zum Empfangen und Senden von SMS erfahren.

AT +CMGF=0

sagt dem Modem, dass alle SMS im PDU Format zum Computer übertragen werden. Das Format beschreibe ich im nächsten Kapitel. Alternativ gibt es noch das Ascii Format (AT+CMGF=1), welches für viele Anwendungen nicht geeignet ist. Im Ascii Modus kann das Modem nämlich nur 140 statt 160 Buchstaben senden und es kann keine Binärdaten (Klingeltöne, Logos, etc) senden. Manche Modem unterstützen den Ascii-Modus gar nicht.

Die folgenden Befehle brauchen Sie zum Empfangen:

AT +CPMS="SM"

legt fest, dass das Modem empfangene Kurzmitteilungen auf der SIM Karte speichern soll. Wenn Ihr Modem den Befehl nicht unterstützt, dann kann es sowieso nur auf der SIM Karte speichern. Der Parameter "ME" ist das Gegenteil und speichert Kurzmitteilungen im Arbeitsspeicher des Modems.

Viele Leute berichteten mir, dass man keine SMS aus dem ME-Speicher auslesen kann. Meine eigenen Tests waren aber erfolgreich. Wenn Sie damit ebenfalls Probleme haben, arbeiten Sie einfach nur mit dem SM-Speicher, dann klappt es auf jeden Fall.

AT +CPMS?

fragt ab, wie weit der Speicherplatz für SMS belegt ist. Das Modem antwortet beispielsweise mit

```
+CPMS: "SM",3,20,"SM",0,20,"SM",0,20
```

Der erste Block bezieht sich auf die empfangenen Nachrichten. Der zweite Block bezieht sich auf gesendete Nachrichten. Der dritte Block wird nur benutzt, wenn Sie das Modem auf eine besondere Art initialisieren, auf die ich in diesem Buch nicht eingehe.

Die Abkürzung in den Anführungsstrichen zeigt an, wo die Nachrichten gespeichert werden (im obigen Beispiel bedeutet SM = SIM-Karte). Danach folgt die Anzahl der tatsächlich gespeicherten SMS und die Grösse dieses Speicherplatzes. Die Anzahl der SIM Speicherplätze hängen von der SIM Karte selbst ab.

Für Sie sind nur die ersten beiden Zahlen "3,20" interessant.

AT +CMGL=x

listet alle gespeicherten SMS auf. Anstelle von x setzen Sie bitte einen der folgenden Werte ein:

0 ungelesene Nachrichten. Das Auflisten zählt nicht als lesen

1 empfangene Nachrichten

2 nicht gesendete, gespeicherte Nachrichten

3 gesendete, gespeicherte Nachrichten

4 alle Nachrichten

AT +CMGR=x

liest eine empfangene Nachricht aus. Geben Sie anstelle von x die Nummer des Speicherplatzes an, den Sie auslesen möchten. Wenn der Speicher leer ist, antworten manche Modems mit einer leeren Antwort und manche Modems antworten mit ERROR. Sie können keine gesendeten Nachrichten auslesen, dafür gibt es keinen Befehl.

Die Nachricht wird in folgenden Format angezeigt:

```
+CMGR: status,name,laenge
daten
OK
```

Status ist ein einstelliger Wert, wie beim AT+CMGL Befehl. Name ist der Absendername, sofern er im Telefonbuch steht. Die Längenangabe brauchen Sie nicht. Das Format der Daten wird im nächsten Kapitel erklärt.

Soweit mir bekannt ist, zeigen nur Siemens Modems den Namen an. Bei anderen Modems entfällt dieser Parameter.

AT +CMGD=x

löscht eine empfangene Nachricht. Geben Sie anstelle von x die Nummer des Speicherplatzes an.

Die folgenden Befehle brauchen Sie zum Senden:

AT +CSCA="+xxxxxxxxxxx"

Mit diesem Befehl stellen Sie die SMS Zentrale ein, über die Sie SMS senden möchten. Geben Sie die Nummer im Internationalen Format an, z.B. +491722270000. Für den Empfang ist diese Einstellung unrelevant, denn jede beliebige SMS Zentrale kann Nachrichten zum Modem senden, egal, was Sie hier einstellen.

AT +CMGS=x

Mit diesem Befehl senden Sie eine SMS. Nach Eingabe des Befehls erscheint ein ">" Zeichen als Eingabeaufforderung für die Daten. Geben Sie dann die Daten ein und drücken Sie zum Schluss Strg-Z.

Das Datenformat ist im nächsten Kapitel erklärt. Geben Sie anstelle von x die Länge der Daten ein, und zwar nach der Formel x=(Anzahl der Zeichen/2)-1. Wenn Ihre Daten z.B. 30 Zeichen Lang sind, dann ist x=14.

4.1.4. PDU Datenformat (Empfangen)

Im vorherigen Kapitel haben Sie die Modembefehle zum Senden und Empfangen von Kurzmitteilungen kennen gelernt. Die Texte werden leider weder im Klartext eingegeben, noch angezeigt. Nun zeige ich Ihnen, wie die Texte codiert werden.

In den Zahlenketten gehören immer zwei Ziffern zusammen. Jedes Pärchen ist ein Byte in Hexadezimaler Darstellung.

Eine Empfangene SMS sieht z.B. so aus:

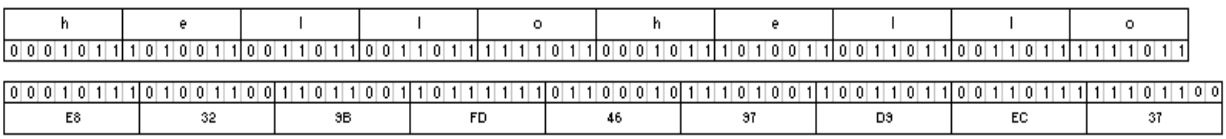
```
07919471227200000040C919471123254F60000031092516195040AE8329BFD4697D9EC37
```

07	Länge der SMSC Nummer, hier 7 Bytes. Formatangabe und Nummer werden gezählt.
91	Format der SMSC Nummer, 91=International, A1=National.
94 71 22 72 00 00	SMSC Nummer (+)491722270000.
04	Bit 6: User Data Header Indikator. Ist 1, wenn die Daten einen Header beinhalten. Nur bei 8 bit
	Bit 5: Ist 1, wenn der Absender einen Status-Report gewünscht hat.
	Bit 2: Ist 0, wenn noch weitere zusammenhängende SMS empfangen werden.
	Bit 1+0: Ist 00, um eine empfangene SMS anzuzeigen.10 wäre ein Status Report.
0C	Länge der Absender-Nummer, hier 12 Ziffern. Die Formatangabe wird nicht mitgezählt.
91	Format der Absender Nummer, 91=International, A1=National, D0=Alphanumerisch.
94 71 12 32 54 F6	Absender (+)49172123456.
00	Protokoll-Art. 00=normale SMS.
00	Bit 3+2: Datencodierung: 00=7bit Text, 01=8bit Daten, 10=Unicode Text
	Bit 1+0: Klasse: 00=Flash-SMS, 01=normale SMS
03 10 92 51 61 95	Sendezeit in der Reihenfolge "Jahr Monat Tag Stunde Minute Sekunde". Jede zweite Ziffer ist vertauscht. Hier steht also 29.01.03 15:16:59
04	Zeitzone in viertel Stunden. Negative Werte sind Möglich und werden durch Addition von 80 dargestellt. 04 bedeutet GMT+1 Stunde
0A	Anzahl der Zeichen (bei 7-Bit) oder Anzahl der Bytes (bei 8-Bit)
E8329BFD4697D9EC37	Text oder Daten. In diesem Fall "hellohello".

Telefonnummern werden mit F aufgefüllt, wenn die Anzahl der Ziffern ungerade ist. Manche SMSC zählen das F bei der Längenangabe mit, und manche zählen es nicht mit. Die Ziffern der Telefonnummern werden vertauscht dargestellt, wie man am obigen Beispiel gut sehen kann.

Alphanumerische Absendernamen werden wie 7-Bit Text codiert. Die Längenangabe davor gibt an, wie viele Hex-Ziffern der Absendername lang ist (also im Grunde genommen wie bei numerischen Absendern).

Texte bestehen aus Zeichen mit 7 Bit Länge. Diese Bits werden aneinander gereiht und dann wieder zerlegt, aber in Bytes (mit jeweils 8 Bit). Und zwar so:



Ich habe die Binärzahlen absichtlich falsch herum dargestellt, weil man das Codierungsschema so besser verstehen kann. Links ist immer das Bit 0 und rechts ist das Bit 6 bzw. 7. Wenn am Ende nicht genügend Bits übrig sind, um ein Byte voll zu bekommen, wird einfach mit 0-Bits aufgefüllt.

Wenn Sie sich dieses Codierschema genau ansehen, werden Sie schnell verstehen, warum 8-Bit Kurzmitteilungen nur 140 Bytes lang sein dürfen, aber Textmitteilungen (7-Bit) 160 Zeichen.

Für die Nummernformate, Flags, Protokoll-Art und Datencodierung gibt es weitaus mehr Werte, als ich Ihnen hier vorgestellt habe. Diese werden in Deutschland aber nicht unterstützt. Wenn Sie sich noch detaillierter informieren möchten, sollten Sie die Spezifikationen ETSI TS 100 900 studieren.

4.1.5. PDU Datenformat (Status Report)

Wenn man Kurzmitteilungen sendet, kann man dabei einen Status Report anfordern. Das SMSC benachrichtigt dann den Absender über den Status. Sie erfahren so, ob die SMS ausgeliefert wurde, oder nicht.

In den Zahlenketten gehören immer zwei Ziffern zusammen. Jedes Pärchen ist ein Byte in Hexadezimaler Darstellung.

Ein Statusreport sieht z.B. so aus:

```
07919471227200000 2010C919471123254F6031092516195040310925171050400...
```

07	Länge der SMSC Nummer, hier 7 Bytes. Formatangabe und Nummer werden gezählt
91	Format der SMSC Nummer, 91=International, A1=National
94	SMSC Nummer (+)491722270000
02	Bit 2: Ist 0, wenn noch weitere zusammenhängende SMS empfangen werden.
	Bit 1+0: Ist 10 als Kennzeichen für einen Status Report
01	ID-Nummer der gesendeten SMS.
0C	Länge der Empfänger-Nummer, hier 12 Ziffern. Die Formatangabe wird nicht mitgezählt.
91	Format der Empfänger-Nummer, 91=International, A1=National, D0=Alphanumerisch.
94 71 12 32 54 F6	Empfänger der SMS, zu der dieser Report gehört. Hier (+)49172123456.
04	Zeitzone in viertel Stunden. Negative Werte sind Möglich und werden durch Addition von 80 dargestellt. 04 bedeutet GMT+1 Stunde.
03 10 92 51 71 05	Auslieferungszeit der SMS beim Endgerät in der Reihenfolge "Jahr Monat Tag Stunde Minute Sekunde“. Jede zweite Ziffer ist vertauscht. Hier steht also 29.01.03 15:17:50. Wenn die SMS noch nicht ausgeliefert wurde, ist das Datum 00.00.00 00:00:00.
04	Zeitzone in viertel Stunden. Negative Werte sind Möglich und werden durch Addition von 80 dargestellt. 04 bedeutet GMT+1 Stunde.
00	Status
...	An dieser Stelle können weitere Daten kommen, deren Bedeutung vom SMSC abhängt.

Telefonnummern werden mit F aufgefüllt, wenn die Anzahl der Ziffern ungerade ist. Manche SMSC zählen das F bei der Längenangabe mit, und manche zählen es nicht mit. Die Ziffern der Telefonnummern werden vertauscht dargestellt, wie man am obigen Beispiel gut sehen kann.

Alphanumerische Absendernamen werden wie 7-Bit Text codiert. Die Längenangabe davor gibt an, wie viele Hex-Ziffern der Absendername lang ist (also im Grunde genommen wie bei numerischen Absendern).

Als Status gibt es folgende Werte (dezimal):

0	Ok, Nachricht wurde ausgeliefert
1	Ok, das Endgerät konnte den Empfang der Nachricht nicht bestätigen
2	Ok, Die Nachricht wurde vom SMSC ersetzt
32	Versuche weiter, das Netz ist nicht bereit
33	Versuche weiter, das Endgerät nicht bereit
34	Versuche weiter, keine Antwort vom Endgerät
35	Versuche weiter, Dienst abgelehnt
36	Versuche weiter, Qualität des Dienstes nicht verfügbar
37	Versuche weiter, Fehler im Endgerät
64	Fehler, Prozedurfehler beim Empfänger
65	Fehler, Ziel nicht kompatibel
66	Fehler, Nachricht vom Endgerät verweigert
67	Fehler, nicht ermittelbar
68	Fehler, Qualität des Dienstes nicht verfügbar
69	Fehler, keine Zusammenarbeit möglich
70	Fehler, Gültigkeit der Nachricht abgelaufen
71	Fehler, Nachricht wurde vom Absender gelöscht (das geht nur bei Direktverbindung zur SMSC)
72	Fehler, Nachricht wurde vom SMSC Administrator gelöscht
73	Fehler, Nachricht existiert nicht

96	Fehler, Netz nicht bereit
97	Fehler, Endgerät nicht bereit
98	Fehler, keine Antwort vom Endgerät
99	Fehler, Dienst abgelehnt
100	Fehler, Qualität des Dienstes nicht verfügbar
101	Fehler, Fehler im Endgerät

4.1.6. PDU Datenformat (Senden)

In den Zahlenketten gehören immer zwei Ziffern zusammen. Jedes Pärchen ist ein Byte in Hexadezimaler Darstellung.

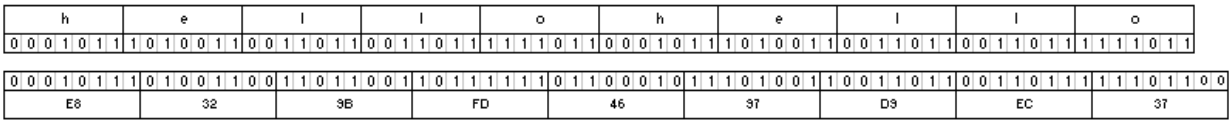
Eine zu sendende SMS sieht z.B. so aus:

```
0001000C919471123254F600000AE8329BFD4697D9EC37
```

00	Länge der SMSC Nummer, 00 bedeutet, dass die SMSC Nummer dem Modem bereits bekannt ist
01	Bit 6: User Data Header Indikator. Ist 1, wenn die Daten einen Header beinhalten. Nur bei 8-Bit.
	Bit 5: Ist 1, wenn ein Status Report gewünscht wird.
	Bit 4+3: Format der Gültigkeitsdauer der SMS. 00=möglichst lange
	Bit 2: Wenn 1, verwirft das SMSC Nachrichten mit gleicher ID-Nummer in der Warteschlange.
	Bit 1+0: Ist 01, um eine zu sendende SMS anzuzeigen.
00	ID-Nummer der Nachricht. Bei 00 vergibt das Modem automatisch eine Nummer.
0C	Länge der Empfänger-Nummer, hier 12 Ziffern. Die Formatangabe wird nicht mit gezählt.
91	Format der Empfänger-Nummer, 91=International, A1=National, D1=Alphanumerisch.
94 71 12 32 54 F6	Empfänger der SMS, zu der dieser Report gehört. Hier (+)49172123456.
00	Protokoll-Art. 00=normale SMS.
00	Bit 7-2: Datencodierung: 111100=7bit Text, 111101=8bit Daten, 000110=Unicode Text.
	Bit 1+0: Klasse: 00=Flash-SMS, 01=normale SMS
(...)	Gültigkeitsdauer der SMS. Entfällt hier, weil in den Bits 4+3 oben 00 steht.
0A	Anzahl der Zeichen (bei 7-Bit) oder Anzahl der Bytes (bei 8-Bit)
E8329BFD4697D9EC37	Text oder Daten. In diesem Fall "hellohello".

Telefonnummern werden mit F aufgefüllt, wenn die Anzahl der Ziffern ungerade ist. Die Ziffern der Telefonnummern werden vertauscht dargestellt, wie man am obigen Beispiel gut sehen kann.

Texte bestehen aus Zeichen mit 7 Bit Länge. Diese Bits werden aneinander gereiht und dann wieder zerlegt, aber in Bytes (mit jeweils 8 Bit). Und zwar so:



Ich habe die Binärzahlen absichtlich falsch herum dargestellt, weil man das Codierungsschema so besser verstehen kann. Links ist immer das Bit 0 und rechts das Bit 6 bzw. 7. Wenn am Ende nicht genügend Bits übrig sind, um ein Byte voll zu bekommen, wird einfach mit 0-Bits aufgefüllt.

Wenn Sie sich dieses Codierschema genau ansehen, werden Sie schnell verstehen, warum 8-Bit Kurzmitteilungen nur 140 Bytes lang sein dürfen, aber Textmitteilungen (7-Bit) 160 Zeichen.

Für die Gültigkeitsdauer (...) der Nachricht gibt es vier Formate:

00	ohne Zeitangabe, der Defaultwert des SMSC gilt.
01	Erweitertes Format, siehe ETSI TS 100 900 Spezifikation
10	Absolutes Format. Wie das Datum/Zeit Feld im Status Report oder in empfangenen SMS. 7 Bytes lang.
11	Relatives Format. 1 Byte.

Beim relativen Format wird die Gültigkeit der Nachricht in einem Byte angegeben:

0-143	(Wert+1)*5 Minuten. Ergibt 5 Minuten bis 12 Stunden.
144-167	12 Stunden + (Wert-143)*30 Minuten. Ergibt 12,5 bis 24 Stunden.
168-196	Wert-166 Tage. Ergibt 2 bis 30 Tage.
197-255	Wert-192 Wochen. Ergibt 5 bis 63 Wochen.

In Deutschland ist die Gültigkeitsdauer auf 2 Tage begrenzt. Grössere Werte werden vom SMSC ignoriert.

Für die Nummernformate, Flags, Protokoll-Art und Datencodierung gibt es weitaus mehr Werte, als ich Ihnen hier vorgestellt habe. Diese werden in Deutschland aber nicht unterstützt. Wenn Sie sich noch detaillierter informieren möchten, sollten Sie die Spezifikationen ETSI TS 100 900 studieren.

4.1.7. Zeichensatz

Handies haben nicht ganz den gleichen Zeichensatz, wie Computer. Die folgende Tabelle listet alle vom Handy darstellbaren Zeichen auf, sowie die entsprechenden Zeichencodes auf Computern im ISO-8859-1 Zeichensatz.

Das Euro-Zeichen und einige andere werden beim Handy durch zwei aufeinanderfolgende 7-Bit Zahlen dargestellt! Manche Länder kennen landesspezifische Erweiterungen, die hier nicht aufgelistet sind.

Hex	Dez	Name	Zeichen	ISO-8859-1
0x00	0	COMMERCIAL AT	@	64
0x01	1	POUND SIGN	£	163
0x02	2	DOLLAR SIGN	$	36
0x03	3	YEN SIGN	¥	165
0x04	4	LATIN SMALL LETTER E WITH GRAVE	è	232
0x05	5	LATIN SMALL LETTER E WITH ACUTE	é	233
0x06	6	LATIN SMALL LETTER U WITH GRAVE	ù	249
0x07	7	LATIN SMALL LETTER I WITH GRAVE	ì	236
0x08	8	LATIN SMALL LETTER O WITH GRAVE	ò	242
0x09	9	LATIN CAPITAL LETTER C WITH CEDILLA	Ç	199
0x0A	10	LINE FEED		10
0x0B	11	LATIN CAPITAL LETTER O WITH STROKE	Ø	216
0x0C	12	LATIN SMALL LETTER O WITH STROKE	ø	248
0x0D	13	CARRIAGE RETURN		13
0x0E	14	LATIN CAPITAL LETTER A WITH RING ABOVE	Å	197
0x0F	15	LATIN SMALL LETTER A WITH RING ABOVE	å	229
0x10	16	GREEK CAPITAL LETTER DELTA	Δ	
0x11	17	LOW LINE	_	95
0x12	18	GREEK CAPITAL LETTER PHI	Φ	
0x13	19	GREEK CAPITAL LETTER GAMMA	Γ	
0x14	20	GREEK CAPITAL LETTER LAMBDA	Λ	
0x15	21	GREEK CAPITAL LETTER OMEGA	Ω	
0x16	22	GREEK CAPITAL LETTER PI	Π	
0x17	23	GREEK CAPITAL LETTER PSI	Ψ	
0x18	24	GREEK CAPITAL LETTER SIGMA	Σ	
0x19	25	GREEK CAPITAL LETTER THETA	Θ	
0x1A	26	GREEK CAPITAL LETTER XI	Ξ	
0x1B	27	ESCAPE TO EXTENSION TABLE		
0x1B0A	27 10	FORM FEED		12
0x1B14	27 20	CIRCUMFLEX ACCENT	^	94
0x1B28	27 40	LEFT CURLY BRACKET	{	123
0x1B29	27 41	RIGHT CURLY BRACKET	}	125
0x1B2F	27 47	REVERSE SOLIDUS (BACKSLASH)	\	92
0x1B3C	27 60	LEFT SQUARE BRACKET	[	91
0x1B3D	27 61	TILDE	~	126
0x1B3E	27 62	RIGHT SQUARE BRACKET	]	93
0x1B40	27 64	VERTICAL BAR	|	124
0x1B65	27 101	EURO SIGN	€	164
0x1C	28	LATIN CAPITAL LETTER AE	Æ	198
0x1D	29	LATIN SMALL LETTER AE	æ	230
0x1E	30	LATIN SMALL LETTER SHARP S (German)	ss	223
0x1F	31	LATIN CAPITAL LETTER E WITH ACUTE	É	201
0x20	32	SPACE		32

Hex	Dez	Name	Zeichen	ISO-8859-1
0x21	33	EXCLAMATION MARK	!	33
0x22	34	QUOTATION MARK	"	34
0x23	35	NUMBER SIGN	#	35
0x24	36	CURRENCY SIGN	¤	164
0x25	37	PERCENT SIGN	%	37
0x26	38	AMPERSAND	&	38
0x27	39	APOSTROPHE	'	39
0x28	40	LEFT PARENTHESIS	(	40
0x29	41	RIGHT PARENTHESIS	)	41
0x2A	42	ASTERISK	*	42
0x2B	43	PLUS SIGN	+	43
0x2C	44	COMMA	,	44
0x2D	45	HYPHEN-MINUS	-	45
0x2E	46	FULL STOP	.	46
0x2F	47	SOLIDUS (SLASH)	/	47
0x30	48	DIGIT ZERO	0	48
0x31	49	DIGIT ONE	1	49
0x32	50	DIGIT TWO	2	50
0x33	51	DIGIT THREE	3	51
0x34	52	DIGIT FOUR	4	52
0x35	53	DIGIT FIVE	5	53
0x36	54	DIGIT SIX	6	54
0x37	55	DIGIT SEVEN	7	55
0x38	56	DIGIT EIGHT	8	56
0x39	57	DIGIT NINE	9	57
0x3A	58	COLON	:	58
0x3B	59	SEMICOLON	;	59
0x3C	60	LEss-THAN SIGN	<	60
0x3D	61	EQUALS SIGN	=	61
0x3E	62	GREATER-THAN SIGN	>	62
0x3F	63	QUESTION MARK	?	63
0x40	64	INVERTED EXCLAMATION MARK	¡	161
0x41	65	LATIN CAPITAL LETTER A	A	65
0x42	66	LATIN CAPITAL LETTER B	B	66
0x43	67	LATIN CAPITAL LETTER C	C	67
0x44	68	LATIN CAPITAL LETTER D	D	68
0x45	69	LATIN CAPITAL LETTER E	E	69
0x46	70	LATIN CAPITAL LETTER F	F	70
0x47	71	LATIN CAPITAL LETTER G	G	71
0x48	72	LATIN CAPITAL LETTER H	H	72
0x49	73	LATIN CAPITAL LETTER I	I	73
0x4A	74	LATIN CAPITAL LETTER J	J	74
0x4B	75	LATIN CAPITAL LETTER K	K	75
0x4C	76	LATIN CAPITAL LETTER L	L	76
0x4D	77	LATIN CAPITAL LETTER M	M	77
0x4E	78	LATIN CAPITAL LETTER N	N	78
0x4F	79	LATIN CAPITAL LETTER O	O	79
0x50	80	LATIN CAPITAL LETTER P	P	80
0x51	81	LATIN CAPITAL LETTER Q	Q	81

Hex	Dez	Name	Zeichen	ISO-8859-1
0x52	82	LATIN CAPITAL LETTER R	R	82
0x53	83	LATIN CAPITAL LETTER S	S	83
0x54	84	LATIN CAPITAL LETTER T	T	84
0x55	85	LATIN CAPITAL LETTER U	U	85
0x56	86	LATIN CAPITAL LETTER V	V	86
0x57	87	LATIN CAPITAL LETTER W	W	87
0x58	88	LATIN CAPITAL LETTER X	X	88
0x59	89	LATIN CAPITAL LETTER Y	Y	89
0x5A	90	LATIN CAPITAL LETTER Z	Z	90
0x5B	91	LATIN CAPITAL LETTER A WITH DIAERESIS	Ä	196
0x5C	92	LATIN CAPITAL LETTER O WITH DIAERESIS	Ö	214
0x5D	93	LATIN CAPITAL LETTER N WITH TILDE	Ñ	209
0x5E	94	LATIN CAPITAL LETTER U WITH DIAERESIS	Ü	220
0x5F	95	SECTION SIGN	§	167
0x60	96	INVERTED QUESTION MARK	¿	191
0x61	97	LATIN SMALL LETTER A	a	97
0x62	98	LATIN SMALL LETTER B	b	98
0x63	99	LATIN SMALL LETTER C	c	99
0x64	100	LATIN SMALL LETTER D	d	100
0x65	101	LATIN SMALL LETTER E	e	101
0x66	102	LATIN SMALL LETTER F	f	102
0x67	103	LATIN SMALL LETTER G	g	103
0x68	104	LATIN SMALL LETTER H	h	104
0x69	105	LATIN SMALL LETTER I	i	105
0x6A	106	LATIN SMALL LETTER J	j	106
0x6B	107	LATIN SMALL LETTER K	k	107
0x6C	108	LATIN SMALL LETTER L	l	108
0x6D	109	LATIN SMALL LETTER M	m	109
0x6E	110	LATIN SMALL LETTER N	n	110
0x6F	111	LATIN SMALL LETTER O	o	111
0x70	112	LATIN SMALL LETTER P	p	112
0x71	113	LATIN SMALL LETTER Q	q	113
0x72	114	LATIN SMALL LETTER R	r	114
0x73	115	LATIN SMALL LETTER S	s	115
0x74	116	LATIN SMALL LETTER T	t	116
0x75	117	LATIN SMALL LETTER U	u	117
0x76	118	LATIN SMALL LETTER V	v	118
0x77	119	LATIN SMALL LETTER W	w	119
0x78	120	LATIN SMALL LETTER X	x	120
0x79	121	LATIN SMALL LETTER Y	y	121
0x7A	122	LATIN SMALL LETTER Z	z	122
0x7B	123	LATIN SMALL LETTER A WITH DIAERESIS	ä	228
0x7C	124	LATIN SMALL LETTER O WITH DIAERESIS	ö	246
0x7D	125	LATIN SMALL LETTER N WITH TILDE	ñ	241
0x7E	126	LATIN SMALL LETTER U WITH DIAERESIS	ü	252
0x7F	127	LATIN SMALL LETTER A WITH GRAVE	à	224

4.1.8. Praktische Beispiele

Zum Abschluss möchte ich Ihnen beispielhaft zeigen, wie man sich selbst eine SMS schickt.

Dies können sie sowohl mit einem GSM Modem, als auch mit einem Mobiltelefon nachvollziehen.

Ich verwende für das Beispiel ein Sharp GX-10 Mobiltelefon mit Infrarotschnittstelle unter Linux. Die Treiber für die Infrarotschnittstelle habe ich bereits geladen und sichergestellt, dass der Computer das Telefon richtig erkannt hat.

Nun starte ich ein beliebiges Terminalprogramm und gebe die Befehle ein, die Sie in den vorherigen Kapiteln kennengelernt haben.

atz
OK

at+cpin?
+CPIN: READY
OK

at+csq
+CSQ: 25,0
OK

at+creg?
+CREG: 0,1
OK

Bis hierhin war alles nur Spielerei. Ich habe das Handy zurückgesetzt, die PIN geprüft, die Empfangsqualität geprüft und kontrolliert, ob das Handy ins Netz eingebucht ist. Natürlich sind diese Befehle bei Handies überflüssig, denn wenn nicht alles Ok wäre, würde im Display eine entsprechende Fehlermeldung erscheinen. Da GSM Modems aber kein Display haben, werden Sie diese Befehle dort brauchen.

Jetzt schicke ich mir selbst eine SMS:

at+cmgf=0
OK

at+csca="+491722270000"
OK

at+cmgs=22
> **0001000C919471123254F600000AE8329BFD4697D9EC37**<Strg-Z>
+CMGS: 106
OK

+CMTI: "SM", 1

Zuerst sagte ich dem Handy, dass ich Nachrichten im PDU Format schicken möchte. Der nächste Befehl legt das SMSC fest, über das ich senden möchte. Hätte ich den Befehl weg gelassen, würde das Handy entscheiden, welches SMSC es nehmen wird.

Mit dem dritten Befehl habe ich die SMS gesendet. Dabei gab ich die Länge der PDU-Daten an, nach der Formel:

(Anzahl der Zeichen /2)-1

Zum Schluss habe ich die PDU Daten eingegeben und Strg-Z gedrückt. Wenige Sekunden später hat das Handy den Versand bestätigt und angezeigt, dass die Nachricht ID-Nummer 106 bekommen hat.

Noch ein paar Sekunden später hat mich das Handy mit +CTMI: "SM",1 benachrichtigt, dass es eine SMS empfangen hat. Sie befindet sich auf der SIM Karte (SM) Speicherplatz 1.

Diese empfangene SMS möchte ich natürlich auch gleich lesen. Und das geht so:

at+cpms?
+CPMS: "SM",1,10,"SM",1,10,"SM",1,10
OK

at+cmgr=1
+CMGR: 0,,28
07919471227200000 40C919471123254F60000301061024051800AE8329BFD4697D9EC37
OK

at+cmgd=1
OK

Der erste Befehl war wieder nicht nötig. Ich habe damit geprüft, wieviele SMS im Handy gespeichert sind. Die (erste) Angabe "SM",1,10 bedeutet, dass auf der SIM Karte eine Nachricht gespeichert ist und maximal 10 Nachrichten darauf passen.

Der zweite Befehl liest die Nachricht aus Speicherplatz 1 aus. Sie hätte auch in einem anderen Speicherplatz liegen können, aber das Handy hat mich ja zuvor darüber informiert, wo die gerade empfangene Nachricht zu finden ist. Hätte ich das nicht gewusst, dann hätte ich alle 10 Speicherpläte durchprobieren müssen oder den Befehl at+cmgl benutzen müssen.

Nachdem ich die Nachricht gelesen habe, löschte ich sie mit dem letzten Befehl.

Sicher ist Ihnen nicht entgangen, dass der Nachrichtentext "hellohello" ist, wie ich Ihnen in den vorherigen Kapiteln als Beispiel gezeigt habe.

Tip: Geben Sie die PDU Daten in das Programm PDU-Spy ein und sehen Sie sich an, wie das Programm die Daten entschlüsselt. Vergleichen Sie die Ausgabe von PDU-Spy mit den vorherigen Kapiteln.

4.2. Software Installation

Sie wissen nun, was Sie alles für Ihre SMS Anwendung benötigen. Auch haben Sie bereits Ihre erste Kurzmitteilung gesendet und empfangen - allerdings manuell.

Es ist also nun der Zeitpunkt gekommen, wo Sie die ganze Sache automatisieren wollen.

Dazu müssen Sie Software installieren, konfigurieren und anwenden. Dieses Kapitel zeigt Ihnen die Installation der Programme, die ich Ihnen zu Beginn des Buches vorgestellt habe.

Ich habe für jedes Betriebssystem ein eigenes Unterkapitel geschrieben.

Nach dem Lesen dieses Kapitels werden Sie imstande sein, einen Webserver mit dynamischen Inhalten (PHP) und Datenbank-Anbindung (MySQL) zu installieren. Außerdem werden Sie die SMS Server Tools installieren können.

Die Installationsanleitungen leiten Sie Schritt für Schritt an, indem Sie Ihnen jeden Befehl zeigen, den Sie eingeben müssen.

Zum Abschluss testen Sie den Webserver und die Datenbank anhand einer kleinen selbstgeschriebenen dynamischen Webseite.

4.2.1. Software Installation unter Solaris

Diese Schritt-für-Schritt Installationsanleitung geht davon aus, dass Sie die Software auf einem leeren bzw. neuen SUN Rechner mit Solaris 6, 7, 8,9 oder 10 installieren.

Gehen Sie dieses gesamte Kapitel der Reihe nach durch!

4.2.1.1. Solaris konfigurieren

Installieren Sie zuerst Solaris von der Solaris CD-Rom, und zwar vollständig oder wenigstens mit Entwicklungswerkzeugen (Developer) und Web-Browser (z.B. Netscape).

Loggen Sie sich als root ein und starten Sie ein Terminal Fenster.

Diese Konfiguration gilt für alle Server ihrer SMS Anwendung.

Falls mysql oder apache bereits installiert ist, entfernen Sie diese wieder, um Versionskonflikte zu vermeiden. Sie werde gleich eigene Versionen davon installieren.

Mit den Befehlen

pkginfo | grep apache

und

pkginfo | grep mysql

können Sie das kontrollieren. Zum Entfernen geben Sie bitte den Befehl

pkgrm Paketname

ein.

Öffnen Sie die Datei /etc/profile in einem Editor und fügen Sie folgende Zeilen an das Dateiende an:

```
CC=gcc
PATH=/usr/local/bin:/usr/local/sbin:/usr/ccs/bin:/usr/local/mysql/bin:$PATH
MANPATH=/usr/local/man:$MANPATH
LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
export CC PATH MANPATH LD_LIBRARY_PATH
```

Loggen Sie sich aus und wieder ein, um die Änderungen wirksam werden zu lassen.

4.2.1.2. Packages Installieren

Downloaden Sie die folgenden Pakete passend zu Ihrer Solaris Version von http://www.sunfreeware.com.

- gcc , der GNU C und C++ Compiler
- make , Zubehör für gcc
- autoconf , Zubehör für gcc
- automake , Zubehör für gcc
- tar , Packer/Entpacker
- gzip , Packer/Entpacker
- sed , Script Interpreter für Texte
- fetchmail , Mails abrufen
- procmail , Mails verarbeiten
- mysql , Datenbank Server
- Libraries für diese Programme:
  - db
  - expat
  - gdbm
  - libxml2
  - ssl

Anleitung zur Installation finden Sie auf dieser Webseite unter dem Link Downloading/Installing. Zusammengefasst:

Downloaden Sie die obigen Pakete alle in ein leeres Verzeichnis. Bis auf gzip selbst sind alle Pakete mit gzip komprimiert und müssen vor der Installation dekomprimiert werden:

pkgadd -d gzip-1.3.5.10
gunzip *.gz
pkgadd -d bzip-2.1.0.4
pkgadd -d sed-4.1.5

...

Falls Sie die Dateien von der Buch-DVD verwenden, geht es etwas anders:

Mounten Sie die DVD und gehen Sie in das Verzeichnis für Ihre Solaris Version. Geben Sie dann ein:

pkgadd -d .

4.2.1.3. MySQL konfigurieren

Erstellen Sie ein Start/Stop-Script mit dem Namen /etc/init.d/mysqld, mit folgendem Inhalt:

```
#! /bin/sh
case "$1" in
   start)
      /usr/local/mysql/bin/mysqld_safe &
      ;;
   stop)
      pkill libexec/mysqld
      ;;
   restart|reload)
      $0 stop && $0 start
      ;;
   *)
      echo "Usage: $0 {start|stop|restart}"
   exit 1
esac
```

Geben Sie diese Befehle ein:

cd /etc/init.d
chmod a+rx mysqld
cd /etc/rc3.d
ln -s /etc/init.d/mysqld S80mysqld
groupadd mysql
useradd -d /usr/local/mysql -g mysql mysql

Installieren Sie eine leere Datenbank. Ohne Datenbank können Sie den MySQL Server später nicht starten:

/usr/local/mysql/bin/mysql_install_db
chown -R mysql /usr/ocal/mysql/var
chgrp -R mysql /usr/ocal/mysql/var

Starten Sie dem MySQL Server, beim nächsten Booten wird er automatisch gestartet:

/etc/init.d/mysqld start

MySQL erlaubt standardmässig allen Usern vollen Zugriff auf alle Datenbanken. Dies sollte man sofort ändern. Ich schlage vor, für root@localhost ein Passwort zu vergeben und alle anonymen Zugänge zu löschen. Später kann man bei Bedarf neue User hinzufügen.

mysqladmin -u root password 'Mein_Passwort'
mysql -u root -p
mysql> **use mysql;**
mysql> **delete from user where user="" or password="";**
mysql> **quit**
mysqladmin -u root -p reload

Der Username root hat nichts mit dem Solaris User root zu tun. MySQL hat sein eigene unabhängige Passwortverwaltung. Bei MySQL ist root der standardmässige Name für den Datenbank-Administrator.

Nun kann man die Datenbank nur noch vom lokalen Rechner mit root-Passwort benutzen. Wie Sie weitere User anlegen, erfahren Sie in einem späteren Kapitel.

4.2.1.4. Apache and PHP compilieren

Installieren Sie keine binary Pakete von Apache oder PHP von irgendwelchen Webseiten. Sie funktionieren nicht korrekt zusammen. Das ist auch bei SunFreeware so. Sie müssen die original Quelltexte downloaden und compilieren, damit es funktioniert.

Besuchen Sie dazu die Webseiten http://httpd.apache.org/ und www.php.net.

Packen Sie die Quelltexte von Apache aus und compilieren Sie diese:

/usr/local/bin/tar -xvzf httpd-2.2.2.tar.gz

cd httpd-2.2.2

**./configure --prefix=/usr/local/apache2 --enable-mods-shared=all --enable-ssl=shared --enabe-ssl **
--with-ssl=/usr/local/ssl --enable-proxy --enable-proxy-connect --enable-proxy-ftp --enable-proxy-http

make clean

make

make install

Packen Sie die Quelltexte von PHP aus und compilieren Sie diese:

/usr/local/bin/tar -xvzf php-5.1.4.tar.gz

cd php-5.1.4

**./configure --prefix=/usr/local/php --with-apxs2=/usr/local/apache2/bin/apxs --with-mysql=/usr/local/mysql **
--with-libxml-dir=/usr/local --with-openssl=/usr/local/ssl

make clean

make

make install

4.2.1.5. Apache und PHP konfigurieren

Kopieren Sie die Beispiel-Konfigurationsdatei von PHP:

cp /usr/local/src/php-5.1.4/php-php.ini-recommended /usr/local/lib/php.ini

Öffnen Sie die php.ini Datei in einem Editor. Schauen Sie sich die Kommentare an. Ich empfehle die folgenden Änderungen:

```
display_errors=on
doc_root=/usr/local/apache/htdocs
```

Wenn Sie sicher sind, daß IhrePHP scripte richtig funktionieren, sollten Sie die Fehlermeldungen (display_errors) wieder deaktivieren.

Öffnen Sie die Datei /usr/local/apache/2conf/httpd.conf und fügen Sie diese Zeilen dort ein, wo die anderen LoadModule Befehle stehen:

```
LoadModule php5_module modules/libphp5.so
AddType application/x-httpd-php .php
```

Achten Sie darauf, daß die Zeilen nicht doppelt vorhanden sind.

Geben Sie folgende Befehle ein:

cd /etc/init.d
ln -s /usr/local/apache/bin/apachectl apache
cd /etc/rc3.d
ln -s /etc/init.d/apache S81apache

Starten Sie den Apache Webserver manuell. Beim nächsten Booten wird er automatisch gestartet.

/etc/init.d/apache start

4.2.1.6. Apache, PHP und MySQL Testen

Testen Sie Ihren Webserver nun bitte.

Starten Sie einen WebBrowser auf dem Webserver und geben Sie in die URL Zeile ein:

http://localhost

Starten Sie einen WebBrowser auf einem anderen Computer und geben Sie in die URL Zeile ein:

http://ihr_webserver_name

Sie sollten in beiden Fällen eine Testseite von Apache zu sehen bekommen. Nun sollten Sie die PHP Funktion und die Anbindung an die MySQL Datenbank testen:

Erstellen Sie auf dem Webserver die Datei /usr/local/apache2/htdocs/test.php, mit folgendem Inhalt:

```
<html><body>
   Das ist HTML Text.<br>

   <?php
      print("Das ist PHP Text.<br>");
      print("Es folgt eine MySQL Abfrage:<br>");
      mysql_connect("localhost","root","Mein_Passwort");
      mysql_select_db("mysql");
      $ergebnis=mysql_query("select user,host,password from user");
      for ($nr = 1; $reihe = mysql_fetch_row($ergebnis); $nr++)
      {
         print("$reihe[0] $reihe[1] $reihe[2]<br>");
      }
   ?>

</body></html>
```

Verwenden Sie beim mysql_connect Befehl den Namen Ihres MySQL Servers und das richtige Passwort. Wenn der Webserver zugleich MySQL Server ist, dann ist localhost als Name einzugeben.

Stellen Sie sicher, daß diese Datei ausreichende Lese-Rechte hat:

chmod a+r /usr/local/apache2/htdocs/test.php

Starten Sie nun einen Web-Browser und geben Sie

http://localhost/test.php

oder

http://ihr_webserver_name/test.php

ein. Es sollte folgendes erscheinen:

```
Das ist HTML Text.
Das ist PHP Text.
Es folgt eine MySQL Abfrage:
root localhost x4fhg425fdfds
```

Die erste Zeile ist ganz normaler HTML Code, die zweite testet die PHP Integration in den Webserver. Die dritte und vierte Zeile testet die Verbindung zur MySQL Datenbank.

Schauen Sie sich noch einmal das Beispielprogramm an. Sie sehen hier deutlich, wie ein Stück Programmcode mitten in die HTML Datei eingefügt wurde. Anfang und Ende werden durch <?php und ?> gekennzeichnet.

Das PHP Programm verbindet sich mit der SQL Datenbank und fragt die Liste der SQL-User mit deren Passwörter ab. Die Passwörter sind verschlüsselt gespeichert, daher sehe Sie ein kryptisches Passwort hinter dem Wort root im Web-Browser.

4.2.1.7. Sourcen installieren

Compilieren Sie die Quelltexte, um ausführbare Programme zu erhalten. Die mm library wird von den SMS Server Tools benötigt, sie kann auf der Webseite http://www.ossp.org/pkg/lib/mm/ downgeloaded werden.

/usr/local/bin/tar -xvzf libmm-1.4.0.tar.gz
cd mm-1.4.0
./configure
make
make install

Ssmtp ist ein kleines Programm zum Senden von eMails durch externe Mailserver, sie bekommen es unter http://www.ibiblio.org/pub/Linux/system/mail/mta.

/usr/local/bin/tar -xvzf ssmtp-2.4.8.tar.gz
cd ssmtp-2.4.8
make
make install

Die SMS Server Tools senden und empfangen SMS durch GSM Modems. Sie bekommen die SMS Server Tools hier:

Version 2.x: http://www.meinemullemaus.de/smstools/
Version 3.x: http://smstools3.kekekasvi.com/

/usr/local/bin/tar -xvzf smstools-2.2.4.tar.gz
cd smstools

Öffnen Sie die Datei mstools/src/Makefile mit einem Editor. Aktivieren sie die Statistik Funktion durch Einfügen einer Raute vor die Zeile mit der NOSTATS Option. Entfernen Sie das Raute Zeichen vor der Option für Solaris.

make
make install
cd /etc/rc3.d
ln -s /etc/init.d/sms s82sms

Wenn Sie später die SMS Server Tools beim Booten automatisch laden wollen, benennen Sie die Datei s82sms einfach in S82sms (großbuchstabe S) um. Warten Sie damit aber bitte, bis Sie das Programm konfiguriert und getestet haben.

4.2.1.8. Merkliste für Solaris

Merken Sie sich bitte die folgenden Dateien, da sie nicht in jedem Betriebssystem identisch sind. In späteren Kapiteln werde ich nicht mehr auf Betriebssystem spezifische Details eingehen.

SMS Server Tools

starten	/etc/init.d/sms start
stoppen	/etc/init.d/sms stop
Konfigurationsdatei	/etc/smsd.conf
Fehler-Protokoll-Datei	/var/log/smsd.log
SMS-Warteschlangen	/var/spool/sms

Apache Webserver

starten	/etc/init.d/apache start
stoppen	/etc/init.d/apache stop
Konfigurationsdatei	/usr/local/apache2/conf/httpd.conf
Fehler-Protokoll-Datei	/usr/local/apache2/logs/error.log
Webseiten	/usr/local/apache2/htdocs

MySQL Datenbank

starten	/etc/init.d/mysqld start
stoppen	/etc/init.d/mysqld stop
Client	mysql -u root -p

Crontab

ändern	EDITOR=dtpad; export EDITOR; crontab -e

4.2.2. Softwareinstallation unter Windows

Diese Schritt-für-Schritt Installationsanleitung geht davon aus, dass Sie die Software auf einem leeren bzw. neuen PC mit einer beliebigen Windows Version installieren.

Gehen Sie dieses gesamte Kapitel der Reihe nach durch!

4.2.2.1. Windows Programme Installieren

Installieren Sie zuerst Windows von der Windows Installations-CD-Rom.

Installieren Sie nun die folgenden Programme:

Editpad Lite

http://www.editpadpro.com/editpadlite.html

Kopieren Sie die Datei editpad.exe in das Windows Verzeichnis (z.B. c:\windows). Starten Sie das Programm. Im Menü Optionen/Einstellungen/Verknüpfungen können Sie das Programm als Standard-Editor registrieren und es in das Senden-An Menü der rechten Maustaste integrieren.

nnCron Lite

http://www.nncron.ru/download.shtml

Installieren Sie dieses Programm, wenn Sie zeitgesteuert Programme starten möchten. Unter Windows XP können Sie dies wahlweise auch mit dem Zeitplaner in der Systemsteuerung erledigen.

Die Konfigurationsdatei ist c:\programme\cron\cron.tab. Im Unterverzeichnis log finden Sie Protokolldateien der Cron-Jobs und ggf. Fehlermeldungen.

War FTP Daemon

http://www.warftp.org/

Installieren Sie den War FTP Daemon, wenn Sie einzelne Verzeichnisse des Servers im Internet per FTP freigeben wollen. Das Programm ist sehr flexibel und dennoch einfach zu konfigurieren, daher werde ich in diesem Buch nicht weiter auf das Programm eingehen.

PDU Spy

http://www.nobbi.com/download.htm#pduspy

Das Programm ist nützlich, wenn Sie Probleme mit dem GSM Modem analysieren möchten. Normalerweise brauchen Sie dieses Programm nicht.

MySQL

http://dev.mysql.com/downloads/

MySQL ist ein Datenbank Server. Installieren Sie MySQL, wenn Sie dynamische Webseiten programmieren wollen oder wenn Sie Daten speichern wollen. Das Setup-Programm richtet für die ersten Zugriffe auch einen root Account an.

PHP und Apache

http://www.php.net/downloads.php

http://httpd.apache.org/

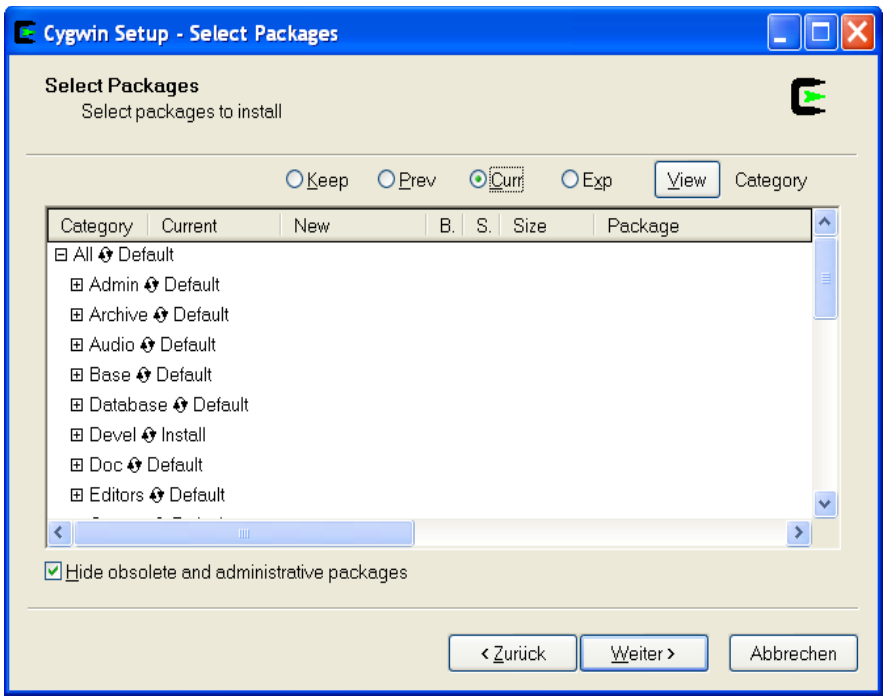
PHP ist eine Programmiersprache für Dynamische Webseiten. Apache ist ein Webserver. Installieren Sie diese beiden Programme, wenn Sie Webseiten anbieten wollen.

4.2.2.2. CygWin Installieren

CygWin benötigen Sie auf allen Servern, auf denen Sie die Beispielprogramme dieses Buches anwenden möchten. Downloaden Sie die setup.exe von http://www.cygwin.com

Starten Sie das Setup Programm und wählen Sie die Option "Install from Internet". Wählen Sie als Zielverzeichnis c:\cygwin.

Lassen Sie alle anderen Optionen unverändert, bis Sie zur Programmauswahl gelangen:



Hier sind zahlreiche Programmpakete in Kategorien zusammengefasst. Sie können für jedes Programm durch Anklicken zwischen folgenden Optionen wählen:

- Skip (Nicht installieren)
- Install (Versions-Nr. erscheint)
- Remove (entfernen)
- Re-Install (erneut installieren)

Wählen Sie keine Programmpakete ab, die bereits vorgegeben sind.

Wählen Sie zusätzlich zur Vorgabe die folgenden Programmpakete:

- Devel
  - gcc
  - make
- Editors
  - vim
- Mail
  - fetchmail
  - procmail
  - ssmtp
- Net
  - inetutils

- ncftp

- System
  - procps

4.2.2.3. Apache und PHP Installieren

Den Webserver brauchen Sie nur auf dem Server installieren, der als Webserver dienen soll. Vieleicht möchten Sie auch gar keinen Webserver betreiben.

Installieren Sie nicht das Paket apache oder apache2 aus CygWin, da diese kein PHP unterstützen!

Installieren Sie den Apache Webserver mit seinem Setup Programm. Die vorgegebenen Einstellungen sind Ok. Packen Sie dann PHP aus, und zwar in das Verzeichnis c:\php. Ein richtiges Setup Programm gibt es bei PHP nicht.

Fügen Sie c:\php in die PATH Variable ein: Bei Windows 95 und 98 schreiben Sie an das Ende der Datei c:\autoexec.bat diese Zeile:

PATH=%PATH%;c:\php

Bei Windows XP und 2000 klicken Sie in der Systemsteuerung auf „System“ und dann auf „Erweitert“. Unten befindet sich der Button „Umgebungsvariablen“, auf den Sie bitte klicken. Suchen Sie die Variable PATH und hängen Sie **;c:\php** daran an. Wie man die PATH Variable bei Windows NT ändert, weiß ich leider nicht.

Starten Sie den Computer jetzt neu, um die neue Path Variable wirksam zu machen.

Kopieren Sie die Datei **c:\php\php.ini-recommended** nach **c:\windows\php.ini**. Öffnen Sie die Datei in einem Editor. Ändern Sie vier Zeilen, daß sie so aussehen:

```
display_errors=on
doc_root=C:\Programme\Apache Software Foundation\Apache2.2\htdocs
extension_dir = "c:/php/ext"
extension=php_mysql.dll
```

Speichern Sie die Änderungen und öffnen Sie dann die Datei C:\Programme\Apache Software Foundation\Apache2.2\conf\httpd.conf. Im Startmenü finden Sie dafür unter Apache einen Menüpunkt. Fügen Sie an das Ende des LoadModule Abschnittes eine Zeile ein:

```
LoadFile "C:/php/php5ts.dll"
LoadModule php5_module "C:/php/php5apache2_2.dll"
AddType application/x-httpd-php .php
```

Jetzt beenden Sie den Apache Monitor (in der Startleiste) und starten Sie ihn erneut. Stoppen und Starten Sie ebenfalls den Apache Webserver.

4.2.2.4. Apache, PHP und MySQL Testen

Testen Sie Ihren Webserver nun bitte.

Starten Sie einen WebBrowser auf dem Webserver und geben Sie in die URL Zeile ein:

http://localhost

Starten Sie einen WebBrowser auf einem anderen Computer und geben Sie in die URL Zeile ein:

http://ihr_webserver_name

Sie sollten in beiden Fällen eine Testseite von Apache zu sehen bekommen. Nun sollten Sie die PHP Funktion und die Anbindung an die MySQL Datenbank testen:

Erstellen Sie auf dem Webserver die Datei c:\programme\Apache Software Foundation\Apache2.2\htdocs\test.php, mit folgendem Inhalt:

```
<html><body>
   Das ist HTML Text.<br>
   <?php
      print("Das ist PHP Text.<br>");
      print("Es folgt eine MySQL Abfrage:<br>");
      mysql_connect("localhost","root","Mein_Passwort");
      mysql_select_db("mysql");
      $ergebnis=mysql_query("select user,host,password from user");
      for ($nr = 1; $reihe = mysql_fetch_row($ergebnis); $nr++)
      {
         print("$reihe[0] $reihe[1] $reihe[2]<br>");
      }
   ?>
</body></html>
```

Verwenden Sie beim mysql_connect Befehl den Namen Ihres MySQL Servers und das richtige Passwort. Wenn der Webserver zugleich MySQL Server ist, dann ist localhost als Name einzugeben.

Starten Sie nun einen Web-Browser und geben Sie

http://localhost/test.php

oder

http://ihr_webserver_name/test.php

ein. Es sollte folgendes erscheinen:

```
Das ist HTML Text
Das ist PHP Text
Es folgt eine MySQL Abfrage
root localhost x4fhg425fdfds
```

Die erste Zeile ist ganz normaler HTML Code, die zweite testet die PHP Integration in den Webserver. Die dritte und vierte Zeile testet die Verbindung zur MySQL Datenbank.

Schauen Sie sich noch einmal das Beispielprogramm an. Sie sehen hier deutlich, wie ein Stück Programmcode mitten in die HTML Datei eingefügt wurde. Anfang und Ende werden durch <?php und ?> gekennzeichnet.

Das PHP Programm verbindet sich mit der SQL Datenbank und fragt die Liste der SQL-User mit deren Passwörter ab. Die Passwörter sind verschlüsselt gespeichert, daher sehe Sie ein kryptisches Passwort hinter dem Wort root im Web-Browser.

4.2.2.5. SMS Server Tools Installieren

Die SMS Server Tools brauchen Sie nur auf dem Server installieren, an dem auch die GSM Modems angeschlossen sind.

Downloaden Sie die libmm von http://www.ossp.org/pkg/lib/mm/ und die SMS Server Tools von:

Version 2.x: http://www.meinemullemaus.de/smstools/
Version 3.x: http://smstools3.kekekasvi.com/

Kopieren Sie die beiden Dateien libmm-1.4.0.tar.gz und smstools-x.x.x.tar.gz in das Verzeichnis c:\cygwin\usr\src. Starten Sie dann ein CygWin Bash Shell Fenter und geben Sie ein:

mkdir /etc/init.d

mkdir /usr/local/src
cd /usr/local/src
tar -xvzf libmm-1.4.0.tar.gz
cd /usr/local/src/mm-1.4.0
./configure
make
make install

cd /usr/local/src
tar -xvzf smstools*.tar.gz
cd /usr/local/src/smstools

Öffnen Sie die Datei c:\cygwin\usr\local\src\smstools\src\Makefile mit einem Editor. Aktivieren sie die Statistik Funktion durch Einfügen einer Raute vor die Zeile mit der NOSTATS Option.

mkdir /var/spool
make
make install

4.2.2.6. Merkliste für Windows

Merken Sie sich bitte die folgenden Dateien, da sie nicht in jedem Betriebssystem identisch sind. In späteren Kapiteln werde ich nicht mehr auf Betriebssystem spezifische Details eingehen.

SMS Server Tools

starten	smsd &
stoppen	pkill smsd
Konfigurationsdatei	/etc/smsd.conf
Fehler-Logdatei	/var/log/smsd.log
SMS-Warteschlangen	/var/spool/sms

Apache Webserver

starten	net start Apache (in DOS Eingabeaufforderung, oder mit dem Monitor Programm)
stoppen	net stop Apache (in DOS Eingabeaufforderung, oder mit dem Monitor Programm)
Konfigurationsdatei	c:\programme\Apache Software Foundation\Apache2.2\conf\httpd.conf
Webseiten	c:\programme\Apache Software Foundation\Apache2.2\htdocs

MySQL Datenbank

starten	net start mysql (in DOS Eingabeaufforderung)
stoppen	net stop mysql (in DOS Eingabeaufforderung)
Client	C:\Programme\MySQL\MySQL Server 4.1\bin\mysqlc -u root -p (in DOS Eingabeaufforderung)

Crontab

ändern	c:\programme\cron\cron.tab
Logdateien	c:\programme\cron\log\

4.2.3. Softwareinstallation unter Linux

Alle Linux Distributionen enthalten sämtliche Software, die ich in diesem Buch nutze. Sogar die SMS Server Tools sind bereits in vielen Linux Distributionen (z.B. Debian) enthalten.

Wenn Sie jedoch die aktuellsten SMS Server Tools benutzen möchten, benötigen Sie zusätzlich zur Standardinstallation die folgenden Pakete:

- gcc
- make
- libmm

Je nach Anwendung benötigen Sie außerdem die folgenden Pakete:

- procmail
- sendmail (oder einen anderen Mailserver)
- fetchmail
- apache
- php
- mysql
- ncftp

Normalerweise sollte das Installationsprogramm von Linux die Programme automatisch konfigurieren. Doch falls dies nicht wie erwartet klappt, helfen eventuell die Hinweise in den folgenden Kapiteln weiter.

4.2.3.1. Vorbereitung

Loggen Sie sich als root User ein, um Software installieren zu können.

Geben Sie den Befehl

grep :initdefault /etc/inittab

ein und notieren Sie sich die Zahl zwischen den beiden Doppelpunkten (meistens eine 3).

Dies ist der Runlevel, der beim Booten aktiviert wird. Sie werden später einige Links in genau diesem Runlevel-Verzeichnis anlegen müssen, damit die Programme (mysql, apache, smstools) automatisch beim booten gestartet werden.

In den folgenden Kapiteln werde ich immer davon ausgehen, dass es sich um Runlevel 3 handelt. Sie müssen die Zahl in diesem Buch ggf. entsprechend Ihrem System anpassen, damit es funktioniert.

4.2.3.2. SMS Server Tools Installieren

Die SMS Server Tools (und OssP mm Shared Memory Library) brauchen Sie nur auf den Servern installieren, wo die Modems angeschlossen werden.

Installieren Sie die OssP mm Shared Memory Library vorzugsweise von Ihren Linux Installations-CD's. Falls die Library nicht dabei ist, dann installieren Sie stattdessen die Sourcen.

Downloaden Sie die libmm von http://www.ossp.org/pkg/lib/mm/ und die SMS Server Tools von:

Version 2.x: http://www.meinemullemaus.de/smstools/
Version 3.x: http://smstools3.kekekasvi.com/

Kopieren Sie die beiden Dateien libmm-1.4.0.tar.gz und smstools-x.x.x.tar.gz in das Verzeichnis /usr/local/src.

cd /usr/local/src
tar -xvzf libmm*tar.gz
cd mm*
./configure
make
make install

Öffnen Sie die Datei /etc/ld.so.conf und fügen sie das Verzeichnis /usr/local/lib ein, falls nicht vorhanden. Geben Sie danach den Befehl **ldconfig** ein.

Manche Linux Versionen haben keinen ldconfig Befehl, weil sie ihn nicht brauchen.

Installieren Sie die SMS Server Tools:

cd /usr/local/src
tar -xvzf smstools*.tar.gz
cd /usr/local/src/smstools

Öffnen Sie die Datei /usr/ocal/src/smstools/src/Makefile mit einem Editor. Aktivieren sie die Statistik Funktion durch Einfügen einer Raute vor die Zeile mit der NOSTATS Option.

make
make install

cd /etc/rc3.d oder **cd /etc/init.d/rc3.d**

bitte die 3 ggf. ändern, wie im vorherigen Kapitel erwähnt

ln -s /etc/init.d/sms s82sms

Wenn Sie später die SMS Server Tools beim Booten automatisch laden wollen, benennen Sie die Datei s82sms einfach in S82sms um.

4.2.3.3. MySQL Konfigurieren

MySQL erlaubt standardmässig allen Usern Vollzugriff auf alle Datenbanken. Dies sollte man sofort ändern. Ich schlage vor, für root@localhost ein Passwort zu vergeben und alle anonymen Zugänge zu löschen. Später kann man bei Bedarf neue User hinzufuegen.

mysqladmin -u root password 'Mein_Passwort'
mysql -u root -p
mysql> **use mysql;**
mysql> **delete from user where user="" or password="";**
mysql> **quit**
mysqladmin -u root -p reload

Der Username root hat nichts mit dem Linux User root zu tun. MySQL hat sein eigene unabhängige Passwortverwaltung. Bei MySQL ist root der standardmässige Name für den Datenbank-Administrator.

Nun kann man die Datenbank nur noch vom lokalen Rechner mit root-Passwort benutzen. Wie Sie weitere User anlegen, erfahren Sie in einem späteren Kapitel.

4.2.3.4. Apache, PHP und MySQL Testen

Testen Sie Ihren Webserver nun bitte.

Starten Sie einen WebBrowser auf dem Webserver und geben Sie in die URL Zeile ein:

http://localhost

Starten Sie einen WebBrowser auf einem anderen Computer und geben Sie in die URL zeile ein:

http://ihr_webserver_name

Sie sollten in beiden Fällen eine Testseite von Apache zu sehen bekommen. Nun sollten Sie die PHP Funktion und die Anbindung an die MySQL Datenbank testen:

Finden Sie heraus, wo das htdocs Verzeichnis liegt.

find / -name htdocs

Erstellen Sie in diesem Verzeichnis die Datei test.php , mit folgendem Inhalt:

```
<html><body>
   Das ist HTML Text.<br>
   <?php
      print("Das ist PHP Text.<br>");
      print("Es folgt eine MySQL Abfrage:<br>");
      mysql_connect("localhost","root","Mein_Passwort");
      mysql_select_db("mysql");
      $ergebnis=mysql_query("select user,host,password from user");
      for ($nr = 1; $reihe = mysql_fetch_row($ergebnis); $nr++)
      {
         print("$reihe[0] $reihe[1] $reihe[2]<br>");
      }
   ?>
</body></html>
```

Verwenden Sie beim mysql_connect Befehl den Namen Ihres MySQL Servers und das richtige Passwort. Wenn der Webserver zugleich MySQL Server ist, dann ist localhost als Name einzugeben.

Stellen Sie sicher, daß diese Datei ausreichende Lese-Rechte hat:

chmod a+r /usr/local/apache2/htdocs/test.php

Starten Sie nun einen Web-Browser und geben Sie

http://localhost/test.php

oder

http://ihr_webserver_name/test.php

ein. Es sollte folgendes erscheinen:

```
Das ist HTML Text
Das ist PHP Text
Es folgt eine MySQL Abfrage
root localhost x4fhg425fdfds
```

Die erste Zeile ist ganz normaler HTML Code, die zweite testet die PHP integration in den Webserver. Die dritte und vierte Zeile testet die Verbindung zur MySQL Datenbank.

Schauen Sie sich noch einmal das Beispielprogramm an. Sie sehen hier deutlich, wie ein Stück Programmcode mitten in die HTML Datei eingefügt wurde. Anfang und Ende werden durch <?php und ?> gekennzeichnet.

Das PHP Programm verbindet sich mit der SQL Datenbank und fragt die Liste der SQL-User mit deren Passwörter ab. Die Passwörter sind verschlüsselt gespeichert, daher sehe Sie ein kryptisches Passwort hinter dem Wort root im Web-Browser.

4.2.3.5. Merkliste für Linux

Merken Sie sich bitte die folgenden Dateien, da sie nicht in jedem Betriebssystem identisch sind. In späteren Kapiteln werde ich nicht mehr auf Betriebssystem spezifische Details eingehen.

SMS Server Tools

starten	/etc/init.d/sms start
stoppen	/etc/init.d/sms stop
Konfigurationsdatei	/etc/smsd.conf
Fehler-Logdatei	/var/log/smsd.log
SMS-Warteschlangen	/var/spool/sms

Apache Webserver

starten	/etc/init.d/apache start
stoppen	/etc/init.d/apache stop
Konfigurationsdatei	/usr/local/apache/conf/httpd.conf
Fehler-Logdatei	/usr/local/apache/logs/error.log
Webseiten	/usr/local/apache/htdocs

Die Position dieser Dateien kann bei manchen Linux Distributionen abweichen.

MySQL Datenbank

starten	/etc/init.d/mysqld start
stoppen	/etc/init.d/mysqld stop
Client	mysql -u root -p

Crontab

ändern	EDITOR=kedit; export EDITOR; crontab -e

Wenn kedit nicht installiert ist, möchten Sie vielleicht xedit oder vi stattdessen benutzen.

4.3. CygWin Schnelleinstieg

Die meisten Windows Administratoren sind mit der Unix Shell nicht vertraut.

Daher stelle ich Ihnen in diesem Kapitel im Schnelldurchgang alles Wissenswerte über die CygWin Bash Shell vor.

Machen Sie sich mit der Shell vertraut, denn sie werden später häufig damit arbeiten müssen. Fast alle Programme, mit denen dieses Buch arbeitet, werden durch Eingabe von Befehlen gesteuert, denn sie haben keine grafische Oberfläche, die Sie mit der Maus bedienen können.

4.3.1. Die Shell

(Die folgenden Bildschirmfotos sind aufgehellt, um Papierstau in Laserdruckern zu vermeiden und Tinte bzw. Toner zu sparen. Ich hoffe, der Kontrast ist ausreichend, um die Texte lesen zu können.)

Die Shell entspricht funktionell der MS-Dos Eingabeaufforderung. Hier geben Sie alle Unix-Befehle ein.

Wenn ich Sie in diesem Buch zur Eingabe eines Befehles auffordere, tun Sie dies immer in einer CygWin Shell, denn in der MS-Dos Eingabeaufforderung können Sie auch nur Dos (und Windows) Befehle eingeben.

Sie können wie gewohnt die Cursortasten benutzen, um die zuletzt eingegebenen Befehle nochmal zu wiederholen.

Mit der Tabulator-Taste können Sie unvollständige Dateinamen automatisch ergänzen lassen. Probieren Sie es aus:

Der erste ls Befehl zeigt das Inhaltsverzeichnis in der einfachsten Form an. Beim zweiten Befehl wird der halbe Dateiname automatisch auf cygwin. ergänzt, sobald Sie die Tabulator-Taste drücken. Das funktioniert auch mit Verzeichnisnamen. Da es zwei Dateien gibt, die mit cygwin. beginnen, überlässt die Shell Ihnen die Wahl, welche der beiden Sie nun anzeigen wollen.

Wenn Sie eine Shell schliessen, beenden sich normalerweise auch alle Programme, die sie darin gestartet haben. Manche Programme bleiben aber im Speicher und laufen weiter! Kontrollieren Sie das im Zweifelsfall mit dem Taskmanager.

4.3.2. Dateinamen

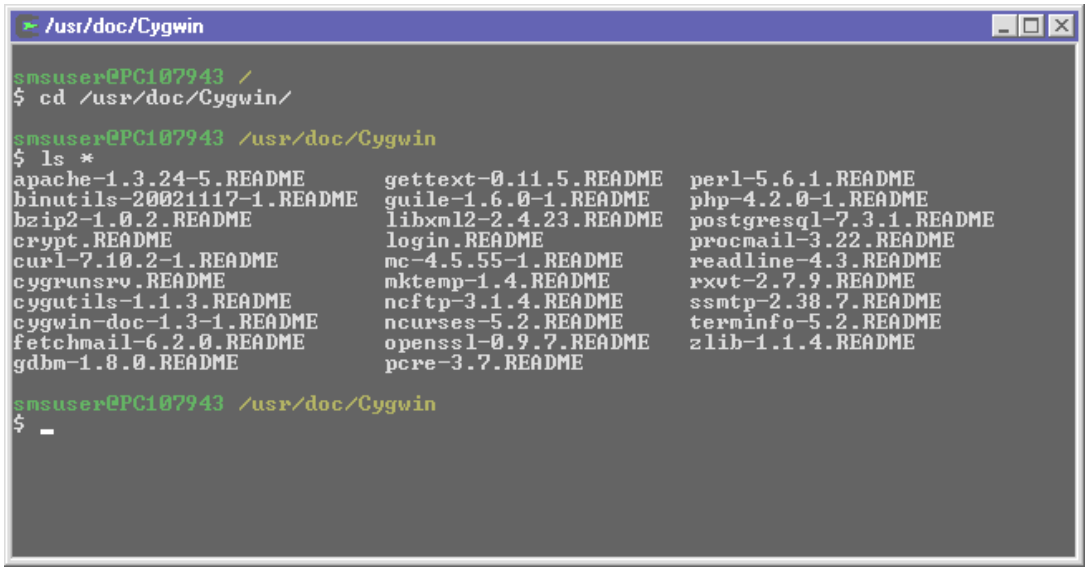
Unter Unix ist die groß/Kleinschreibung wichtig und muss immer beachtet werden.

Wenn Sie in Dateinamen Jokerzeichen verwenden, beachten Sie diese Unterschiede zur MS-Dos Eingabeaufforderung:

Ein Stern steht für beliebig viele Zeichen.

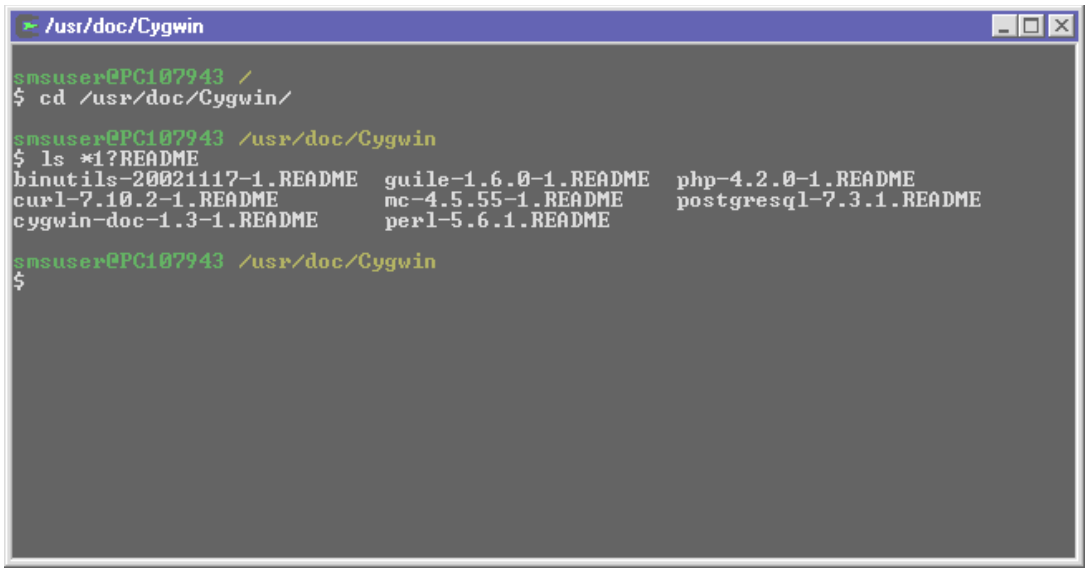
Ein Fragezeichen steht für genau ein Zeichen.

Der Punkt hat unter Unix keine besondere Bedeutung. Angenommen, es gibt mehrere README-Dateien:

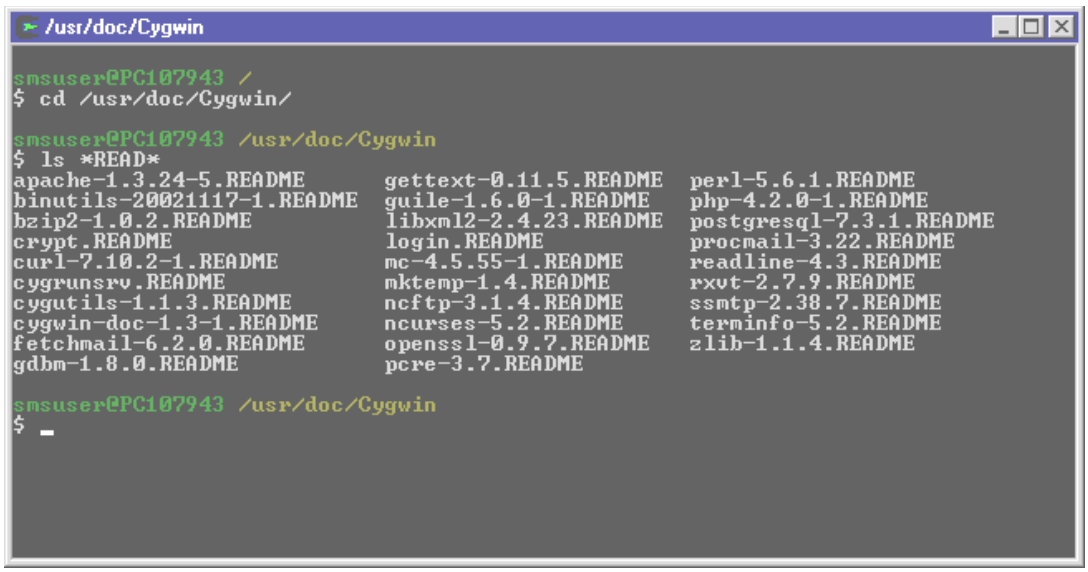


Dann zeigt der Befehl ls * diese auch alle an. In der MS-Dos Eingabeaufforderung hätten Sie dir *.* eingeben müssen, weil dort ein Stern als Jokerzeichen für alles, ausser den letzten Punkt gilt. Unter Unix ist der Stern auch ein Joker für alle Zeichen, Punkte stellen keine Ausnahme dar.

Gleiches gilt auch für das Fragezeichen:

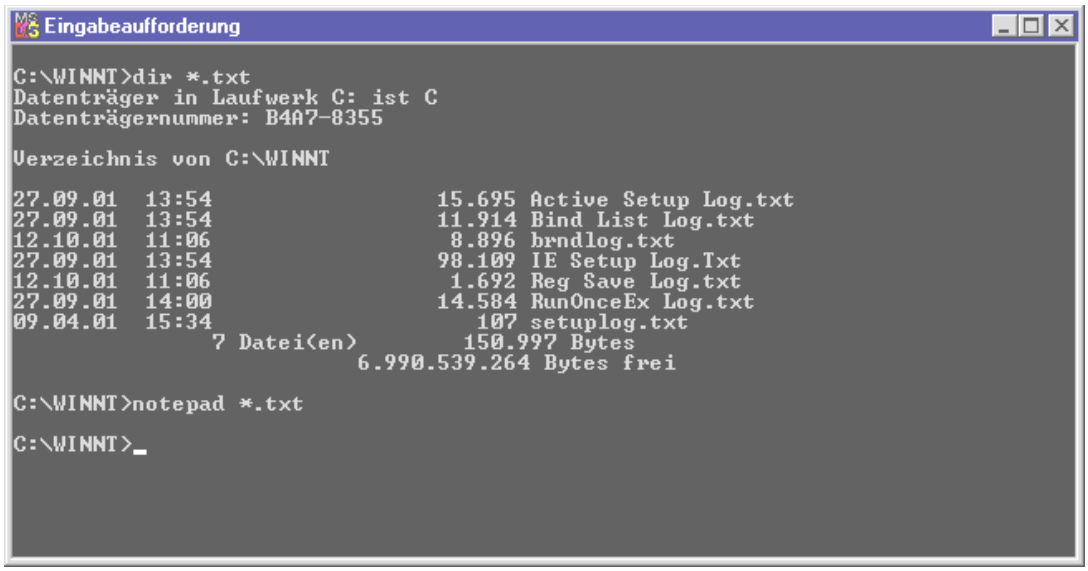


Sie können Jokerzeichen beliebig kombinieren, z.B. so:



Dieser Befehl zeigt alle Dateien an, in deren Name irgendwo "READ" vorkommt.

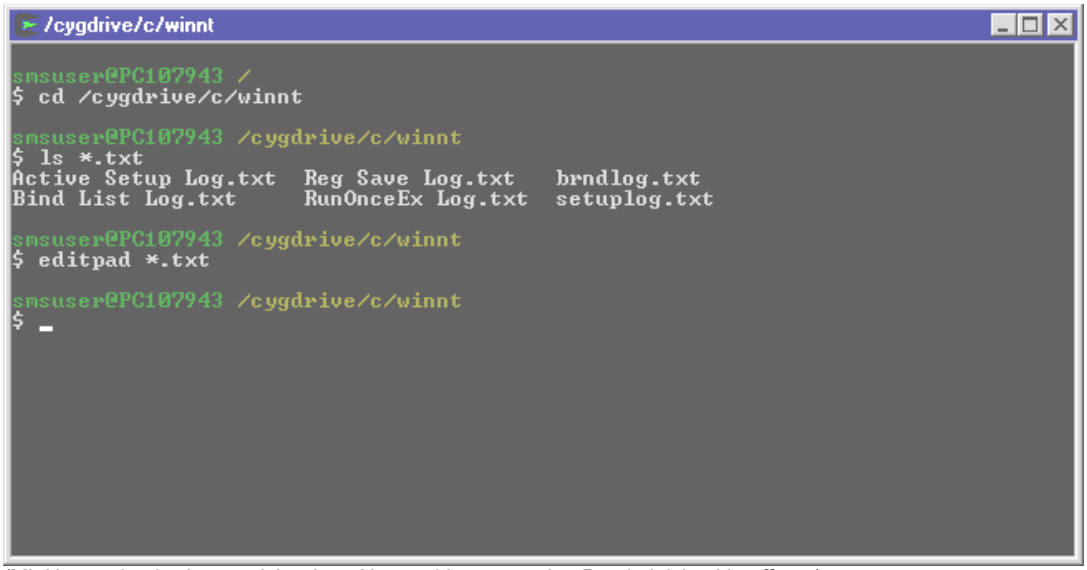
Unter MS-Dos werden die Jokerzeichen vom Befehl selbst erkannt und verarbeitet. Sie können z.B. dir *.txt eingeben, weil der dir Befehl weiss, was er mit dem Stern anfangen soll. Versuchen Sie das einmal mit Notepad:



Dann wird Notepad meckern, denn mit dem Stern kann es nichts anfangen und es versucht unerwarteter Weise die Datei *.txt zu laden - die freilich nicht existiert.

Unter Windows werden Sie fest stellen, dass sehr viele Programme Jokerzeichen korrekt erkennen, aber leider nicht alle. MS-Word 2000 (und vielleicht auch ältere Versionen) kann es.

Unter Unix werden die Jokerzeichen von der Shell erkannt und durch die "echten" Dateinamen ausgetauscht. Sie könnten dort also Editpad mit Jokerzeichen starten(editpad *.txt), weil *.txt in eine Liste mit allen passenden Dateinamen umgewandelt wird, bevor das Programm gestartet wird:



(Mit Notepad geht das so nicht, denn Notepad kann nur eine Datei gleichzeitig öffnen.)

Ausführbare Programme haben unter Unix meistens keine besondere Endung, während Windows Programme immer an der Endung .exe erkennbar sind.

4.3.3. Verzeichnisnamen

Unter Unix werden Verzeichnisnamen durch Slash getrennt, während sie unter Windows durch Backslash getrennt werden. Beispiel:

ls /usr/local/

Der Backslash dient zwei anderen Zwecken:

Verlängern Sie eine Befehlszeile, indem Sie ans Ende der unvollständigen Zeile ein Backslash schreiben. z.B.:

**ls **
"Mein Text.doc"

Dieser Befehl entspricht ls "Mein Text.doc". Bei sehr langen Befehlszeilen mag das nützlich sein.

Sparen Sie Anführungsstriche um Dateinamen mit Leerzeichen, indem Sie ein Backslash vor das Leerzeichen schreiben. z.B:

ls Mein\ Text.doc

Das Leerzeichen wird in diesem Fall nicht als Trennzeichen für mehrere Dateinamen betrachtet.

Es gibt keine Laufwerksbuchstaben.

Das Hauptverzeichnis des Systems heisst schlicht "/". In der CygWin SHell erscheint das Verzeichnis c:\cygwin als Hauptverzeichnis.

Sie wollen nun sicher wissen, wie Sie denn in der Shell an Ihr Hauptverzeichnis c:\ herankommen. Dazu müssen Sie nur wissen, dass alle Laufwerksbuchstaben unter /cygdrive wiederzufinden sind:

Windows	**CygWin Shell**
c:\cygwin	/
c:\cygwin\tmp	/tmp
c:\	/cygdrive/c
d:\	/cygdrive/d
e:\	/cygdrive/e
c:\windows	/cygdrive/c/windows

Wenn Sie ls /cygdrive/ eingeben, sehen Sie vieleicht nicht alle Laufwerksbuchstaben, die sie erwarten. Ich konnte nicht herausfinden, warum das so ist, aber sie existieren und funktionieren auf jeden Fall, auch wenn sie nicht angezeigt werden!

4.3.4. Zugriffsrechte

Unter Ms-Dos kann jeder Benutzer jede Datei öffnen. Sie können Veränderungen verbieten, indem Sie die Datei mit dem attrib Befehl schreibschützen. Doch jeder Benutzer kann den Schreibschutz mit dem selben befehl wieder entfernen.

Unter Unix funktioniert das etwas anders, weil Unix von Anfang an für mehrere Benutzer pro Rechner ausgelegt war und von Anfang an Netzwerk-Funktionen hatte. CygWin simuliert die Zugriffsrechte von Unix, damit Unix Programme ohne große Veränderung auch unter Windows laufen.

Jede Datei hat drei Gruppen von Zugriffsrechten:

- Besitzer (u)
- Gruppe (g)
- Fremde (o)

Besitzer ist derjenige, der die Datei erstellt hat.

Die Gruppe ist immer "unknown". Alle Benutzer des Windows PC sind Mitglied der Gruppe "unknown", weil es unter Windows 95 keine Benutzergruppen (in Unix Stil) gibt und CygWin ursprünglich für Windows 95 entwickelt wurde.

Fremde sind alle anderen, also Personen im Netzwerk, die der PC nicht kennt. Diese haben normalerweise keinen Zugriff auf Dateien. Daher sind die Zugriffsrechte für Fremde unter Windows uninteressant.

Für jede dieser drei Gruppen gibt es drei Zugriffsrechte:

- Lesen (r)
- Schreiben (w)
- Ausführen (x)

Lesen bedeutet, dass die Datei gelesen werden darf.

Schreiben bedeutet, dass die Datei beschrieben werden darf.

Ausführen bedeutet, dass das Programm gestartet werden darf, macht insofern nur bei Programmen Sinn.

Auch Verzeichnisse haben Zugriffsrechte. Die Bedeutung der Rechte ist hier aber etwas anders, wenngleich die selben Buchstaben benutzt werden:

- Inhalt anzeigen (r)
- Inhalt verändern (w)
- In das Verzeichnis rein gehen (x)

Das w Recht erlaubt Ihnen, den Inhalt des Inhaltverzeichnisses zu verändern. Damit ist das Erstellen neuer Dateien, sowie das Löschen und Umbenennen vorhandener Dateien gemeint, denn dadurch verändern Sie das Inhaltsverzeichnis selbst und nicht das Inhaltsverzeichnis.

Das x Recht erlaubt es Ihnen, mit dem cd Befehl in das Verzeichnis hinein zu gehen.

Mit dem Befehl **ls -l /etc** sehen Sie das Inhaltsverzeichnis von /etc in ausführlicher Form:

```
-rw-r--r-- 1 SF unknown 12546 Oct 1 05:53 termcap
```

Das erste Zeichen ist bei Verzeichnissen ein d und bei normalen Dateien ein Strich. Danach kommen drei Blöcke von rwx Rechten. Links für den Besitzer, in der Mitte für die Gruppe und rechts für Fremde.

Die obige Datei kan vom Besitzer gelesen und beschrieben werden. Die Gruppe und andere dürfen sie nur lesen.

Die Datei wurde vom Benutzer "SF" erstellt, der Mitglied in der Benutzergruppe "unknown" ist.

Die Datei ist 12546 Bytes groß, wurde am 1. Oktober zuletzt geändert und heißt termcap.

Niemand darf die Datei ausführen, denn es gibt kein "x" Recht - es ist nämlich kein ausführbares Programm.

4.3.5. Vergleich Windows/Unix Befehle

Da sie vermutlich mit DOS/Windows Befehlen vertraut sind, möchte ich Ihnen die wichtigsten Befehle anhand einer Vergleichstabelle auflisten:

Windows Befehl	Unix Befehl
dir c:\windows Inhalt des Windows-Verzeichnisses anzeigen	**ls -l /cygdrive/c/windows**
cd \temp Ins Temp-Verzeichnis gehen	**cd /tmp**
md test Ein neues Verzeichnis mit Namen test anlegen	**mkdir test**
rm test Das Verzeichnis test entfernen	**rmdir test**
del hallo Die Datei hallo löschen	**rm hallo**
deltree muell Das ganze Verzeichnis muell mitsamt Inhalt löschen	**rm -R muell**
befehl /? Hilfe zum Befehl anzeigen.	**befehl --help | more** **befehl -h | more** **befehl -h | more**
Windows Taskmanager Bricht das Programm hart ab, ohne Rücksicht auf Verluste. Dies entspricht einem mehrmaligen Klick auf "Task Beenden" im Windows Taskmanager.	**ps -ef** **kill nummer** **kill -9 nummer**
attrib -w dateiname Nimmt allen Usern (a) das Schreibrecht auf die Datei egal. Anstelle von a können Sie auch u, g oder o verwenden. +w würde das Schreibrecht hinzufügen.	**chmod a-w dateiname**
move datei ziel Verschiebt eine Datei in ein anderes Verzeichnis.	**mv datei ziel**
ren alt neu	**mv alt neu**

Benennt eine Datei um

---

type datei **cat datei**

Zeigt den Inhalt einer Textdatei an

---

type datei | more **cat datei | more**
more datei
less datei

Zeigt den Inhalt einer Textdatei seitenweise an. Less kann sogar zurück scrollen.

---

exit **exit**

Beendet eine Shell

4.3.6. Suchpfade

Als Dos/Windows User sind sie sicher mit der PATH Variable vertraut. Sie bestimmt, wo ausführbare Programme gesucht werden, fall sie nicht im aktuellen Arbeitsverzeichnis stehen.

Auch unter Unix gibt es eine solche Variable. Mit dem Befehl

echo $PATH

können Sie den Suchpfad anzeigen lassen. Beachten Sie, dass unter Unix alle Verzeichnisse durch Doppelpunkte getrennt werden, während unter Dos/Windows dafür Semikolons verwendet werden.

Um einen Pfad hinzuzufügen, geben Sie diese Befehle ein:

PATH=$PATH:/irgend/ein/verzeichnis
export PATH

Eine Eigenart hat Unix: Es sucht ausführbare Programme niemals im Arbeitsverzeichnis, es sei denn, dieses ist ebenfalls in der PATH Variable angegeben. Daher enthält die PATH-Variable meistens auch das Verzeichnis ".". Wie unter Dos/Windows ist "." ein Synonym für "das aktuelle Arbeitsverzeichnis".

4.4. Arbeiten mit Cron Jobs

Cron Jobs sind Programme, die zu bestimmten Uhrzeiten wiederholt ausgeführt werden sollen.

Unter Windows benötigen Sie dazu das Programm nnCron Lite. Unter Linux und Solaris gehört diese Funktion hingegen zum Standard.

Unter Windows müssen Sie dazu nur die Datei c:\programme\cron\cron.tab editieren.

Unter Unix geben Sie den Befehl **EDITOR="vi";export EDITOR;crontab -e** ein, um die Cron-Tabelle zu verändern. Unix startet dann denn Editor vi. Sie können stattdessen auch jeden anderen Texteditor benutzen, den Sie bevorzugen (z.B. dtpad oder kedit).

Bitte beachten Sie, dass unter Unix jeder User seine eigene Cron-Tabelle hat. Sie werden warscheinlich meistens die Cron-Tabelle von root ändern wollen, loggen Sie sich also vorher als root ein.

Egal welchen Editor sie verwenden und welches Betriebssystem Sie bevorzugen, das Format der Cron-Tabelle ist immer das Selbe. Ein Beispiel:

```
datei=/tmp/muell.txt
1,31  *   *   *   *   rm $datei
*/10  *   *   *   *   cp /tmp/logfile.alt /tmp/logfile.neu
5     1   *   *   1   cp /tmp/logfile.alt /tmp/logfile.montag
0     1-3 *   *   *   rm /tmp/unsinn.txt
# Das ist ein Kommentar
```

Die erste Zeile setzte die Umgebungsvariable datei auf /tmp/muell.txt. Alle Cronjobs können diese Variable sehen, während sie laufen. Das werden Sie sicher nur sehr selten benötigen. Sie können beliebig viele Variablen auf diese Art setzen.

Die folgenden vier Zeilen starten Programme zu bestimmten Uhrzeiten. Jede Zeile hat genau 5 Zeitangaben!

```
Minute (0-59)
Stunde (0-23)
Tag (1-31)
Monat (1-12)
Wochentag 0=Sonntag, 1=Montag, usw.
```

Der erste Cron-Job wird täglich jede Stunde zur 1. und 31. Minute ausgeführt.

Der Zweite Cron-Job wird täglich zu jeder 10. Minute ausgeführt (Solaris kann das so nicht! Geben Sie 0,10,20,30,40,50 anstatt */10 ein).

Der dritte Cron-Job wird jeden Montag um 01:05 Uhr ausgeführt.

Der vierte Cronjob wird jeden Tag um 1:00, 2:00 und 3:00 ausgeführt.

Wie sie sehen, steht das * für "jede" Minute, Stunde, Tag, Monat oder Wochentag.

***/x** bedeutet, jede xte Minute, Stunde, Tag oder Monat.

Mehrere Zahlen können durch Kommas getrennt angegeben werden. Das Komma steht für "und".

Zahlenbereiche können durch Minus-Zeichen getrennt werden und bedeuten von-bis.

Solaris versteht die Schreibweise ***/x** nicht!

Unter Windows schreibt das Programm nnCron Lite alle Bildschirmausgaben der Befehle in Logfiles.

Was jetzt kommt, gilt nur für Unix:

Unter Unix wird jeweils eine eMail an den User geschickt, in der die Ausgaben der Programme erscheinen.

Die eMails können Sie unterdrücken, indem Sie die Ausgaben der Befehle in eine Datei umleiten:

```
* * * * * command >/var/log/logfile
```

So werden "normale" Ausgaben in die Datei umgeleitet. Achtung: Das Verzeichnis /var/log ist nur für root beschreibbar! Für andere User müssen Sie die Datei bereits vorher anlegen und Schreibrechte vergeben.

Wenn Sie die Ausgabe nach /dev/null umleiten, werden diese ins Nirvana geschrieben. /dev/null verschluckt alles auf nimmer wiedersehen.

Fehlermeldungen erhält der User aber immer noch per eMail. Auch diese kann man unterdrücken, und zwar so:

```
* * * * * command >/var/log/logfile 2>&1
```

Diese Kombination sollte man möglichst nicht mit /dev/null machen, weil man sonst überhaupt nicht weiss, ob der Cronjob richtig funktioniert hat.

4.5. Arbeiten mit eMails

Viele Anwender möchten SMS Dienste mit eMail kombinieren.

Die folgenden Seiten gehen davon aus, dass Sie bereits einen funktionstüchtigen Mailserver betreiben. Ich werde Ihnen zeigen, wie Sie diesen mit ein paar wenigen Werkzeugen aus Sicht eines Mail-Clients an der Kommandozeile benutzen.

Die hier vorgestellten Befehle können Sie später innerhalb von Shell-Scripten anwenden, um den Versand und dem Empfang von eMails zu automatisieren.

4.5.1. Sendmail Konfigurieren

Dieses Kapitel betrifft nur Unix Systeme (also Linux und Solaris).

Sendmail ist das komplizierteste Programm zum Senden von eMails, denn es kann einfach alles.

Dies wäre eigentlich Grund genug, ein anderes einfacheres Programm zu benutzen, aber sendmail ist als Standardprogramm auf jedem Unix System vorhanden und es gibt nicht wirklich viele Alternativen.

Ein frisch installiertes sendmail kann nur lokale eMails zustellen. Ich möchte Ihnen in den folgenden Kapiteln zeigen, wie sie sendmail so konfigurieren, dass alle eMails über einen anderen bereits funktionierenden Mailserver versendet werden.

Wenn Sie darüber hinaus mehr über sendmail lernen möchten, lesen Sie bitte das Buch:

sendmail
O'Reilly Verlag
ISBN 1-56592-222-0

Das Buch enthält 1050 Seiten Lesestoff. Das volle Verständnis von sendmail ist nur der absoluten Elite unter den Unix Administratoren vorbehalten.

Bitte beachten Sie, dass die Konfigurationsdateien von Sendmail bei jeder Version etwas unterschiedlich sind. Die Anleitungen in diesem Kapitel können Sie also eventuell nicht 1:1 übernehmen.

4.5.1.1. Sendmail konfigurieren unter SuSE Linux

Aktuelle SuSE Linux Versionen installieren standardmässig nicht sendmail, sondern postfix (eine Alternative zu sendmail). Da ich in diesem Buch nicht noch ein weiteres Programm beschreiben möchte, deinstallieren Sie bitte postfix und installieren Sie stattdessesn sendmail.

Aktuelle SuSE Linux Versionen ermöglichen es Ihnen, sendmail mit Yast2 zu konfigurieren.

Unter älteren SuSE Linux Versionen, z.B. SuSE 7.3 müssen Sie manuell vorgehen:

Editieren Sie bitte zuerst die Datei /etc/rc.config.d/sendmail.rc.config gemäss Ihrer Netzwerkumgebung. Die Konfigurationsdatei enthält eine Menge ausführlicher Kommentare.

Geben Sie in die Zeile SENDMAIL_SMARTHOST den Hostnamen Ihres Mailservers ein. Alle eMails werden dann durch diesen Mailserver verschickt. Führen Sie danach /sbin/SuSEconfig aus.

Testen Sie den Versand von eMail mit den Befehl:

/usr/lib/sendmail -v receiver@mail.isis.de
From: sender@mail.isis.de

Hallo
<Strg-D>

Setzen Sie in dem Befehl sinnvolle eMail-Adressen ein. Die eMail sollte den Empfänger erreichen.

4.5.1.2. Sendmail konfigurieren unter RedHat Linux

Diese Anleitung bezieht sich auf RedHat 7.3. Neuere Versionen werden hoffentlich ein komfortables Konfigurationsprogramm beinhalten. Diese Anleitung ist vermutlich auch unter älteren RedHat Versionen anwendbar.

Unter RedHat Linux brauchen Sie nur eine vorbereitete Konfigurationsdatei anpassen.

Editieren Sie /etc/mail/sendmail.mc. Dort gibt es eine Zeile, die mit "dnl" auskommentiert ist. Löschen die Buchstaben "dnl" und setzen Sie den Namen Ihres Mailservers ein:

```
define(`SMART_HOST',`mail.isis.de')dnl
```

Geben Sie nun die folgenden Befehle ein, um diese Konfigurations-Vorlage in eine sendmail.cf zu übersetzen:

cd /etc/mail
m4 sendmail.mc > sendmail.cf

Testen Sie den Versand von eMail mit den Befehl:

/usr/lib/sendmail -v receiver@mail.isis.de
From: sender@mail.isis.de

Hallo
<Strg-D>

Setzen Sie in dem Befehl sinnvolle eMail-Adressen ein. Die eMail sollte den Empfänger erreichen.

4.5.1.3. Sendmail Konfgurieren unter Solaris

Diese Anleitung habe ich unter Solairs 6 und Solaris 8 getestet. Sollten Sie damit keinen Erfolg haben, lesen Sie bitte das Buch "sendmail" vom O'Reilly Verlag.

Tragen Sie in /etc/hosts Ihren mailserver mit dem Namen mailhost ein. Z.B. so:

```
192.168.1.12 mail.isis.de mailhost
```

Kopieren Sie /etc/mail/subsidiary.cf nach /etc/mail/sendmail.cf.

Testen Sie den Versand von eMail mit den Befehl:

/usr/lib/sendmail -v receiver@mail.isis.de
From: sender@mail.isis.de

Hallo
<Strg-D>

Setzen Sie in dem Befehl sinnvolle eMail-Adressen ein. Die eMail sollte den Empfänger erreichen.

4.5.2. ssmtp Konfigurieren

Dieses Kapitel betrifft nur Windows Systeme.

ssmtp simuliert das Programm sendmail, welches unter Unix das Standardprogramm zum Senden von eMails ist.

Im Installationsprogramm von CygWin gibt es einen kleinen Fehler. Es fehlt ein Link, der es ermöglicht, ssmtp wie das klassische sendmail aufzurufen. Fügen Sie diesen Link manuell hinzu:

cd /usr/lib
ln -s /usr/sbin/ssmtp.exe sendmail

Legen Sie sich die Konfigurationsdatei /etc/ssmtp/ssmtp.conf an, mit folgendem Inhalt:

```
mailhub=mail.isis.de:25
#hostname=meinpc.local
FromLineOverride=YES
```

Mailhub ist der Name und Portnummer Ihres eMail Servers (z.B. Microsoft Exchange). Die Portnummer ist eigentlich immer 25.

Hostname ist der Name des PC's. Diese Zeile ist in der Regel nicht nötig und kann daher wie oben gezeigt auskommentiert werden.

Die letzte Zeile legt fest, dass die From: Zeile in der eMail unverändert gesendet wird.

Testen Sie den Versand von eMail mit den Befehl:

/usr/lib/sendmail -v receiver@mail.isis.de
From: sender@mail.isis.de

Hallo
<Strg-D>

Setzen Sie in dem Befehl sinnvolle eMail-Adressen ein. Die eMail sollte den Empfänger erreichen.

4.5.3. eMail senden

Sie haben nun entweder sendmail oder ssmtp auf Ihren Computer installiert und sind damit bereit, eMails zu senden.

Die einfachtste Art, eine eMail zu senden ist diese:

/usr/lib/sendmail empfänger@irgendwo.de
Hallo!
<Strg-D>

Nur wenige Mailserver akzeptieren solche eMails, weil der Header fehlt. Die eMail würde beim Empfänger völlig anonym erscheinen.

Die obige Mail wäre mit einem Brief vergleichbar, den Sie ohne Beschriftung in den Briefkaten des Empfänger werfen lassen. Innen auf dem Briefpapier stände einfach nur "Hallo!". Viele Mailserver lassen solche anonymen eMails nicht mehr zu, und verlangen wenigstens nach einem Absendernamen.

Eine vollständige eMail sendet man so:

/usr/lib/sendmail klara.musterfrau@irgendwo.de
From: "Michael Mustermann" <michael.mustermann@irgendwo.de>
To: "Klara Musterfrau" <klara.musterfrau@irgendwo.de>
Subject: Testmail

Hallo, das ist ein Test!
<Strg-D>

Die Leerzeile zwischen dem Header und dem Text ist wichtig. Beim Absender und Empfänger können Sie die Namen auch weglassen und einfach nur die eMail Adressen benutzen. Für die Darstellung im eMails Client (z.B. Outlook) ist Angabe von richtigen Namen aber vorteilhaft. Beispiel:

/usr/lib/sendmail klara.musterfrau@irgendwo.de
From: michael.mustermann@irgendwo.de
To: klara.musterfrau@irgendwo.de
Subject: Testmail

Hallo, das ist ein Test!
<Strg-D>

Wenn Sie Probleme beim Versand von eMails haben, dann starten Sie sendmail mit der Option -v (vor der Empfängeradresse). Das Programm zeigt dann einige Informationen über Erfolg oder Misserfolg an.

4.5.4. eMail empfangen

Um eMails von einem Mailserver zu empfangen, können Sie das Programm fetchmail benutzen.

Legen Sie sich die Konfigurationsdatei /etc/fetchmailrc nach folgendem Schema an (alles in eine Zeile):

```
poll mail.isis.de protocol POP3 user mustermann pass geheim mda "cat >`mktemp
/tmp/mail.XXXXXXXX`"
```

Sie können mehrere POP3 Postfächer abrufen, indem Sie für jedes Postfach eine solche Zeile in die Datei schreiben.

In diesem Fall ist mail.isis.de der Mailserver. Das POP3 Konto hört auf den Namen mustermann, und das Passwort ist geheim. Wenn Sie mehrere Postfächer abrufen wollen, tragen Sie in diese Konfigurationsdatei einfach entsprechend mehrere Zeilen ein.

Geben Sie den Befehl

chmod 0710 /etc/fetchmailrc

ein, weil fetchmail die Datei sonst nicht lesen will. Nun können Sie eMail abrufen, indem Sie fetchmail so starten:

fetchmail -f /etc/fetchmailrc

Mehr Statusinformationen erhalten sie mit der Option -v.

Die empfangenen Emails werden jeweils in eigene Dateien unter /tmp/mail.??????? abgelegt. Anstelle der Fragezeichen werden zufällige Zeichenfolgen benutzt. Der Befehl mktemp stellt sicher, dass keine Dateinamen doppelt vergeben werden.

Sie können fetchmail auch als Dämon starten, indem Sie die Option -d 300 beim Start mit angeben. Dann läuft das Programm im Hintergrund und prüft alle 300 Sekunden auf neue eMails. Mit der Option --quit Können Sie den Dämon wieder beenden:

fetchmail -f /etc/fetchmailrc -d 300

fetchmail --quit

Eine typische eMail-Datei sieht etwa so aus:

```
Return-Path: <root@linux.local>
Date: Wed, 12 Feb 2003 17:36:42 +0100
From: root <root@linux.local>
Message-Id: <200302121636.h1CGagJ16586@linux.local>
To: s.frings@linux.local
Subject: Test 3

Hallo Du Schlafmütze!
```

Jedes Programm, das an der Übertragung der Mail beteiligt ist, fügt weitere Informationen in den Header ein. Wundern Sie sich also nicht, wenn Ihre Mails sehr viel grössere Header haben.

4.5.5. eMail Verarbeiten

Sie wissen nun, wie Sie eMails aus POP3 Postfächern abrufen und in Dateien speichern. Doch wie sorgt man dafür, dass diese sofort nach dem Empfang von irgendwelchen Programmen verarbeitet werden?

Ändern Sie die /etc/fetchmailrc wie folgt ab:

```
poll mail.isis.de protocol POP3 user mustermann pass geheim mda "/usr/local/bin/popmail"
```

(Schreiben Sie dies bitte wieder in eine einzige Zeile, sonst funktioniert es nicht.)

Immer, wenn eine eMail empfangen wird, startet fetchmail nun das Programm popmail und übergibt diesem die Mail. Das Programm popmail könnte ein selbstgemachtes Shell-Script dieser Art sein:

```
#!/bin/sh
filename=`mktemp /tmp/mail.XXXXXXX`
cat >$filename
echo "Sie haben Post! (in Datei $filename)"
```

Falls sie gerade beabsichtigen, diesen Code abzutippen, dann vergessen sie nicht, die x-Rechte einzuschalten, sonst dürfen Sie das Script nicht starten:

chmod a+rx /usr/local/bin/popmail

Nun werden die empfangenen Mails wiederum in Dateien gespeichert, aber zusätzlich erscheint die Meldung:

```
"Sie haben Post! (in Datei /tmp/mail.QE4F3AB9)"
```

An Stelle des simplen echo-Befehls können Sie auch andere Programme aufrufen. Sie werden in späteren Kapiteln sinnvollere Anwendungsbeispiele finden.

4.5.6. Formail anwenden

Das Programm formail ist sehr nützlich, wenn Sie aus eMail Dateien in ihre einzelnen Bestandteile zerlegen möchten oder sie verändern möchten.

Nehmen wir die eMail aus dem vorletzten Kapitel als Vorbild:

```
Return-Path: <root@linux.local>
Date: Wed, 12 Feb 2003 17:36:42 +0100
From: root <root@linux.local>
Message-Id: <200302121636.h1CGagJ16586@linux.local>
To: s.frings@linux.local
Subject: Test 3

Hallo Du Schlafmütze!
```

Angenommen, sie möchten nur den Absender haben, dann geht das so:

formail -zx From: < /tmp/mail.QE4F3AB9

Um aus der Mail nur einige bestimmte Header-Zeilen und den Text zu extrahieren, geben Sie dies ein:

formail -k -X From: -X Subject: < /tmp/mail.QE4F3AB9

Um aus der Mail nur den Text zu entnehmen und alle Header abzuschneiden, verwenden Sie den Befehl:

formail -I "" < /tmp/mail.QE4F3AB9

Der Nächste Befehl ändert den Empfänger der Mail:

formail -f -I "To: musterman@irgendwo.de" < /tmp/mail.QE4F3AB9

Alle diese Befehle ändern nicht die Datei selbst, sondern zeigen das Ergebnis auf dem Bildschirm an. Sie können diese Ausgaben jeweis in Dateien oder in Variablen umleiten:

formail -zx From: < /tmp/mail.QE4F3AB9 > neuer_dateiname

from=`formail -zx From:< /tmp/mail.QE4F3AB9`
echo "Der Absender ist $from"

4.6. Arbeiten mit FTP

FTP Schnittstellen ermöglichen Ihnen die Übertragung von Dateien zwischen mehreren Computern (auch im Internet). Das FTP Protokoll ist ist der kleinste gemeinsame Nenner, den alle Computer unterstützen.

Bei FTP unterscheidet man zwischen dem FTP Server, und dem FTP Client.

Der Server stellt ein oder mehrere Verzeichnisse seiner Festplatte für die Clients zur Verfügung. Unter Windows brauchen Sie dazu den War FTP Dämon, der anhand seiner grafischen Oberfläche konfiguriert wird.

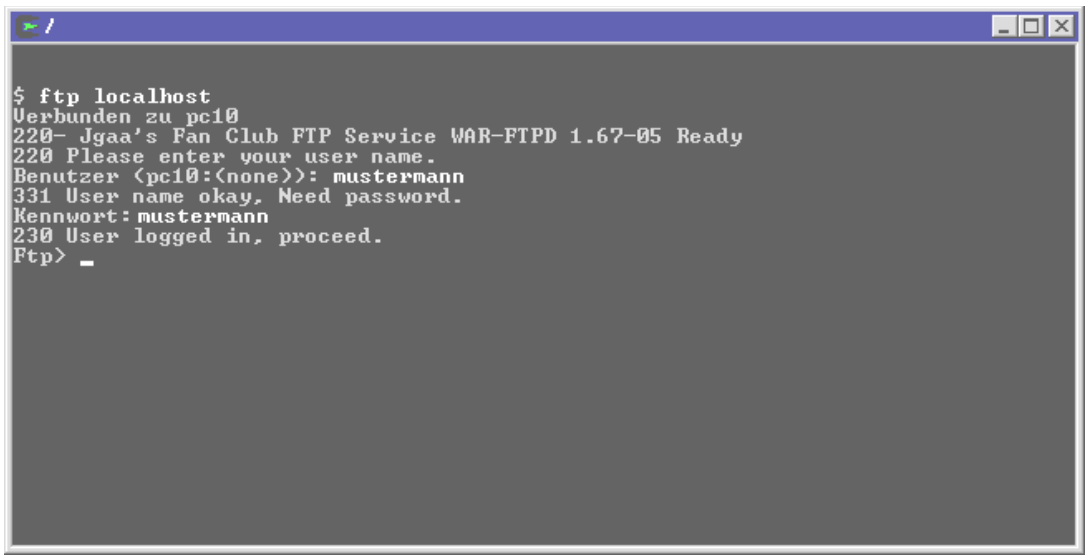
Unter Unix brauchen Sie den inet-Daemon, der zur Standardinstallation gehört. Jeder reguläre Unix User kann sich mit einem FTP Client einloggen und hat Zugriff auf alle Verzeichnisse und Dateien, auf die er auch in einer Telnet Session zugriff hätte. Das heist, dass man den FTP Server unter Unix nicht extra konfigurieren muss.

Der Client verbindet sich mit dem Server und sendet oder empfängt ausgewählte Dateien. Die nächsten Kapitel beschreiben, wie das geht.

4.6.1. FTP Client benutzen

Jedes Betriebssystem wird mit dem Standard FTP Client ausgeliefert. Dieser ist für interaktives Arbeiten gut geeignet, jedoch für automatische Dateiübertragungen nicht so gut.

Den FTP Client startet man in der Shell oder in der Dos-Eingabeaufforderung mit dem Befehl "ftp Servername". Anstelle des Servernamens kann man auch dessen IP-Adresse angeben.



Nach dem Verbindungsaufbau werden Sie zur Eingabe des Usernamens und Passwortes aufgefordert.

Ich empfehle, danach immer gleich die folgenden Befehle einzugeben:

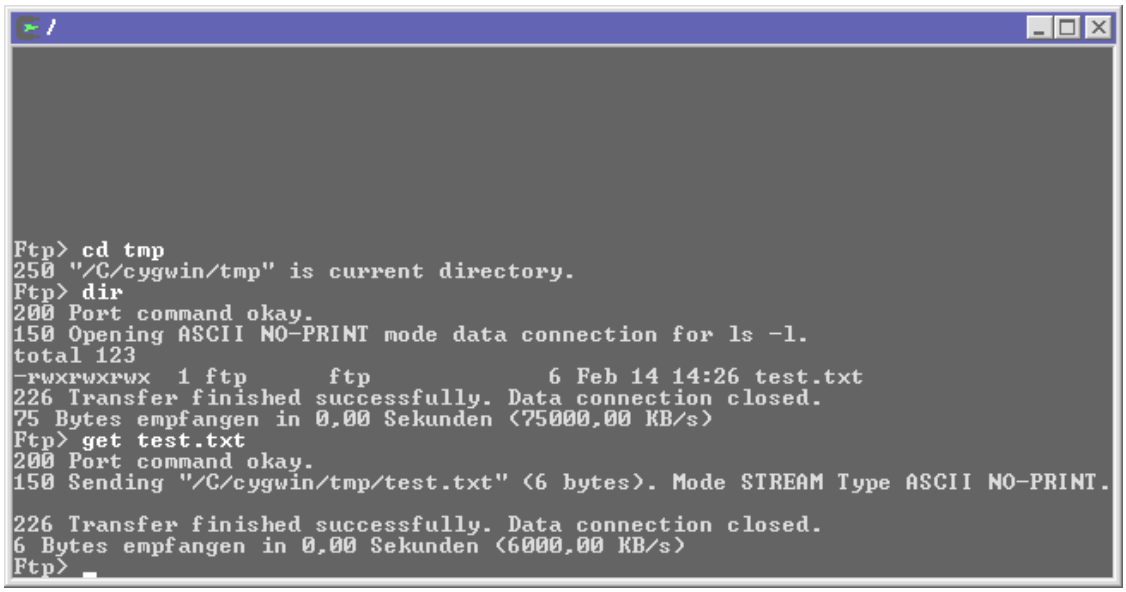
binary	Schaltet von 7 Bit Transfer auf 8 Bit um. Das ist für alle Dateien nötig, ausser ASCII Texte. Das Gegenteil (text) konvertiert die Zeilenumbrüche und entfernt in jedem Byte das höchste Bit.
hash	Aktiviert eine Statusanzeige, nützlich bei großen Dateien

Zur Datenübertragung braucht man einen oder mehrere dieser Befehle:

cd Verzeichnis	wechselt in ein anderes Verzeichnis auf der Seite des Servers
dir	listet das Inhaltsverzeichnis auf
rm Dateiname	löscht eine Datei
rmdir Verzeichnis	löscht ein Verzeichnis
mkdir Verzeichnis	erstellt ein Verzeichnis

put Dateiname	lädt eine Datei auf den FTP Server hoch
get Dateiname	lädt eine Datei vom FTP Server herunter
mput Dateiname	lädt mehrere Dateien auf den FTP Server hoch. Dateiname mit Jokerzeichen
mget Dateiname	lädt mehrere Dateien vom FTP Server herunter. Dateiname mit Jokerzeichen
prompt	schaltet die Sicherheitsabfrage bei mput und mget aus
bye	Beendet die FTP Verbindung

Das folgende Bildschirmfoto zeigt, wie ich die Datei /tmp/test.txt downloade.



4.6.2. Automatische FTP Downloads

Um Dateien automatisch zu übertragen, verwenden Sie das Programmpaket ncftp. Die dazu interessanten Befehle lauten ncftpput und ncftpget. Unter Windows müssen Sie ncftp in einer CygWin Bash Shell starten.

Downloaden Sie Dateien vom Server mit dem Befehl:

ncftpget -u Username -p Passwort Servername Zielverzeichnis Quelldateien

Als Quelldateien können Sie einzelne Dateien angeben (z.B. /tmp/test.txt) oder auch Jokerzeichen benutzen. Wenn Sie unter Unix Jokerzeichen benutzen, müssen Sie aber unbedingt einfache Anführungsstriche benutzen, weil sonst die Shell das Jokerzeichen durch alle passenden lokalen Dateinamen ersetzt - und das wollen Sie nicht.

Zwei richtige Beispiele:

ncftpget -u mustermann -p test MeinServer /download /tmp/test.txt
ncftpget -u mustermann -p test MeinServer /download '/tmp/test.*'

Dieser Befehl holt test.txt vom Server ab und speichert sie in das Verzeichnis download.

Wenn Sie die Option -DD anwenden (also ncftpget -DD -u ...), werden die Dateien nach erfolgreicher Übertragung vom Server gelöscht.

Sie können auch mehrere einzelne Quelldateien zum Download benennen.

Mit der Option -R downloaden sie ein ganzes Verzeichnis. z.B.:

ncftpget -R -u mustermann -p test MeinServer /download /tmp

Nach diesem Download haben Sie auf Ihrem Rechner das Verzeichnis /download/tmp mit allen Dateien.

Ncftpget liefert einen Return-Code, der Aufschluss über Erfolg oder Misserfolg anzeigt. Innerhalb von Scripten können Sie damit nach erfolglosen Übertragungen entsprechend reagieren.

Sie können den Returncode anzeigen, indem Sie unmittelbar nach der Datenübertragung den Befehl **echo $?** eingeben. 0 bedeutet, dass die Übertragung erfolgreich war. Andere Zahlen zeigen unterschiedliche Fehler an.

Die manual-Page (man ncftpget) enthält an Ende ein Liste der möglichen Fehlercodes.

4.6.3. Automatische FTP Uploads

Das Senden von Dateien geht prinzipiell genau so, wie das Empfangen.

Uploaden Sie Dateien vom Server mit dem Befehl:

ncftpput -u Username -p Passwort Servername Zielverzeichnis Quelldateien

Als Quelldateien können Sie einzelne Dateien angeben (z.B. /tmp/test.txt) oder auch Jokerzeichen benutzen. Wenn Sie unter Unix Jokerzeichen benutzen, müssen Sie aber unbedingt einfache Anführungsstriche benutzen, weil sonst die Shell das Jokerzeichen durch alle passenden lokalen Dateinamen ersetzt - und das wollen Sie nicht.

Zwei richtiges Beispiele:

ncftpput -u mustermann -p test MeinServer /upload /tmp/test.txt

ncftpput -u mustermann -p test MeinServer /upload '/tmp/test.*'

Dieser Befehl sendet test.txt zum Server, und zwar in das Verzeichnis /upload.

Wenn Sie die Option -DD anwenden (also ncftpput -DD -u ...), werden die Dateien nach erfolgreicher Übertragung auf Ihrem Rechner gelöscht.

Sie können auch mehrere einzelne Quelldateien zum Upload benennen.

Mit der Option -R uploaden sie ein ganzes Verzeichnis. z.B.:

ncftpput -R -u mustermann -p test MeinServer /upload /tmp

Nach diesem Upload haben Sie auf dem Server das Verzeichnis /upload/tmp mit allen Dateien.

Ncftpput liefert einen Return-Code, der Aufschluss über Erfolg oder Misserfolg anzeigt. Innerhalb von Scripten können Sie damit nach erfolglosen Übertragungen entsprechend reagieren.

Sie können den Returncode anzeigen, indem Sie unmittelbar nach der Datenübertragung den Befehl echo $? eingeben. 0 bedeutet, dass die Übertragung erfolgreich war. Andere Zahlen zeigen unterschiedliche Fehler an.

Die manual-Page (man ncftpput) enthält an Ende ein Liste der möglichen Fehlercodes.

4.7. MySQL Anwenden

SQL Datenbanken sind komplizierte Programme, weil sie vielseitig, schnell und dennoch einfach anwendbar sein sollen. Genau aus diesem Grund sind sie auch so weit verbreitet.

MySQL ist ein kostenloser SQL Server. Sein Funktionsumfang ist minimalistisch, doch für SMS Anwendungen völlig ausreichend.

Zu MySQL gehört eine knappe, aber vollständige Dokumentation. Da die Dokumentation für Anfänger schwierig ist, habe ich mich entschlossen, Ihnen in diesem Kapitel die Grundlagen von MySQL zu zeigen.

Das Kapitel beschreibt nicht alle Funktionen und Schikanen von MySQL, sondern es soll Ihnen lediglich einen einfachen Einstieg ermöglichen. Die Details können Sie sich später anhand der Originalen MySQL Dokumentation aneignen - falls dazu überhaupt Bedarf besteht.

Tippen Sie alle Fett geschriebenen Befehle der Reihe nach ab, um die Beispiele nachzuvollziehen. Sie bauen aufeinander auf, daher lassen Sie bitte keine Befehle aus.

Wenn Sie sich bereits mit SQL auskennen, können Sie das Kapitel überspringen.

4.7.1. Das Funktionsprinzip von MySQL

MySQL ist eine Client-Server Anwendung. Das bedeutet, dass Sie einen Datenbank-Server betreiben, sowie einen oder mehrere Datenbank-Clients. Beide können physikalisch auf dem selben Computer laufen.

Der Server speichert Ihre Daten, während der Client diese verändert oder ausliest. Beliebig viele Clients können gleichzeitig auf die Datenbanken zugreifen.

Auf den folgenden Seiten werde ich ab und zu die Begriffe endlos und beliebig gebrauchen. Tatsächlich gibt es Betriebssystemabhängige Einschränkungen, die im "echten" Leben aber keine Rolle spielen.

Ein MySQL Server speichert beliebig viele Datenbanken.

Jede Datenbank beinhaltet beliebig viele Tabellen.

Jede Tabelle hat beliebig viele Reihen und Spalten. In den Tabellenfeldern werden die Daten gespeichert.

Eine Tabellenreihe wird als Datensatz bezeichnet. Eine Adresse wäre z.B. ein Datensatz. Die einzelnen Bestandteile der Adresse werden in den Spalten gespeichert und werden Datenfelder genannt. Jede Tabellenspalte hat ein bestimmtes Format, z.B. Text, Datum oder Zahl. Das Format der Spalten bestimmen Sie, wenn Sie eine Datenbank anlegen.

Sie bestimmen beim Anlegen einer Datenbank, wieviele Spalten diese haben wird. Die Anzahl der Reihen ist offen und beginnt zunächst mit Null. Man kann nachträglich Spalten hinzufügen oder löschen, wenn man dies für nötig hält.

4.7.2. Datentypen

Wenn Sie eine Datenbank anlegen, geben Sie der Datenbank einen Namen, sonst nichts weiter. Mit Hilfe des Datenbank-Namens können Sie mehrere Tabellen gruppieren und so für Ordnung sorgen.

In dieser Datenbank können Sie nun Tabellen anlegen. Dabei legen Sie fest, welche Datentypen Sie in den einzelnen Spalten speichern werden. Die Anzahl der Reihen bleibt zunächst unbestimmt.

Wenn Sie eine Tabelle gestalten, benutzen sie immer den kleinsten Datentyp, der für Ihre Anwendung ausreicht. Denn so sparen sie nicht nur Speicherplatz, sondern sorgen zugleich auch für die bestmögliche Geschwindigkeit.

Die folgende Tabelle gibt einige Datentypen wieder, die MySQL unterstützt:

INTEGER	ganze Zahl im Bereich -2147483648 bis 2147483647
BIGINT	ganze Zahl im Bereich -9223372036854775808 bis 9223372036854775807
FLOAT	Fliesskommazahl im Bereich -3.4E38 bis +3.4E38 mit 4 Byte Genauigkeit
DOUBLE	Fliesskommazahl im Bereich -1.79E308 bis +1.79E308 mit 8 Byte Genauigkeit
DATE	Datum im Format YYYY-MM-DD
TIME	Zeit im Format HH:MM:ss
DATETIME	Datum und Zeit im Format YYYY-MM-DD HH:MM:ss
TIMESTAMP	Ein Zeitstempel im Format YYYYMMDDHHMMss. Wird automatisch bei jedem Schreibzugriff auf den Datensatz gesetzt.
CHAR(n)	Zeichenkette mit n Zeichen (max. 255)
VARCHAR	Zeichenkette mit 0-255 Zeichen
TEXT	Zeichenkette mit 0-65535 Zeichen
BLOB	Zeichenkette mit 0-65535 Zeichen
LONGTEXT	Zeichenkette mit 0-4294967295 Zeichen
LONGBLOB	Zeichenkette mit 0-4294967295 Zeichen

Wenn Sie ein Datum speichern, dürfen Sie die Jahreszahl auch zweistellig schreiben. MySQL wandelt die Jahreszahl automatisch in vierstellige Darstellung um. MySQL Erlaubt die Verwendung von 00 in Datum und Uhrzeit, also z.B. 0000-00-00 00:00:00.

Zeichenketten vom Typ CHAR haben einen festen Speicherplatzbedarf. Sie können kürzere Zeichenketten speichern, aber es wird immer der festgelegte Speicherplatz belegt. Vorteil: Der Zugriff auf CHAR ist sehr schnell. Nachteil: Es wird Speicherplatz verschwendet, wenn der reservierte Speicherplatz nicht voll genutzt wird.

Zeichenketten vom Typ TEXT, LONGTEXT, BLOB und LONGBLOB belegen genau soviel Speicher, wie notwendig. Dafür ist aber der Zugriff darauf langsamer.

Wenn Sie Zeichenketten vergleichen oder suchen, sind TEXT und LONGTEXT von der groß/Kleinschreibung unabhängig, während bei BLOB und LONGBLOB die groß/Kleinschreibung beachtet wird.

Wenn Sie Zeichenketten speichern, werden überschüssige Leerzeichen auf der rechten Seite abgeschnitten. Leerzeichen auf der linken Seite bleiben jedoch erhalten.

4.7.3. Optionen zu Datentypen

Bei INTEGER und BIGINT können Sie die anzuzeigende Länge in Klammern angeben:

```
INTEGER(n)
BIGINT(n)
```

Beim Auslesen werden die Zahlen dann rechtsbündig in der entsprechenden Spaltenbreite dargestellt. Sie können dennoch auch grössere Zahlen speichern. In Kombination mit der Option ZEROFILL werden Integer Zahlen mit Nullen anstatt Leerzeichen aufgefüllt:

```
INTEGER(n) ZEROFILL
BIGINT(n) ZEROFILL
```

Wenn in einer Spalte nichts gespeichert wird, zeigt MySQL dies durch das Schlüsselwort NULL an. Durch die Option NOT NULL erzwingen Sie, dass die Spalte ausgefüllt werden muss, z.B:

```
INTEGER NOT NULL
```

Durch die Option DEFAULT legen Sie einen Standardwert fest, der automatisch benutzt wird, wenn Sie einen neuen Datensatz ohne Angabe der Werte anlegen, z.B.:

```
CHAR(5) DEFAULT "Hallo"
```

Durch die Option AUTO_INCREMENT erhalten Sie eine Spalte, deren Werte automatisch hochgezählt werden. Dabei wird mit der 1 begonnen.

Jede Tabelle sollte eine Spalte mit der Option PRIMARY KEY NOT NULL haben. Jede Reihe in der Tabelle bekommt so zwangsweise ein eindeutiges Kennzeichen in dieser Spalte. Sie muss immer ausgefüllt sein und muss einen eindeutigen Wert haben, der in keiner Reihe mehrfach vorkommt. Meistens ist die erste Spalte einer Tabelle so festgelegt:

```
id INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL
```

4.7.4. Eine Datenbank mit zwei Tabellen anlegen

Starten Sie den MySQL Client durch Eingabe von **mysql -u root -p**. Geben Sie das Passwort ein, dass Sie während der Installation festgelegt haben.

Nun legen Sie eine Datenbank an, indem Sie **create database adressen;** eingeben. Die neue Datenbank heisst nun adressen. Teilen Sie MySQL mit, dass Sie bei allen folgenden Befehlen diese Datenbank benutzen möchten:

use adressen;

Legen Sie in dieser Datenbank eine Tabelle an und bestimmen Sie die Datentypen der Spalten:

```
create table anschriften(
    id INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
   name CHAR(30),
   vorname CHAR(30),
    strasse CHAR(30),
    plz CHAR(5),
    ort CHAR(30)
);
```

Legen Sie eine zweite Tabelle in der Adressdatenbank an:

```
create table kontostand(
    id INTEGER PRIMARY KEY NOT NULL,
    betrag FLOAT,
    stand TIMESTAMP
 );
```

Sie können nun Adressen speichern, sowie Kontostände. Natürlich könnten Sie auch alles in einer einzigen Tabelle speichern, aber das wäre ja langweilig. Hier geht es darum, zu sehen, wie man mehrere Tabellen in einer Datenbank anlegt und zusammen benutzt.

Um sich später nochmal anzusehen, wie die Spalten der Tabellen festgelegt sind, geben Sie diese Befehle ein:

show databases;

use adressen;

show tables;

describe anschriften;

describe kontostand;

Sie können nun beginnen, diese Tabellen mit Adressen und Kontoständen zu füllen. Der Zeitstempel beim Kontostand wird bei jeder Änderung automatisch gesetzt. Die ID-Nummern der Adressen werden automatisch fortlaufend vergeben, während Sie die ID-Nummern der Kontostände selbst manuell vergeben.

Das nächste Kapitel zeigt Ihnen, wie das geht.

4.7.5. Tabellen mit Daten Füllen

Geben Sie eine erste Adresse in die Tabelle ein:

```
insert into anschriften set
   name="Mustermann",
   vorname="Markus",
   strasse="Bachstrasse 13",
   plz="11111",
   ort="Irgendwo";
```

Die Reihenfolge der Spalten können sie ruhig vertauschen, sie muss nicht mit der Tabellen-Definition übereinstimmen. Sie dürfen auch einzelne Spalten auslassen, z.B.:

```
insert into anschriften set
   vorname="Michaela",
   name="Musterfrau";
```

Die Zeilenumbrüche während der Befehlseingabe dienen nur dazu,die Darstellung übersichtlicher zu gestalten. Sie können SQL Befehle immer auch in eine einzige Zeile schreiben.

Eine andere gleichwertige Schreibweise des obigen Befehls ist diese:

```
insert into anschriften (vorname, name)
  values ("Michaela", "Musterfrau");
```

Schauen Sie sich nun die Tabelle an, indem Sie **select * from anschriften;** eingeben. Sie sehen, dass die id-Spalte automatisch fortlaufend vergeben wurde und dass die leeren Felder von Michaela Musterfrau durch das Schlüsselwort NULL gekennzeichnet werden:

```
mysql> select * from anschriften;
+----+------------+----------+----------------+-------+----------+
| id | name       | vorname  | strasse        | plz   | ort      |
+----+------------+----------+----------------+-------+----------+
|  1 | Mustermann | Markus   | Bachstrasse 13 | 11111 | Irgendwo |
|  2 | Musterfrau | Michaela | NULL           | NULL  | NULL     |
+----+------------+----------+----------------+-------+----------+
2 rows in set (0.00 sec)
```

Angenommen, sie möchten nun nachträglich die fehlende Postleitzahl und den Ort eintragen, dann tun Sie das mit dem Befehl

```
update anschriften set
   plz="22222",
   ort="Wasserdorf"
   where id=2;
```

Nun wissen Sie auch, warum ein eindeutiges ID-Feld so wichtig ist - nämlich um später eine Reihe in der Tabelle (einen Datensatz) exakt zu bezeichnen. Mit dem Schlüsselwort where geben Sie an, wo genau Sie die Tabelle ändern wollen. Das nächste Kapitel erklärt den where Ausdruck etwas detailierter.

Das Ergebnis des letzten update Befehles wird Sie nicht überraschen:

```
mysql> select * from anschriften;
+----+------------+----------+----------------+-------+------------+
| id | name       | vorname  | strasse        | plz   | ort        |
+----+------------+----------+----------------+-------+------------+
|  1 | Mustermann | Markus   | Bachstrasse 13 | 11111 | Irgendwo   |
|  2 | Musterfrau | Michaela | NULL           | 22222 | Wasserdorf |
+----+------------+----------+----------------+-------+------------+
2 rows in set (0.02 sec)
```

Legen Sie jetzt bitte noch eine dritte Adresse an, damit Sie die Beispiele der nächsten Kapitel besser nachvollziehen können:

```
insert into anschriften set
   name="Mustermann",
   vorname="Sohnemann",
   strasse="Bachstrasse 13",
   plz="11111",
   ort="Irgendwo";
```

4.7.6. Tabellen Auslesen

Ihre Datenbank sollte momentan so aussehen:

```
mysql> select * from anschriften;
+----+------------+-----------+----------------+-------+------------+
| id | name       | vorname   | strasse        | plz   | ort        |
+----+------------+-----------+----------------+-------+------------+
|  1 | Mustermann | Markus    | Bachstrasse 13 | 11111 | Irgendwo   |
|  2 | Musterfrau | Michaela  | NULL           | 22222 | Wasserdorf |
|  3 | Mustermann | Sohnemann | Bachstrasse 13 | 11111 | Irgendwo   |
+----+------------+-----------+----------------+-------+------------+
3 rows in set (0.00 sec)
```

Die einfachste Form des select Befehles haben Sie bereits mehrfach angewendet und sind damit vertraut.

Ebenfalls recht einfach ist es, die Anzahl der Reihen zu zählen, und zwar mit

select count(*) from anschriften;

Das where Schlüsselwort haben Sie im letzten Kapitel bereits einmal benutzt, um beim update Befehl die gewünschte Tabellenreihe zu bezeichnen. Dies können Sie auch beim select Befehl tun:

select * from anschriften where id=2;

So lassen Sie sich nur die Reihe 2 anzeigen. Sie können auch alle Reihen anzeigen, in denen andere Bedingungen zutreffen, z.B. so:

select * from anschriften where plz="11111";

select * from anschriften where name="Mustermann";

select * from anschriften where name="Mustermann" and vorname<>"Markus";

Der dritte Befehl zeigt alle Anschriften von Familie Mustermann an, aber nicht den Markus (in unserem Fall ist das nur der Sohnemann).Mit and und or können Sie mehrere Ausdrücke verknüpfen.

Sie können auch nach halben Zeichenketten suchen, und zwar so:

select * from anschriften where name like "Muster%";

Sie bekommen so alle Anschriften angezeigt, deren Name mit Muster beginnt. Das geht auch anders herum:

select * from anschriften where name like "%mann";

Das Prozent Zeichen ist ein Joker für mehrere Buchstaben, während der Unterstrich "_" ein Jokerzeichen für genau ein einziges Zeichen ist.

So, wie sie beim select befehl mit where angeben, welche Zeilen Sie sehen wollen, bestimmen Sie auch beim update Befehl, welche Zeilen sie verändern wollen. Wenn Sie das where Schlüsselwort einfach weglassen, arbeiten Sie immer mit der ganzen Tabelle (jeder Reihe).

Bei großen Tabellen möchten Sie sicher nicht immer alle Spalten auf einmal sehen - schliesslich ist Ihr Bildschirm nicht unendlich groß. Mit * selektieren Sie alle Spalten. Stattdessen können Sie auch die Spalten angeben, die Sie sehen möchten:

```
mysql> select id,name,ort from anschriften;
+----+------------+------------+
| id | name       | ort        |
+----+------------+------------+
|  1 | Mustermann | Irgendwo   |
|  2 | Musterfrau | Wasserdorf |
|  3 | Mustermann | Irgendwo   |
+----+------------+------------+
3 rows in set (0.00 sec)
```

Bei der Eingabe von SQL Befehlen sollten Sie Zeichenketten und Datumswerte immer zwischen doppelte Anführungsstriche (") schreiben, und Zahlen sollten Sie immer ohne Anführungsstriche schreiben. Die Beispiele der beiden letzten Kapitel haben Ihnen dies vorgeführt.

Bevor Sie jetzt lange über die Postleitzahl nachgrübeln: Die Postleitzahl ist eine Zeichenkette und keine Zahl, weil die entsprechende Spalte der Tabelle so angelegt wurde (char(5)). MySQL kann Zahlen auch automatisch in Zeichenketten umwandeln und umgekehrt, aber nicht in jedem Fall. Durch die korrekte Verwendung von doppelten Anführungsstrichen um Zeichenketten vermeiden Sie Fehlermeldungen und Konvertierungsfehler.

4.7.7. Tabellen Verknüpfen

Nun möchte ich Ihnen zeigen, wie man zwei Tabellen miteinander verknüpft. Da wir bisher nur drei Adressen eingegeben haben, aber noch keine Kontostände, holen wir dies jetzt nach. Die nötigen Befehle sind Ihnen bereits bekannt.

Michaela Musterfrau hat 6.5 Euro

Markus Mustermann hat 18.3 Euro

Sohnemann Mustermann hat 23.1 Euro.

Diese Daten geben wir nun in die zweite Tabelle für die Kontostände ein. Die ID-Nummern vergeben wir manuell, und zwar entsprechend der Anschriften-Tabelle.

insert into kontostand set id=2, betrag=6.5;

insert into kontostand set id=1, betrag=18.3;

insert into kontostand set id=3, betrag=23.1;

Sie haben nun zwei Tabellen, eine mit den Adressen der drei Muster-Personen und eine mit deren Kontoständen:

```
mysql> select * from anschriften;
+----+------------+-----------+----------------+-------+------------+
| id | name       | vorname   | strasse        | plz   | ort        |
+----+------------+-----------+----------------+-------+------------+
|  1 | Mustermann | Markus    | Bachstrasse 13 | 11111 | Irgendwo   |
|  2 | Musterfrau | Michaela  | NULL           | 22222 | Wasserdorf |
|  3 | Mustermann | Sohnemann | Bachstrasse 13 | 11111 | Irgendwo   |
+----+------------+-----------+----------------+-------+------------+
3 rows in set (0.00 sec)


mysql> select * from kontostand;
+----+--------+----------------+
| id | betrag | stand          |
+----+--------+----------------+
|  1 |   18.3 | 20030302131155 |
|  2 |    6.5 | 20030302131206 |
|  3 |   23.1 | 20030302131230 |
+----+--------+----------------+
3 rows in set (0.00 sec)
```

Wie sie sehen, hat der SQL Server die Spalte mit dem Namen "stand" automatisch ausgefüllt, genau so, wie ich es Ihnen versprochen hatte. An dieser Spalte können Sie immer sehen, wann der Kontostand zuletzt geändert wurde.

Nun kommen wir zu dem eigentlichen Thema dieses Kapitels, dem verknüpfen von mehreren Tabellen.

Einen Teil dieser Aufgabe haben Sie gerade erledigt,nämlich, indem Sie zusammengehörende Reihen durch gleiche Id-Nummern gekennzeichnet haben. Dadurch haben Sie die Tabellen bereits verknüpft. Die Tabelle der Kontostände speichert zwar keine Namen, doch mit Hilfe der Id-Nummer können sie in die andere Tabelle schauen und den zugehörigen Namen auslesen.

Genau das müssen Sie nicht manuell tun. Sie können auch den SQL Server beauftragen, mehrere Tabellen verknüpft und zusammenhängend anzuzeigen. Der dazu nötige Befehl ist ganz einfach:

```
mysql> select anschriften.id,name,vorname,betrag
-> from anschriften,kontostand
-> where anschriften.id=kontostand.id;
+----+------------+-----------+--------+
| id | name       | vorname   | betrag |
+----+------------+-----------+--------+
|  1 | Mustermann | Markus    |   18.3 |
|  2 | Musterfrau | Michaela  |    6.5 |
|  3 | Mustermann | Sohnemann |   23.1 |
+----+------------+-----------+--------+
3 rows in set (0.00 sec)
```

Hinter dem Schlüsselwort from geben Sie beide Tabellen an, aus denen Sie bestimmte Spalten abfragen wollen.

Hinter dem Befehl select geben Sie wie gehabt alle Spalten an, die Sie sehen wollen. Da es die Spalte id in beiden Tabellen gibt, müssen Sie zusätzlich die Tabelle nennen, aus der Sie die id Spalte sehen wollen. In diesem Fall ist das also die Spalte anschriften.id.

Hinter dem Schlüsselwort where müssen Sie die beiden Spalten benennen, durch die Ihre Tabellen verknüpft sind. In diesem Fall sind es die beiden Id-Spalten, also anschriften.id=kontostand.id.

Wenn Sie nicht alle Kontostände sehen, wollen, sondern nur die von Familie Mustermann, tun Sie das so:

```
select anschriften.id,name,vorname,betrag
   from anschriften,kontostand
   where anschriften.id=kontostand.id and name="Mustermann";
```

4.7.8. NULL Werte

Bitte geben Sie zur Vorbereitung diesen Befehl im SQL Client ein:

update anschriften set plz="" where id=3;

Ihre Anschriften-Tabelle sieht jetzt so aus:

```
mysql> select * from anschriften;
+----+------------+-----------+----------------+-------+------------+
| id | name       | vorname   | strasse        | plz   | ort        |
+----+------------+-----------+----------------+-------+------------+
|  1 | Mustermann | Markus    | Bachstrasse 13 | 11111 | Irgendwo   |
|  2 | Musterfrau | Michaela  | NULL           | 22222 | Wasserdorf |
|  3 | Mustermann | Sohnemann | Bachstrasse 13 |       | Irgendwo   |
+----+------------+-----------+----------------+-------+------------+
3 rows in set (0.00 sec)
```

Beachten Sie bitte, dass Michael Musterfrau keine Strasse hat, und Sohnemann Mustermann keine Postleitzahl. Merkwürdig erscheint hier , dass bei Michaela NULL steht, während das Feld bei Sohnemann einfach nur leer ist.

Das lässt sich wie folgt erklären:

In SQL Datenbänken ist "Nichts" etwas anderes, wie eine leere Zeichenkette.

"Nichts" wird durch das Schlüsselwort NULL angezeigt. Das bedeutet, dass die Strasse nicht eingegeben wurde, also keine Daten vorhanden sind.

Bei der Postleitzahl sieht es anders aus. Diese haben Sie gerade als Vorbereitung zu diesem Kapitel eingegeben, und zwar als leere Zeichenkette "".

```
Der Unterschied fällt besonders beim Select Befehl auf:
mysql> select * from anschriften where strasse="";
Empty set (0.00 sec)

mysql> select * from anschriften where strasse is NULL;
+----+------------+----------+---------+------+------------+
| id | name       | vorname  | strasse | plz  | ort        |
+----+------------+----------+---------+------+------------+
|  2 | Musterfrau | Michaela | NULL    | 22222| Wasserdorf |
+----+------------+----------+---------+------+------------+
1 row in set (0.00 sec)
```

Es gibt keine Anschrift, wo die Strasse "" ist, aber es gibt eine Adresse, wo die Strasse NULL ist.

Beachten Sie, dass sie den Ausdruck hinter die Schlüsselwörter where und NULL etwas anders benutzen müssen,als wenn Sie Zahlen oder Zeichenketten vergleichen:

select * from anschriften where strasse is NULL;

select * from anschriften where strasse is not NULL;

Eine kleine Anmerkung nebenbei: Wenn Sie ein bereits gesetztes Feld wieder löschen wollen, also auf NULL setzen wollen, tun Sie das so:

update anschriften set plz=NULL where id=3;

4.7.9. Weitere Schlüsselwörter für Select

Wenn Sie große Tabellen haben, sind die Schlüsselwörter "order by" und "limit" für Sie interessant.

Mit "order by" sortieren Sie die Ausgabe. Beispiel:

```
mysql> select * from anschriften order by plz;
+----+------------+-----------+----------------+-------+------------+
| id | name       | vorname   | strasse        | plz   | ort        |
+----+------------+-----------+----------------+-------+------------+
|  3 | Mustermann | Sohnemann | Bachstrasse 13 | NULL  | Irgendwo   |
|  1 | Mustermann | Markus    | Bachstrasse 13 | 11111 | Irgendwo   |
|  2 | Musterfrau | Michaela  | NULL           | 22222 | Wasserdorf |
+----+------------+-----------+----------------+-------+------------+
3 rows in set (0.00 sec)
```

Die Ausgabe wird so anhand der Postleitzahl sortiert ausgegeben. Benutzen Sie diese Funktion nur, wenn Sie Ihnen wirklich nützlich ist, denn das Sortieren kann bei vielen Reihen lange dauern und viel Arbeitsspeicher erfordern.

Rückwärts sortieren geht natürlich auch, und zwar so:

select * from anschriften order by ort desc;

In Zusammenhang mit sortieren Ausgaben macht es manchmal Sinn,die Ausgabe auf weniger Reihen zu beschränken.

Wenn Sie zum Beispiel die Person mit der kleinsten Postleitzahl sehen wollen, geben Sie ein:

```
mysql> select * from anschriften order by plz limit 1;
+----+------------+-----------+----------------+------+----------+
| id | name       | vorname   | strasse        | plz  | ort      |
+----+------------+-----------+----------------+------+----------+
|  3 | Mustermann | Sohnemann | Bachstrasse 13 | NULL | Irgendwo |
+----+------------+-----------+----------------+------+----------+
1 row in set (0.00 sec)
```

Durch das Schlüsselwort "limit" bestimmen Sie, wieviele Reihen Sie maximal sehen wollen.Im obigen Beispiel begrenzen Sie die Ausgabe auf eine einzige Reihe.

4.7.10. Spalten Ändern

Die Struktur einer SQL Tabelle ist veränderbar. Sie können Spalten hinzufügen, löschen und verändern.

Wir wollen nun zwei Spalten hinzufügen, für Geschlecht und Gebutsdatum:

```
mysql> alter table anschriften
-> add geschlecht CHAR(1),
-> add geburtsdatum DATE;
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0


mysql> select * from anschriften;
+----+------------+-----------+----------------+-------+------------+------------+--------------+
| id | name       | vorname   | strasse        | plz   | ort        | geschlecht | geburtsdatum |
+----+------------+-----------+----------------+-------+------------+------------+--------------+
|  1 | Mustermann | Markus    | Bachstrasse 13 | 11111 | Irgendwo   | NULL       | NULL         |
|  2 | Musterfrau | Michaela  | NULL           | 22222 | Wasserdorf | NULL       | NULL         |
|  3 | Mustermann | Sohnemann | Bachstrasse 13 | NULL  | Irgendwo   | NULL       | NULL         |
+----+------------+-----------+----------------+-------+------------+------------+--------------+
3 rows in set (0.00 sec)
```

Sie haben nun zwei Spalten hinzugefügt, die natürlich noch leer sind. Angenommen, sie stellen fest, dass manche Strassennamen länger als 30 Zeichen sind, dann müssen Sie die Spalte für die Strassennamen entsprechend verbreitern. Das tolle an MySQL ist, dass Ihnen die vorhandenen Strassennamen bei dieser Änderung nicht verloren gehen:

alter table anschriften modify strasse char(50);

Sie können jetzt längere Strassennamen eingeben, als bisher möglich.

Was jetzt noch fehlt, ist der Befehl, mit dem Sie ganze Spalten löschen:

```
mysql> alter table anschriften drop geschlecht;
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0


mysql> select * from anschriften;
+----+------------+-----------+----------------+-------+------------+--------------+
| id | name       | vorname   | strasse        | plz   | ort        | geburtsdatum |
+----+------------+-----------+----------------+-------+------------+--------------+
|  1 | Mustermann | Markus    | Bachstrasse 13 | 11111 | Irgendwo   | NULL         |
|  2 | Musterfrau | Michaela  | NULL           | 22222 | Wasserdorf | NULL         |
|  3 | Mustermann | Sohnemann | Bachstrasse 13 | NULL  | Irgendwo   | NULL         |
+----+------------+-----------+----------------+-------+------------+--------------+
3 rows in set (0.00 sec)
```

4.7.11. Daten Löschen

Wie Sie einzelne Spalten löschen, haben Sie gerade gesehen. Der Befehl dazu war:

alter table anschriften drop geschlecht;

Die Spalte mit dem Geschlecht ist nun verschwunden.

Einzelne Reihen löschen sie so:

```
mysql> delete from anschriften where id=3;
Query OK, 1 row affected (0.00 sec)

mysql> select * from anschriften;
+----+------------+----------+----------------+-------+------------+---------------+
| id | name       | vorname  | strasse        | plz   | ort        | geburtsdatum |
+----+------------+----------+----------------+-------+------------+---------------+
|  1 | Mustermann | Markus   | Bachstrasse 13 | 11111 | Irgendwo   | NULL          |
|  2 | Musterfrau | Michaela | NULL           | 22222 | Wasserdorf | NULL          |
+----+------------+----------+----------------+-------+------------+---------------+
2 rows in set (0.00 sec)
```

Vergessen Sie nicht das Schlüsselwort where, sonst löschen Sie versehentlich alle Reihen!

Zum Schluss zeige ich Ihnen die Befehle, mit denen Sie die beiden gesamten Tabellen löschen, sowie die ganze Datenbank:

drop table anschriften;
drop table kontostand;

drop database adressen;

Nun sind die ganzen Test-Daten, die Sie angelegt haben, während Sie diese Kapitel durchgearbeitet haben, gelöscht.

4.7.12. SQL Abfragen in Shell Scripten

Sicher wollen Sie die Ergebnisse von SQL Abfragen nicht immer in der tabellarischen Form sehen, wie ich es Ihnen in den vorherigen Kapiteln gezeigt habe. Manchmal möchten Sie Listen in Dateiform erzeugen oder einfach nur einen einzigen Wert abfragen.

Da Sie vermutlich auch die letzten drei Befehle des vorherigen Kapitels brav abgetippt haben, haben Sie jetzt keine Test-Datenbank mehr, an der Ich Ihnen die nötigen Befehle zeigen kann. Legen Sie sich daher bitte jetzt erst einmal eine neue Test-Datenbank an:

```
create database demo;

use demo;

create table tabelle (
   id INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
   wert INTEGER);

insert into tabelle set wert=10;
insert into tabelle set wert=13;
insert into tabelle set wert=19;
insert into tabelle set wert=21;
insert into tabelle set wert=57;
```

Sie haben nun eine kleine Demo-Tabelle angelegt, die so aussieht:

```
mysql> select * from tabelle;
+----+------+
| id | wert |
+----+------+
|  1 |   10 |
|  2 |   13 |
|  3 |   19 |
|  4 |   21 |
|  5 |   57 |
+----+------+
5 rows in set (0.00 sec)
```

Die gleiche select Abfrage können Sie auch im nicht-interaktiven Modus an der Shell-Eingabeaufforderung oder innerhalb von Shell scripten starten. Geben Sie erstmal exit; ein, um den mysql Client zu verlassen. Danach können Sie diesen Befehl eingeben:

```
mysql -u root -p'xxxxx' -D demo -e 'select * from tabelle;'
```

Beachten Sie, dass zwischen -p und dem Passwort (hier xxxxx) kein Leerzeichen stehen darf. Mit -D legen Sie die Datenbank fest (wie beim use Befehl) und mit -e legen Sie die den SQL Befehl fest, der ausgeführt werden soll.

Das Ergebnis wird wie oben als gerahmte Tabelle angezeigt und danach beendet sich der MySQL Client wieder von selbst.

Durch einfügen der Option -B unterdrücken Sie die Rahmen um die Felder herum und mit -N unterdrücken Sie die Spaltenüberschriften:

```
mysql -u root -p'xxxxx' -D demo -N -B -e 'select * from tabelle;'
```

```
1	10
2	13
3	19
4	21
5	57
```

Sie erhalten dann diese einfache schlichte Anzeigeform, die Sie in anderen Programmen (z.B. Microsoft Excel oder KSpread) sehr viel leichter einlesen können. Was Sie hier nicht sehen können, ist, dass die Spalten in der Ausgabe durch Tab-Zeichen getrennt sind. Für Excel und KSpread ist das genau richtig.

Die Ausgabe des Befehles schreiben Sie durch Anhängen von "> Dateiname" in eine Datei, anstatt auf den Bildschirm. Beispiel:

```
mysql -u root -p'xxxxx' -D demo -N -B -e 'select * from tabelle;' > /tmp/test.txt
```

Mit der Option -H sorgen sie dafür, dass der MySQL Client eine Tabelle im HTML Format erzeugt, anstatt einer Textdatei. Die Ausgabe sieht dann so aus:

```
<TABLE BORDER=1>
    <TR>    <TH>id</TH>  <TH>wert</TH>   </TR>
    <TR>    <TD>1</TD>   <TD>10</TD>     </TR>
    <TR>    <TD>2</TD>   <TD>13</TD>     </TR>
    <TR>    <TD>3</TD>   <TD>19</TD>     </TR>
    <TR>    <TD>4</TD>   <TD>21</TD>     </TR>
    <TR>    <TD>5</TD>   <TD>57</TD>     </TR>
</TABLE>
```

Wenn Sie die Option -H mit -N komibinieren, erhalten Sie eine gerahmte Tabelle ohne Spaltenüberschriften.

Wenn Sie einen einzigen Wert innerhalb eines Shell-Scriptes abfragen wollen und diese in eine Variable speichern wollen, können Sie das so tun:

derwert=`mysql -u root -p'xxxxx' -D demo -N -B -e 'select wert from tabelle where id=1;'`
echo Der Wert ist $derwert

Achten Sie auch korrekte Hochkomma-Setzung, denn es gibt drei sehr ähnliche Zeichen! Sie können auf diese Weise nur SQL Befehle ausführen, die einen einzigen Wert liefern, weil Sie sonst alle Werte hintereinander in der Variablen stehen haben, und das nützt Ihnen nichts.

4.7.13. Datensicherung

Dies ist nun das letzte Kapitel über MySQL, dennoch ist es sehr wichtig.

Es gibt viele Methoden, Backups von MySQL Datenbänken anzulegen, von denen ich Ihnen zwei Methoden vorschlagen möchte:

1. Cold Backup

Unter Cold Backup versteht man eine Datensicherung, während die MySQL Server Software gestoppt ist. Das heisst, dass während des Backups überhaupt kein Zugriff auf die Datenbank besteht.

- Stoppen Sie dazu zuerst den MySQL Server.
- Kopieren Sie die Dateien *.frm, *.MYD und *.MYI auf einen anderen Datenträger, z.B. ein Bandlaufwerk.
- Starten Sie den MySQL Server wieder.

2. Hot Backup

Beim Hot Backup sichern Sie einzelne Tabellen im laufenden Betrieb. Leider besteht während dieses Vorganges nur Lesezugriff auf die Tabelle. Andere Programme, die in die Tabelle schreiben wollen, müssen warten.

- Geben Sie im MySQL Client den Befehl **backup table demo.tabelle to "/tmp"** ein. Dadurch wird die Tabelle "tabelle" aus der Datenbank "demo" nach /tmp kopiert.
- Kopieren Sie anschliessend die Dateien /tmp/*.frm und /tmp/*.MYD auf einen anderen Datenträger, z.B. ein Bandlaufwerk.

Mit dem Befehl **restore table demo.tabelle from "/tmp"** spielen Sie ein Backup wieder zurück. Dieser Befehl funktioniert nur, wenn die zu restaurierende Tabelle nicht existiert (ggf. also vorher löschen).

Tips zum Hot Backup

Wenn Sie ein Hot Backup von einer Tabelle machen, kann diese gleichzeitg nur gelesen werden. Schreibzugriffe werden verzögert, bis die gesamte Tabelle in die Backup-Datei geschrieben wurde. Bei großen Tabellen (mehrere Megabytes) und schnellen Anwendungen kann dies zu Performanceproblemen führen.

Stellen Sie sich vor, Sie bieten 24 Stunden Dienstleistungen an, die kostenpflichtig sind und monatlich abgerechnet werden. Sie sammeln dann alle anfallenden Kosten pro Kunde in einer Tabelle - die z.B. so ausieht:

Kunde-Nr	Datum	Artikel	Betrag
10001	01.03.03	SMS	0.49
10001	02.03.03	SMS	0.49
10030	02.03.03	SMS	0.49
10023	03.03.03	SMS	0.49
10001	05.03.03	SMS	0.49
10001	06.03.03	SMS	0.49

Für jede kleine Dienstleistung, die Sie erbringen, fügen Sie eine entsprechende Reihe in diese Tabelle ein. Die Tabelle muss also ständig bereit sein, beschrieben zu werden.

Jetzt können Sie aber kein Backup machen, weil während des Backups keine neue Einträge gespeichert werden können. Und bei sehr großen Tabellen kann das Backup einige Minuten dauern. Für Sie ist es nicht hinnehmbar, dass Sie solange warten müssen, bis Sie die erbrachte Dienstleistung abspeichern können.

Tabellen, die schnell verfügbar sein müssen, sollten Sie daher möglichst klein halten. Kleine Tabellen können Sie unbemerkt mittels Hot Backup sichern. Eine Verzögerung von wenigen Sekunden ist sicher akzeptabel.

Die Lösung für das Problem ist, diese eine Tabelle in zwei aufzuteilen.

Die erste Tabelle speichert alle Rechnungsdaten, die zum laufenden Monat gehören und später auf ein Rechnungsformular ausgedruckt werden. Die zweite Tabelle speichert alle Rechnungsdaten, die schon ausgedruckt wurden.

Im Laufe von mehreren Monaten oder gar Jahren wir Ihre zweite Tabelle dann immer grösser und das Hot Backup dauert dementsprechend immer länger. Das ist aber unkritisch, denn nur die erste Tabelle muss schnell verfügbar sein. Die erste Tabelle speichert nur Daten vom laufenden Monat und daher bleibt sie stets klein.

Und so kopieren Sie den Inhalt von tabelle1 in tabelle2:

insert into tabelle2 select * from tabelle1;

oder

insert into tabelle2 select * from tabelle1 where datum < "2003-04-01" ;

Eine etwas andere Variante des Befehls ist praktisch, wenn die Zieltabelle noch nicht existiert:

create table tabelle2 select * from tabelle1;

oder

create table tabelle2 select * from tabelle1 where datum < "2003-04-01" ;

4.8. Dynamische Webseiten mit PHP

Es gibt viele Scriptsprachen, mit denen man dynamische Webseiten programmieren kann. Eine davon ist PHP, welche ich Ihnen in diesem Kapitel vorstellen möchte.

Ich zeige Ihnen die grundlegenden Befehle und Regeln zum schreiben von dynamischen Webseiten mit PHP. Nach dem Durcharbeiten dieses Kapitels werden Sie imstande sein, eigene Seiten zu Programmieren und Ihre PHP Kenntnisse anhand der Original Dokumentation von PHP zu erweitern.

PHP ist sehr umfangreich, daher sollten Sie auf jeden Fall einen Blick in die Original Dokumentation werden, insbesondere in die Funktionsliste.

4.8.1. Die erste Webseite

In den Merklisten habe ich die genaue Position der Konfigurationsdateien und Webseiten angegeben. Im Folgenden schreibe ich nur noch vom htdocs Verzeichnis oder der httpd.conf Konfigurationsdatei. Schauen Sie gegenenenfalls nochmal in der Merkliste nach, wo sie die Dateien zu suchen haben.

Falls Sie nicht das vorgegebene htdocs Verzeichnis für die Webseiten benutzen wollen, dann legen Sie sich jetzt bitte irgendwo anders ein leeres Verzeichnis an. Öffnen Sie dann die Datei httpd.conf und tragen Sie Ihr neues Verzeichnis dort ein. Relevant sind die folgenden zwei Zeilen:

```
DocumentRoot "/usr/local/apache/htdocs"
<Directory "/usr/local/apache/htdocs">
```

Die Änderungen werden erst nach einem Neustart des Apache Webservers wirksam.

Schreiben Sie nun bitte die folgende Datei, und speichern Sie sie als test.html in das htdocs Verzeichnis ab:

```
<html><body>
Dies ist meine erste HTML Seite.
</body></html>
```

Ein gutes Buch über HTML ist übrigens Selfhtml von Stefan Münz. Dieses Buch gibt es sowohl in gedrucker Form, als auch Online im Internet unter http://www.selfhtml.de.

Prüfen Sie, ob Sie diese Datei in irgendeinem Web-Browser aufrufen können, indem Sie in die URL Zeile http://meinserver/test.html eingeben. Anstelle von meinserver setzen Sie bitte den Namen oder die IP-Adresse Ihres Webservers ein.

Falls nichts erscheint, dann prüfen Sie, ob der Apache Webserver gestartet wurde. Das geht z.B. so:

ps -ef | grep httpd

```
root      1058     1  0 20:13 ?        00:00:00 /usr/local/apache/bin/httpd
nobody    1073  1058  0 20:13 ?        00:00:00 /usr/local/apache/bin/httpd
nobody    1074  1058  0 20:13 ?        00:00:00 /usr/local/apache/bin/httpd
nobody    1075  1058  0 20:13 ?        00:00:00 /usr/local/apache/bin/httpd
nobody    1076  1058  0 20:13 ?        00:00:00 /usr/local/apache/bin/httpd
nobody    1077  1058  0 20:13 ?        00:00:00 /usr/local/apache/bin/httpd
stefan    5896  5865  0 20:28 pts/1    00:00:00 grep httpd
```

Die letzte Zeile mit dem grep Befehl zählt nicht. Sie sehen nun, wieviele Prozesse vom Apache Webserver laufen. Sollten Sie keine sehen, dann wurde der Webserver nicht gestartet.

Zur Problembehebung kann auch ein Blick in die Logfiles helfen. Wo die stehen, habe ich ebenfalls in die Merklisten geschrieben.

Wenn die Fehlermeldung

```
Forbidden
You don't have permission to access /test.html on this server.
```

im Webbrowser erscheint, dann hat das Verzeichnis oder die Datei nicht ausreichend Zugriffsrechte. Alle User (auch fremde) müssen Lese-Recht auf das Verzeichnis, alle übergeordneten Verzeichnisse und die Datei haben.

4.8.2. Die erste dynamische Webseite

Bevor Sie hier weiter machen, sollten Sie sicher stellen, dass Sie die erste Webseite aus dem vorherigen Kapitel problemlos aufrufen können.

Erstellen Sie die Datei test2.php im htdocs Verzeichnis mit folgendem Inhalt:

```
<html><body>
Dies ist meine erste dynamische Webseite mit PHP Code.<p>
<?php
            print("Heute ist der ".strftime("%d.%m.%Y").
                        " und es ist ".strftime("%H:%M:%S")." Uhr.");
?>
</body></html>
```

Machen Sie sich an dieser Stelle vorläufig nicht allzu viele Gedanken über die verwendeten Befehle. Diese lernen Sie in den nächsten Kapiteln kennen.

Rufen Sie diese Datei nun in ihrem Webbrowser auf, indem Sie http://meinserver/test2.php eingeben. Dann erscheint dies:

```
Dies ist meine erste dynamische Webseite mit PHP Code.
Heute ist der 07.03.2003 und es ist 11:25:57 Uhr.
```

PHP Programme werden direkt in die HTML Dateien eingebettet. Die erste Zeile wurde durch ganz normalen HTML Code dargestellt, wie im vorherigen Kapitel. Die zweite Zeile jedoch ist die Ausgabe des PHP Programmes.

Sie kennzeichnen PHP Programme durch zwei Dinge:

- durch die Dateiendung .php anstatt .html
- durch <?php und ?> um den Code herum

Sie können innerhalb einer Datei mehrere PHP Programme schreiben. Diese werden dann von oben nach unten der Reihe nach ausgeführt. Beispiel:

```
<html><body>
Dies ist meine erste dynamische Webseite mit PHP Code.<p>
Heute ist der <?php print(strftime("%d.%m.%Y")); ?>
und es ist <?php print(strftime("%H:%M:%S")); ?>
Uhr.
</body></html>
```

Speichern Sie sich diese Datei unter dem Namen test3.php ab. Sie erzeugt im Webbrowser exakt die gleiche Ausgabe, wie test2.php.

4.8.3. print(), printf() und sprintf()

Im vorherigen Kapitel haben Sie den print() Befehl benutzt, um Ausgaben zu erzeugen.

print("Hallo"); erzeugt die Ausgabe

```
Hallo
```

print("Hallo\nWelt"); erzeugt die Ausgabe

```
Hallo
Welt
```

Anmerkung: Im Webbrowser werden Sie den Zeilumbruch nicht direkt sehen, weil Webbrowser grundsätzlich keine Zeilenumbrüche anzeigen. Lassen Sie sich vom Web-Browser den Seiten-Quelltext anzeigen, dann sehen Sie das Ergebnis wie oben dargestellt.

Die Zeichenfolge \n wird Escape-Sequenz genannt. Sie wird nicht 1:1 ausgegeben, sondern durch einen Zeilenvorschub ersetzt. Weitere gebräuchliche Escape-Sequenzen sind:

- \\ wenn Sie das Zeichen \ selbst ausgeben wollen
- \" wenn Sie doppelte Anführungsstriche ausgeben wollen

Print() gibt einfach nur Text aus. Mit printf() können Sie Werte in formatierter Form ausgeben, was besonders in Kombination mit Variablen nützlich ist.

```
$tag=1;
$monat=4;
$jahr=2003;

printf("Das Datum ist %d.%d.%d ",$tag,$monat,$jahr);
```

```
Das Datum ist 1.4.2003
```

Printf erwartet als ersten Parameter eine Zeichenkette, die ein oder mehrere Platzhalten enthält. Alle weiteren Parameter sind die Werte, die anstelle der Platzhalter ausgegeben werden.

Gebräuchliche Platzhalter sind:

- %d Dezimalzahl
- %i ebenfalls Dezimalzahl
- %f Fliesskommazahl
- %s Zeichenkette
- %x Hexadezimalzahl mit Kleinbuchstaben
- %X Hexadezmalzahl mit großbuchstaben

Wenn Sie das % Zeichen ausgeben wollen, schreiben Sie es einfach doppelt (%%), dann wird es nicht ungewollt als Platzhalter betrachtet.

Wenn Sie dem Platzhalter eine Zahl voranstellen, die mit 0 beginnt, wird der Wert entsprechend breit rechtsbündig ausgegeben. Zu große Zahlen werden in voller Länge ausgegeben.

```
$tag=1;
$monat=4;
$jahr=2003;

printf("Das Datum ist %02d.%02d.04%d ",$tag,$monat,$jahr);
```

```
Das Datum ist 01.04.2003
```

Bei Fliesskommazahlen geben Sie die Länge für die Vor- und Nachkommastellen an:

```
$zahl1=138.45865;
$zahl2=1.2;
printf("Die Zahl1 ist %02.2f <br>",$zahl1);
printf("Die Zahl2 ist %02.2f ",$zahl2);
```

```
Die Zahl1 ist 138.45
Die Zahl2 ist 01.20
```

Vor dem Dezimalpunkt werden immer alle Ziffern ausgegeben, auch wenn es zuviele sind. Aber dahinter wird die Zahl mathematisch gerundet, so dass sie genau so lang ist, wie Sie beim Platzhalter festgelegt haben.

In diesem Beispiel habe ich die Zeichenfolge <br> benutzt, um einem sichtbaren Zeilenumbruch im Webbrowser zu erzeugen.

Der Befehl sprintf() funktioniert wie printf, nur schreibt er seine Ausgabe in eine Variable anstatt auf den Bildschirm.

$zahl=123;
$ergebnis=sprintf("Die Zahl ist %d",$zahl);

printf($ergebnis);

```
Die Zahl1 ist 123
```

4.8.4. Variablen und Datentypen

Variablen haben Sie bereits im Vorigen Kapitel gesehen. Variablen speichern:

- Boolean Werte
- Integer Zahlen
- Fliesskomma Zahlen
- Zeichenketten
- Arrays
- NULL

Variablen können auch Objekte und Referenzen speichern, auf die ich in diesem Buch aber nicht eingehe. Im folgenden erkläre ich die obigen Variablentypen.

Boolean Werte

Boolean Werte können nur TRUE oder FALSE sein. Sie dürfen diese Schlüsselworte auch klein schreiben:

```
$wert1 = true;
$wert2 = false;
```

Integer Zahlen

Integer Zahlen können Sie dezimal, oktal und hexadezimal schreiben. Oktakzahlen beginnen mit einer 0, Hexadezimalzahlen beginnen mit 0x. Die folgenden Beispiele haben alle den gleichen Wert:

```
$zahl1 = 123;
$zahl2 = 0173;
$zahl3 = 0x7B;
```

Fliesskomma Zahlen

Fliesskommazahlen schreiben Sie mit Dezimalpunkt oder in Exponentialschreibweise:

```
$zahl1 = 1234.56;
$zahl2 = 12.3456e2;
$zahl3 = 0.345;
$zahl4 = 345e-3;
```

Zeichenketten

Zeichenketten müssen Sie in einfache oder doppelte Anführungsstriche einschliessen:

```
$text1 = 'Hallo, das ist Stefan\'s Text';
$text2 = "Der erste text war: $text1"
```

Der Unterschied zwischen den beiden Versionen ist, dass Sie innerhalb von doppelten Anführungsstrichen auch Variablen einfügen können. Bei einfachen Anführungsstrichen geht das nicht.

Wenn Sie im Text selbst Anführungsstriche schreiben wollen (siehe erstes Beispiel), müsen sie ein Backslash (\) voranstellen, weil PHP das Zeichen sonst als Zeichenketten-Ende betrachten würde.

Arrays

Arrays sind Listen von mehreren Werten, denen Schlüssel (Indexe) zugeordnet sind. Beispiel:

```
$name[1] = "Miriam";
$name[2] = "Rolf";
$name[3] = "Mucki";
```

Die Schlüssel dürfen auch 0 und negativ sein. Sie dürfen ebenfalls Zeichenketten als Schlüssel verwenden:

```
$MutterVon["Miriam"] = "Petra";
$MutterVon["Rolf"] = "Heidi";
$MutterVon["Mucki"] = "Micki";
```

NULL

NULL ist ein spezieller Wert für Variablen, denen noch kein Wert zugeordnet wurde oder deren Wert mit unset() gelöscht wurde. Sie kennen das bereits von MySQL.

Wie Sie an den obigen Beispielen gesehen haben, beginnen Variablennamen immer mit einem Dollarzeichen, gefolgt von einem Buchstaben. Nach dem ersten ersten Buchstaben dürfen Sie auch Ziffern verwenden. Die groß/Kleinschreibung ist wichtig. Die $TEST ist eine andere Variable, als $test und $Test.

4.8.5. Operatoren

Operatoren benutzen Sie zum Rechnen oder zum Vergleichen. Wenn Sie Operanden mit einem Operator verbinden, entsteht ein Ausdruck. Jeder Ausdruck liefert eine Zahl, eine Zeichenkette oder einen Boolean-Wert als Ergebnis.

Arithmetische Operatoren

+	Addition **1 + 2** ergibt 3
-	Subtraktion **5 - 2** ergibt 3
*	Multiplikation **2 * 3** ergibt 6
/	Division **36 / 3** ergibt 12
%	Ist der Modulus Operator, mit anderen Worten, der Rest einer Division **12 % 5** ergibt 2

Bit Operatoren

&	Und. Nur die Bits, die in beiden Operanden gesetzt sind, sind im Ergebnis gesetzt. **11 & 7** ergibt 6
|	Oder. Im Ergebnis sind alle Bits gesetzt, die in einem oder beiden Operanden gesetzt sind. **4 | 2** ergibt 6
^	Exklusiv-Oder. Im Ergebnis sind alle Bits gesetzt, die in nur einem der Operanden gesetzt sind. **6 ^ 4** ergibt 2
<<	Nach links schieben. Im Ergebnis sind alle Bits um n Positionen nach links verschoben. **4 << 1** ergibt 8
>>	Nach rechts schieben. Im Ergebnis sind alle Bits um n Positionen nach rechts verschoben. **4 >> 1** ergibt 2

Vergleichsoperatoren

==	Gleich **3 == 3** ergibt TRUE
!=	Ungleich **3 != 4** ergibt TRUE
<>	Ungleich (wie !=) **3 != 4** ergibt TRUE
<	Kleiner als **3 < 4** ergibt TRUE
>	Grösser als **4 > 3** ergibt TRUE
<=	Kleiner oder gleich **3 < 3** ergibt TRUE
>=	Grösser oder gleich **4 > 4** ergibt TRUE

Logische Operatoren

and	Und **TRUE and TRUE** ergibt TRUE
&&	Und **TRUE && TRUE** ergibt TRUE

or	Oder **TRUE or FALSE** ergibt TRUE
||	Oder **TRUE || FALSE** ergibt TRUE
xor	Exklusiv-Oder **TRUE xor TRUE** ergibt FALSE **TRUE xor FALSE** ergibt TRUE
!	Negation **! TRUE** ergibt FALSE

Der Zuweisungsoperator ist das Gleicheitszeichen. Sie benutzen ihn, um einer Variablen ein Wert zuzuweisen. Beispiel:

$ergebnis = 34 + 2;

Für die Arithmetischen Operatoren und Bit Operatoren gibt es eine Kombination mit Zuweisung nach diesen Schema:

$a = 3;

$a += 4; ergibt 7

Die Variable, der Sie das Ergebnis zuweisen ist zugleich der linke Operator des Operanden.

Sie können externe Programme ausführen und deren Textausgabe einer Variablen zuweisen:

$ergebnis = `ls -l` ;

Der ls Befehl listet das Inhaltsverzeichnis des aktuellen Verzeichnisses auf. Diese Liste weisen Sie der Variablen $ergebnis zu. Achten Sie auf die richtige Zeichensetzung, hier müssen Sie die "Backquotes" verwenden, nicht die einfachen Anführungsstriche (Quotes)!

Zeichenketten können mit dem Punkt verkettet werden:

"Hallo "."Maxi" ergibt "Hallo Maxi"

4.8.6. Kontrollstrukturen

Programme werden normalerweise der Reihe nach von oben nach unten ausgeführt. Mit Kontrollstrukturen durchbrechen Sie diese Monotonie.

Kontrollstrukturen ermöglichen die wiederholte Ausführung eines Programmabschnittes oder die bedingte Ausführung eines Abschnittes.

Programmabschnitte sind Befehle, die Sie in geschweifte Klammern einschliessen und so als zusammengehörigen Block kennzeichnen. Beispiele dazu finden Sie in den folgenden Unterkapiteln.

4.8.6.1. if...elseif...else

Durch if,elseif und else führen Sie einen Programmabschnitt bedingt aus. Hinter if müssen Sie einen Audruck in Klammern schreiben. Der Programmabschnitt wird nur ausgeführt, wenn der Ausdruck wahr ist (TRUE oder nicht Null):

```
if ($a > 3)
{
   Befehle;
   Befehle;
   Befehle;
}
```

Mit else können Sie einen alternativen Programmabschnitt anhängen, der ausgeführt wird, wenn der Ausdruck FALSE oder 0 ist:

```
if ($a >3)
{
   Befehle;
   Befehle;
   Befehle;
}
else
{
   Alternative Befehle;
   Alternative Befehle;
   Alternative Befehle;
}
```

Mit elseif können Sie mehrere alternative Bedingungen verketten:

```
if ($a < 0)
{
   print("$a ist negativ.");
}
elseif ($a > 0)
{
   print("$a ist positiv.");
}
else
{
   print("$a ist weder negativ, noch positiv, also Null.");
}
```

Sie müssen nicht so viele Zeilenumbrüche verwenden, wie ich das im obigen Beispiel getan habe. So ist es auch korrekt:

```
if ($a < 0) {
   print("$a ist negativ.");
} elseif ($a > 0) {
   print("$a ist positiv.");
} else {
  print("$a ist weder negativ, noch positiv, also Null.");
}
```

Wenn Sie möchten, können Sie alles sogar in eine Zeile schreiben - übersichtlich wäre das dann aber nicht mehr.

4.8.6.2. switch...case

Wenn Sie in einer if Anweisung viele Alternativen abfragen wollen, wird der Programmcode schnell unübersichtlich. Ein schlechtes Beispiel wär dies:

```
if ($name=="Maria")
{
    Befehle_fuer_Maria;
}
elseif ($name=="Josef")
{
    Befehle_fuer_Josef;
}
elseif ($name=="Martin")
{
    Befehle_fuer Martin;
}
elseif ($name=="Kerstin")
{
    Befehle_fuer_Kerstin;
}
else
{
    print("Unbekannter Name!");
}
```

PHP gibt Ihnen die Möglichkeit, solche langen if Anweisungen anders auszudrücken. Vielleicht gefällt Ihnen die switch...case Anweisung besser:

```
switch ($name):
    case "Maria":
        Befehle_fuer_Maria;
        break;
    case "Maria":
        Befehle_fuer_Josef;
        break;
    case "Maria":
        Befehle_fuer_Martin;
        break;
    case "Maria":
        Befehle_fuer_Kerstin;
        break;
    default:
        printf("Unbekannter Name!")
endswitch
```

Vergessen Sie die break Befehle nicht, sonst werden die nachfolgenden Befehle ungewollt ausgeführt. Die switch...case Anweisung ist eine Alternative zur if Anweisung. Entscheiden Sie selbst, wann Sie welche Variante bevorzugen.

4.8.6.3. while und do...while

Mit while führen Sie einen Programmabschnitt mehrmals aus, und zwar solange eine Bedingung erfüllt ist.

Hinter while müssen Sie einen Ausdruck in Klammern schreiben. Der Programmabschnitt wird solange wiederholt, wie der Ausdruck wahr ist (TRUE oder nicht Null). Dabei wird vor jeder Wiederholung geprüft, ob der Ausdruck wahr ist.

```
while ($a < 3)
{
   Befehle;
   Befehle;
   Befehle;
}
```

Eine andere Variante der while Schleife ist die do...while Schleife. Auch hier wird der Programmabschnitt wiederholt, solange der Ausdruck wahr ist. Aber dieses mal wird erst nach jeden Durchlauf geprüft, ob der Ausdruck noch wahr ist.

```
do
{
   Befehle;
   Befehle;
   Befehle;
}
while ($a < 3);
```

4.8.6.4. for

Die for Schleife dient dem Zweck, einen Programmabschnitt mehrmals zu wiederholen und zugleich die Wiederholungen zu zählen.

```
for ($i=1; i<=6; $i=$i+1)
{
   print("i ist jetzt $i <br>");
}
```

Diese Schleife erzeugt die Bildschrimausgabe:

```
i ist jetzt 1
i ist jetzt 2
i ist jetzt 3
i ist jetzt 4
i ist jetzt 5
i ist jetzt 6
```

Die Schleife wurde 6 mal ausgeführt. Dabei wurde der Schleifenzähler von 1 bis 6 hochgezählt.

Hinter for müssen Sie drei Ausdrücke angeben, die gemeinsam in Klammern eingeschlossen sind und durch Semikolons getrennt sind:

```
for (Initialisierung; Ausdruck; Veraenderung)
```

Der Initialisierungs-Befehl weist der Zählvariablen einen Startwert zu.

Der Ausdruck prüft vor jeder Wiederholung, ob die Schleife nochmal ausgeführt werden soll. Wenn der Ausdruck nicht mehr wahr ist, wird die Schleife nicht nocheinmal wiederholt.

Der Veränderungs-Befehl dient dem Zweck, die Zählvariable NACH jedem Schleifendurchlauf zu verändern. Meistens will man bei jeder Schleifenwiederholung 1 addieren.

Die for Schleife ist genaugenommen nur eine andere Schreibweise für die while Schleife.

```
for ($i=1; i<=6; $i=$i+1)
{
   Befehle;
}
```

ist identisch mit:

```
$i=1;
while ($i<=6)
{
   Befehle;
   $i=$i+1;
}
```

4.8.6.5. foreach

Die foreach Schleife dient dazu, mit jedem Element eines Arrays etwas zu tun:

```
$namen[1]="Mary";
$namen[2]="Hans";
$namen[3]="Maxi";
$namen[4]="Hannes";

foreach ($namen as $n) {
   print("Der Name ist $n <br>");
}
```

Dieses Beispiel erzeugt die Bildschirmausgabe:

```
Der Name ist Mary
Der Name ist Hans
Der Name ist Maxi
Der Name ist Hannes
```

Zu foreach gibt es noch die eine erweiterte Version, bei der Sie auch den Schlüssel abfragen können:

```
$monate["Januar"]="kalt";
$monate["April"]="wechselhaft";
$monate["Juli"]="warm";


foreach ($monate as $schluessel => $wert) {
   print("Der $schluessel ist $wert <br>");
}
```

Dieses Beispiel erzeugt die Bildschirmausgabe:

```
Der Januar ist kalt
Der April ist wechselhaft
Der Juli ist warm
```

4.8.6.6. break, continue und exit

Eine Schleife können Sie ausnahmsweise abbrechen, indem Sie den break Befehl benutzen:

```
$birnen=0;

while ($birnen < 1000)
{
   print("Wir nehmen noch eine Birne dazu <br>");

  if ($birne=12)
   {
      $faul=true;
   }

  if ($faul == true)
   {
      break;
   }

  $birnen=$birnen+1;
}

print("Wir haben jetzt entweder 1000 Birnen oder wegen einer faulen abgebrochen.");
```

Diese Schleife addiert immer mehr Birnen, bis es die 1000 erreicht hat. Wenn aber eine faule Birne erkannt wird, bricht die Schleife ab. Um dies möglichst einfach zu demonstrieren, wird die 12. Birne immer als Faul gekennzeichnet.

Der Continue Befehl erzwingt die nächte Schleifen-Wiederholung, auch wenn der Programmabschnitt innerhalb der Schleife noch nicht zu Ende ausgeführt wurde:

```
$birnen=0;
$versuche=0;
```

```
while ($versuche < 1000)
{
   $versuche=$versuche+1;

  if ($faul == true)
   {
      print("Die Birne Nr. $versuche ist faul <br>");
      continue;

  }

  print("Die Birne Nr. $versuche nehme ich nun, hmm schmeckt lecker.<br>");
   $birnen=$birnen+1;
}

print("Wir haben jetzt 1000 Birnen oder einige faule Früchte übersprungen.<br>");
print("$birnen Früchte waren gut.");
```

Diese Schleife versucht 1000 Birnen zu nehmen. Wenn eine faule Frucht erkannt wurde, wird diese jedoch übersprungen. Die verbleibenden Befehle

```
  print("Die Birne Nr. $versuche nehme ich nun, hmm schmeckt lecker.<br>");
   $birnen=$birnen+1;
```

werden bei faulen Früchten nicht ausgeführt, weil der Continue Befehl den nächsten Schleifendurchlauf erzwingt. Die Variable $versuche wird kontinuierlich um eins erhöht, während die Variable $birnen nur bei guten Birnen erhöht wird.

Continue und Break im Vergleich:

Der Break Befehl unterbricht eine Schleife. Als nächstes werden die Befehle ausgeführt, die hinter der Schleife im Programm stehen.

Der Continue Befehl unterbricht die Befehle innerhalb einer Schleife und erzwingt den nächsten Schleifendurchlauf.

Break und continue können Sie auch bei do...while und for anwenden.

Es gibt noch einen Abbruch-Befehl, er lautet exit. Exit unterbricht das gesamte PHP Programm mit sofortiger Wirkung.

4.8.7. include

Mit der include Anweisung laden Sie eine andere Datei zum Porgrammcode hinzu. Beispiel:

Die Datei teil1.include hat diesen Inhalt:

```
$a=2;
$b=3;
```

Die Datei test4.php hat diesen Inhalt:

```
<html><body>
<?php
    include 'teil1.include';
    print("Die Variable a ist $a und die Variable b ist $b");
?>
</body></html>
```

Dies ist sehr nützlich. Sie können so Sammlungen von Variablen und Funktionen anlegen, die Sie in mehreren php Dateien gemeinsam nutzen.

4.8.8. Funktionen

Funktionen sind kleine Programmabschnitte, die sie später wie PHP Befehle aufrufen können. Sie dienen hauptsächlich dem Zweck, große Programme in kleinere Abschnitte (Funktionen) zu zerlegen, um sie so übersichtlicher zu gestalten. Beispiel:

```
function berechne($wert1, $wert2)
{
   $ergebnis = $wert1 + $wert2 * 24;
   return $ergebnis;
}

$super = berechne(3,4);
print("Die Berechnung von 3 und 4 ergibt $super <br>");

$klasse = berechne(9,12);
print("Die Berechnung von 9 und 12 ergibt $klasse <br>");

print("Oder so " . berechne(3,2) . "<br>");

printf("Oder so %i <br>", berechne(1,2));
```

Im obigen Beispiel wird die Funktion berechne() vier mal benutzt, um zwei Zahlen miteinander zu verrechnen. Die Funktion liefert ihr Ergebnis mittel return Befehl an das Hauptprogramm ab.

```
Die Berechnung von 3 und 4 ergibt 99
Die Berechnung von 9 und 12 ergibt 297
Oder so 51
Oder so 49
```

Ein anderes Beispiel, diesmal mit einer Funktion, die etwas tut, aber kein Ergebnis zurückgibt:

```
function fehlermeldung($text)
{
   print("Es ist ein Fehler aufgetreten: <br>");
   print("$text <br>");
   print("Das Programm wird jetzt abgebrochen <br>");
   exit;
 }

Befehle;
Befehle;
Befehle;
fehlermeldung("Die Festplatte ist voll");
Befehle;
Befehle;
Befehle;
fehlermeldung("Text nicht gefunden");
Befehle;
Befehle;
Befehle;
fehlermeldung("Falsche Eingabe, 0 ist nicht erlaubt.");
```

Hier benutzen wir eine Funktion mehrmals, um eine Fehlermeldung anzuzeigen. Das Hauptprogramm bleibt so übersichtlich und man muss den selben Programmcode nicht mehrmals schreiben. Die Ausgabe der ersten Fehlermeldung würde so aussehen:

```
Es ist ein Fehler aufgetreten:
Die Festplatte ist voll
Das Programm wird jetzt abgebrochen
```

4.8.9. Formular Variablen und Datei-Uploads

Webseiten können Formulare enthalten. Wenn der Betrachter ein Formular ausgefüllt hat, und auf dem "Submit" Button klickt, werden die ausgefüllten Felder an den Webserver geschickt.

In PHP Programmen finden Sie die Inhalte der Formularfelder in den Arrays wieder.

- $_POST
- $_GET
- $_FILES
- $_COOKIE

Die Gestaltung des Formulars bestimmt, in welchem Array Sie die Variablen wiederfinden. Angenommen, Sie haben ein solches Formular (test5.html):

```
<html><body>
    <form action="test5.php" method="post">
        Name: <input type="text" name="username"><br>
        <input type="submit">
    </form>
</body></html>
```

Im Webbrowser erscheint dieses Formular als ein Eingabefeld mit einem "Submit" Button.

Die Eingabe wird an das PHP Programm test5.php übergeben, und zwar als Variable $_POST["username"].

Sie könnten z.B. so darauf zugreifen (test5.php):

```
<html><body>
<?php
    printf("Sie haben %s eingegeben. Danke", $_POST["username"]);
?>
</body></html>
```

Sie können Formulardaten auch mit der get-Methode senden. Der Unterschied ist, dass sie dann in der Adress-Leiste des Webbrowsers erscheinen, etwa so:

```
http://localhost/test5.php?username=Stefan
```

File Upload funktionieren so (test6.html):

```
<html><body>
<form method="post" enctype="multipart/form-data" action="test6.php">
    <INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000">
    <input type="file" name="userfile">
        <input type="submit">
    </form>
</body></html>
```

Der Webbrowser prüft, ob die zu sendende Datei zu groß ist (anhand der MAX_FILE_SIZE Angabe) und verweigert ggf. das Senden.

$_FILES["userfile"]["name"]	enthält den urpsrünglichen Dateinamen
$_FILES["userfile"]["size"]	enthält die Dateigrösse in Bytes
$_FILES["userfile"]["tmp_name"]	enthält den temporären Dateinamen, unter dem die Datei auf dem Webserver gespeichert wurde
$_FILES["userfile"]["error"]	enthält den Fehlercode der Dateiübertragung. 0=kein Fehler.

Verlassen Sie sich nicht darauf, dass der Webbrowser die maximale Dateigrösse korrekt berücksichtigt! Fehlerhafte Browser könnten auch zu große Dateien abliefern, ihr Programm muss damit klarkommen und wenigstens eine ordentliche Fehlermeldung anzeigen.

Ein Beispiel, wie Sie upgeloadete Dateien empfangen können (test6.php):

```
<html><body>
<?php
   if (is_uploaded_file($_FILES["userfile"]["tmp_name"]))
    {
        copy($_FILES["userfile"]["tmp_name"],
             "/dokumente/upgeloadete.jpg");
        print("Danke, die Datei wurde gespeichert");
    }
    else
    {
        print("Sie haben die Datei nicht wirklich gesendet!");
    }
?>
</body></html>
```

Die Funktion is_uploaded_file prüft, ob die Datei auch wirklich gesendet wurde. Benutzen Sie diese Kontrolle immer, um Hackerangriffe zu verhindern. Ohne diese Kontrolle könnte ein Hacker Ihnen gefälschte Dateinamen senden, die Ihr Programm in die Irre führen.

Die Funktion copy kopiert die temporäre Datei in eine endgültige Datei um. An dieser Stelle könnten Sie natürlich auch andere Aktionen mit der Datei durchführen.

Wenn Sie die Datei später noch brauchen, müssen Sie sie auf jeden Fall umkopieren, denn der Webserver löscht die temporäre Datei unmittelbar nach Beendigung Ihres PHP Programmes selbstständig.

Wie Sie Cookies anwenden, können Sie gerne in dem originalen PHP Handbuch nachlesen. Ich rate dringend davon ab, Cookies zu verwenden, weil immer mehr Internet-User diese Funktion im Webbrowser sperren und sie somit nicht sicher zur Verfügung steht.

4.8.10. SQL Abfragen

SQL Abfragen lassen sich in PHP Programmen sehr leicht durchführen. Das folgende Beispiel zeigt Ihnen, wie Sie einen select-Befehl ausführen, und das Ergebnis als HTML Tabelle anzeigen (test7.php):

```
<html><body>
Dies ist das Ergebnis einer SQL Abfrage: <br>
<?php
mysql_connect("localhost","root","mein_passwort");
$result=mysql_db_query("datenbank","select * from tabelle")
if ($result)
{
  $reihen = mysql_num_rows($result);
  $spalten = mysql_num_fields($result);
  print("<table border=1>\n");
  print("  <tr>\n");
  for ($i=0; $i<$spalten; $i++)
  {
    $name=mysql_field_name($result,$i);
    printf("    <th> %s </th>\n",$name);
  }
  print("  </tr>\n");
  for ($j=0; $j<$reihen; $j++)
  {
    print("  <tr>\n");
    $reihe=mysql_fetch_row($result);
    for ($i=0; $i<$spalten; $i++)
    {
      printf("    <td> %s </td>\n",$reihe[$i]);
    }
    print("  </tr>\n");
  }
  print("</table>\n");
}
else
{
  printf("Es ist ein Fehler aufgetreten: <br> %s",mysql_error());
}
?>
</body></html>
```

Nun möchte ich Ihnen dieses Programm Zeile für Zeile erklären:

<html><body>
**Dies ist das Ergebnis einer SQL Abfrage:
**

Dies ist der beginn des HTML Codes. Zuerst geben wir einen festen Text als Überschrift der Tabelle aus, gefolgt von einem Zeilenumbruch.

```
<?php
```

So wird der Beginn des PHP Programmcodes gekennzeichnet.

```
mysql_connect("localhost","root","mein_passwort");
```

Wir verbinden uns mit dem SQL Server auf dem eigenen Rechner (localhost) und melden uns mit dem Usernamen root und dem Passwort mein_passwort an. Ändern Sie den Servernamen und das Passwort ggf. entsprechend Ihrer Einstellung.

```
$result=mysql_db_query("datenbank","select * from tabelle")
```

Mit mysql_db_query schicken wir einen SQL befehl zum SQL Server und fragen ab, ob die Abfrage erfolgreich war. Die Variable $result erhält bei Erfolg eine Kennzahl, die man später braucht, um das Ergebnis auszulesen. Wenn der Befehl fehlerhaft war, ist die Variable NULL.

```
if ($result)
{
  ...
}
else
{
  printf("Es ist ein Fehler aufgetreten: <br> %s",mysql_error());
}
```

Mit der if Kontrollstrukut wird geprüft, ob $result einen Wert enthält. Wenn $result NULL ist, wird der else-Abschnitt ausgeführt, der eine Fehlermeldung anzeigt. Beachten Sie hier die Verwendung der Funktion mysql_error(). Sie liefert den Fehlertext zum gerade fehlgeschlagenen SQL Befehl.

Anstelle von **if ($result)** könnte man auch **if ($result != 0)** schreiben. Beide Versionen erfüllen den gleichen Zweck.

Der Programmabschnitt hinter **if** wird nur dann ausgeführt, wenn der Ausdruck TRUE oder nicht NULL ist. Auch die schlichte Angabe eines Variablennamens ist bereits ein gültiger Ausdruck.

```
$reihen = mysql_num_rows($result);
$spalten = mysql_num_fields($result);
```

Mit diesen beiden Funktionen ermitteln wir die Anzahl der Reihen und Spalten des Ergebnisses. Nur select Befehle liefern ein tabellarisches Ergebnis. Alle anderen SQL Befehle sind einfach nur Erfolgreich oder nicht erfolgreich.

```
for ($i=0; $i<$spalten; $i++)
{
  $name=mysql_field_name($result,$i);
  printf("    <th> %s </th>\n",$name);
}
```

Diese Programmschleife soll die Namen aller Spalten als Tabellenüberschrift ausgeben. Dabei ist es wichtig, zu wissen, dass die Erste Spalte die Nummer 0 hat. Die for Schleife beginnt daher mit 0 zu zählen. Mit der Funktion mysql_field_name() ermitteln wir die Namen jeder Spalte und geben diesen mit einem printf() Befehl aus.

```
for ($j=0; $j<$reihen; $j++)
{
  print("  <tr>\n");
  $reihe=mysql_fetch_row($result);
  for ($i=0; $i<$spalten; $i++)
  {
    printf("    <td> %s </td>\n",$reihe[$i]);
  }
  print("  </tr>\n");
}
```

Hier haben wir zwei for Schleifen ineinander verschachtelt, denn wir wollen nun alle Reihen der Tabelle auslesen und in jeder Reihe alle Spalten auslesen. Auch bei den Reihen beginnt die Nummerierung mit 0, daher beginnen hier wieder beide for Schleifen mit 0 zu zählen. Mit print() und printf() Befehlen, werden die Werte ausgegeben.

Die HTML-Ausgabe, die von diesem PHP Programm erzeugt wird, sieht z.B. so aus:

```
<html><body>
Dies ist das Ergebnis einer SQL Abfrage: <br>
<table border=1>
  <tr>
    <th> a </th>
    <th> b </th>
    <th> c </th>
  </tr>
  <tr>
    <td> 1 </td>
    <td> 2 </td>
    <td> 3 </td>
  </tr>
  <tr>
    <td> 4 </td>
    <td> 5 </td>
    <td> 6 </td>
  </tr>
</table>
</body></html>
```

Diese Tabelle hat drei Spalten mit den Namen a, b und c, sowie zwei Reihen mit den Zahlen 1 bis 6, wiel dies die Werte aus meiner Tabelle waren. Wenn Sie eine andere Tabelle verwenden, werden Sie auch andere Spaltennamen und Werte sehen.

Machen Sie sich mit diesem Beispiel gut vertraut. Spielen Sie ein wenig herum, indem Sie das Ausgabeformat ändern, oder weitere Informationen, anzeigen. Sie könnten z.B. die Anzahl der Reihen und Spalten im Webbroser anzeigen. Versuchen Sie, das Ergebnis anders Formatiert anzuzeigen, also nicht als HTML Tabelle. Erstellen Sie sich ein HTML Formular, in dass Sie jeden beliebigen SQL Befehl eingeben können um anschliessen das Ergebnis zu sehen. Sie müssen dann anstelle des festen SQL Befehls die Variable $_POST["Feldname"] verwenden.

Als weitere Übung könnten Sie sich einen Satz HTML Formulare und PHP Scripte bauen, mit denen Sie eine Adress-Datenbank eingeben und anzeigen können.

Für die Erstellung dynamischer Webseiten ist es sehr wichtig, SQL Abfragen und Formulare zu beherrschen.

4.8.11. Lokale und globale Variablen

Variablen werden nicht nur durch ihren Namen bezeichnet, sondern auch durch ihren Gültigkeitsbereich. Es gibt zwei Gültigkeitsbereiche:

- lokale Variablen
- globale Variablen

Alle Variablen, die Sie innerhalb einer Funktion benutzen, sind lokal. Variablen, die Sie ausserhalb einer Funktion benutzen, sind global. Ein Beispiel (test8.php):

```
<html><body>
<?php
    function test1()
    {
       $a = "Zwei"
       print("Der lokale Wert in test1 ist $a <br>");
    }

   function test2()
    {
       print("Der lokale Wert in test2 ist $a <br>");
    }
   $a = "Eins";
    print("Der globale Wert ist $a <br>");
    test1();
    test2();
    print("Der globale Wert ist immer noch $a <br>");
?>
</body></html>
```

Speichern Sie sich dieses Programm ab und probieren Sie es aus. Die Bildschirmausgabe wird so aussehen:

```
Der globale Wert ist Eins
Der lokale Wert in test1 ist Zwei
Der lokale Wert in test2 ist
Der globale Wert ist immer noch Eins
```

Interessant sind hier besonders die dritte und vierte Ausgabezeile. Bei der dritten Ausgabe fehlt der Wert, denn innerhalb der Funktion test2 wurde der Variablen kein Wert zugewiesen.

Es spielt keine Rolle, dass es eine gleichnamige Variable bereits global gibt, sowie lokal in der anderen Funktion. Innerhalb von test2 gibt es $a nicht.

Die vierte Ausgabezeile demonstriert Ihnen, dass die globale Variable $a immer noch unverändert ist. Die Zuweisung $a="Zwei" in der Funktion test1 hat sich also überhaupt nicht auf die globale Variable ausgewirkt.

Sie sehen, dass es hier also zwei Variablen mit dem selben Namen $a gibt. Beide haben nichts miteinander zu tun.

Sie haben im vorherigen Kapitel bereits gesehen, wie Sie Variablen an eine Funktion übergeben, so dass sie auch dort bekannt ist. Sie müssen die zu übergebenden Variablen hinter den Namen der Funktion in Klammern schreiben, also so (test9.php):

```
<html><body>
<?php
    function test($a)
    {
       print("Der Wert in test ist $a <br>");
       $wert="Zwei"
       print("Der neue Wert in test ist $a <br>");
    }
```

```
   $a = "Eins";
    print("Der globale Wert ist $a <br>");
    test($a);
    printf( "Der globale Wert ist immer noch $a <br>");
?>
</body></html>
```

Dieses Programm erzeugt die Ausgabe:

```
Der globale Wert ist Eins
Der Wert in test ist Eins
Der neue Wert in test ist Zwei
Der globale Wert ist immer noch Eins
```

Der Inhalt der Variable $a wird hier als Argument an die Funktion übergeben und ist daher jetzt auch innerhalb der Funktion bekannt.

Wie sie sehen, wird wirklich nur der Inhalt übergeben, und nicht die Variable selbst. Denn die Funktion hat der Variablen den neuen Wert "Zwei" zugewiesen, doch davon hat das Hauptprogramm nichts mitbekommen.

Variablen, die Sie als Argument einer Funktion übergeben, können dort also verwendet werden, doch sind Änderung durch die Funktion im Hauptprogrogramm nicht sichtbar!

Schauen Sie sich jetzt bitte das nächste Beispiel an (test10.php):

```
<html><body>
<?php
    function test($wert)
    {
       print("Der Wert in test ist $wert <br>");
       $wert="Zwei"
       print("Der neue Wert in test ist $wert <br>");
    }
    $a = "Eins";
    print("Der globale Wert ist $a <br>");
    test($a);
    print("Der globale Wert ist immer noch $a <br>");
?>
</body></html>
```

Dieses Beispiel erzeigt exakt die gleiche Bildschrimausgabe, wie das vorherige Programm. Wie sie sehen, muss die Variable innerhalb der Funktion nicht genau so heissen, wie im Hauptprogramm.

Der Grund: Es wird nur der Wert der Variablen übergeben, und nicht die Variable selbst - kommt Ihnen das bekannt vor? Merken Sie sich, dass Funktionsargumente nicht genau so heissen müssen, wie im Hauptprogramm.

Was Ihnen jetzt noch fehlt, ist eine Möglichkeit, innerhalb einer Funktion auf die globalen Variablen zuzugreifen. Erfahrene Programmierer vermeiden die Nutzung dieser Möglichkeit, denn sie erschwert das Kopieren alter Funktionen in neue Programme. Das folgende Beispiel (test11.php) demonstriert die Verwendung globaler Variablen:

```
<html><body>
<?php
    function test()
    {
       global $a, $b;
       print("Die Werte in test sind $a und $b <br>");
       $a="Drei"
    }
   $a = "Eins";
    $b = "Zwei";
    print("Die globalen Werte sind $a und $b <br>");
    test();
    print("Die globalen Werte sind jetzt $a und $b <br>");
?>
</body></html>
```

Dieses Programm erzeugt die Ausgabe:

```
Die globalen Werte sind Eins und Zwei
Die Werte in test sind Eins und Zwei
Die globalen Werte sind jetzt Drei und Zwei
```

Durch den Befehl "global" machen Sie der Funktion die angegeben Variablen bekannt. Sie sind nun also sowohl global als auch lokal verwendbar. Das beweist die zweite Zeile der Bildschirmausgabe.

Die dritte Zeile beweist, dass jetzt auch Änderungen innerhalb der Funktion im Hauptprogramm sichtbar sind.

4.8.12. Variable Variablen

Der Titel dieses Kapitel hört sich wie ein Witz an, aber das ist nicht der Fall. Variable Variablen werden selten genutzt. Doch ab und zu kann man ein Programm durch Ihre Benutzung erheblich vereinfachen. Ein Beispiel (test12.php):

```
<html><body>
<?php
    $Januar = "kalt";
    $Juli = "warm";
    print("Im Januar ist es $Januar <br>");
    print("Im Juli ist es $Juli <br>");
    $Monat="Juli"
    print("Im Monat $Monat ist es $$Monat <br>");
?>
</body></html>
```

Das Programm erzeugt die Bildschirmausgabe:

```
Im Januar ist es kalt
Im Juli ist es warm
Im Monat Juli ist es warm
```

Der dritte print Befehl und die entsprechend dritte Ausgabezeile ist hier besonders interessant. $$Monat ist eine variable Variable. Sie bedeutet soviel wie "Zeige mir den Inhalt der Variable, deren Name in $Monat steht".

Da $Monat den Wert "Juli" hat, entspricht $$Monat gleich $Juli. Und $Juli hat den Wert "warm".

4.8.13. PHP Scripte ausserhalb von Apache

Wenn Sie sich mit PHP vertraut gemacht haben, werden sie diese Programmiersprache möglicherweise auch ausserhalb des Apache Webserver anwenden wollen.

Das ist unter Unix durchaus möglich, unter Windows leider nicht.

Das Beispiel test2.php war eine dynamische Webseite, die das aktuelle Datum und die Uhrzeit anzeigt. Ein klein wenig abgeändert können sie dieses Script auch an der Shell Kommandozeile starten:

```
#!/usr/local/bin/php
Dies ist meine erstes PHP-Shell-Script.
<?php
            print("Heute ist der ".strftime("%d.%m.%Y").
                        " und es ist ".strftime("%H:%M:%S")." Uhr.\n");
?>
```

Geben Sie den Befehl

chmod a+x test2.php

ein, damit die Datei als Script ausgeführt werden darf. Wenn Sie nun an der Kommandozeile

./test2.php

eingeben, wird das Script ausgeführt und Sie erhalten die gleiche Ausgabe, wie zuvor - diesmal aber im Terminalfenster und nicht im Webbrowser.

4.8.14. Wie geht es weiter?

Sie haben nun die Grundlagen gelernt, dynamische Webseiten mit MySQL, PHP und dem Apache Webserver zu erstellen. Sie haben erkannt, dass die Verwendung von HTML Formularen nützlich ist.

PHP ist eine mächtige Scriptsprache mit sehr vielen Funktionen. Sie können nicht alle Funktionen auf einmal lernen - dafür sind es einfach zu viele.

Ich möchte Sie nun für alle weiteren Fragen über PHP auf die Original Dokumentation verweisen. Sie können diese direkt von der PHP Webseite http://www.php.net abrufen. Die Webseite hat eine integrierte Suchmaschine und enthält zusätzlich Kommentare von unterschiedlichen PHP Anwendern, die oft nützlich sind.

Ich möchte Ihnen empfehlen, sich zunächst in der angegebenen Reihenfolge mit folgenden Funktionsreferenzen des PHP Handbuches zu beschäftigen:

- Datums und Zeitfunktionen
- Mathematische Funktionen
- String Funktionen
- Funktionen des Dateisystems
- Array Funktionen
- Fehlerbehandlung und Logging Funktionen
- HTTP Funktionen
- Mail Funktionen
- MySQL Funktionen
- Reguläre Ausdrücke

Wenn Sie danach noch nicht genug haben, dann schauen Sie sich die restlichen Funktionsreferenzen an. Sie sind so ausführlich beschrieben (vor allem auf der PHP Webseite), dass es keinen Sinn macht, sie in einem Buch nochmal extra zu beschreiben.

4.9. Shell Script Programmierung

Unter Unix werden viele Aufgaben durch Shell Scripte anstatt mit "echte" Programme erledigt. Shell Scripte können schnell programmiert und geändert werden. Der Nachteil von Shellscripten ist, dass sie relativ langsam sind. Bei einfachen Aufgaben ist das normalerweise nicht schlimm.

Dos Anwender kennen den Begriff Shell-Script nicht, aber sie kennen Batch-Dateien. Im Grunde genommen ist dies das Selbe, Shell Scripte können allerdings deutlich mehr. Während man unter Dos nur selten über die sequentielle Abarbeitung einer Befehlsliste hinaus kommt, ist es in Shell Scripten üblich, Kontrollstrukturen zu benutzen, also if-Befehle, While-Schleifen, For-Schleifen usw.

In diesem Kapitel zeige ich Ihnen, wie Sie Shell Scripte schreiben.

Auch unter Windows können Sie dieses Kapitel anwenden, sofern Sie das CygWin Paket installiert haben.

Unter Unix gibt es zahlreiche unterschiedliche Shells, die alle ihre eigene Programmiersprache haben. In diesem Buch beziehe ich mich auf die Bourne Shell, die zum Standardumfang jedes Unix Betriebssystems gehört.

Bitte nehmen Sie zur Kenntnis, dass Sie alle einzelnen Befehle auch manuell an der Eingabeaufforderung eingeben können. Sie müssen also nicht jedes Beispiel abschreiben und in eine Datei speichern - können Sie aber, wenn Sie möchten.

4.9.1. Das erste Shell Script

Speichern Sie sich das folgende Script als test1.sh ab:

```
#!/bin/sh
# Das ist das erste Beispielscript test1.sh
echo "Hallo"
echo -e "Heute ist der \c"
date '+%d.%m.%y'
```

Dieses Script zeigt das aktuelle Datum an. Damit Sie es ausführen können, müssen Sie den Befehl

chmod a+x test1.sh

eingeben, sonst wird es von der Shell nicht als ausführbares Script erkannt. Geben Sie jetzt

./test1.sh

ein, um das Script auszuführen.

Die erste Zeile sollten Sie in jedes Script exakt wie oben schreiben. Sie teilt der Shell, die Ihre Eingabeaufforderung darstellt, mit, welche Shell das Script ausführen soll. /bin/sh ist die Borune Shell. Wenn Sie diese Zeile weg lassen, kann es passieren, dass das Script von einer falschen Shell ausgeführt wird und es somit zu unerwarteten Fehlermeldungen kommt.

Die zweite Zeile ist ein Kommentar. In der dritten und vierten Zeile wird der echo Befehl benutzt, um Text auszugeben.

Der date Befehl zeigt das aktuelle Datum an. Eine Erklärung des Parameters '+%d.%m.%y' finden Sie in der Manual Page von date:

man date

Echo Befehle schliessen die Textausgabe immer mit einem Zeilenumbruch ab, es sei denn sie schreiben ein \c an das Ende der Zeichenkette. Daher erscheint das Datum direkt hinter dem Wort "der" und nicht in einer eigenen Zeile.

Andere Nützliche Steuerzeichen sind \t für Tabulator, \a für einen Piepston und \n für einen zusätzlichen Zeilenumbruch.

Die Option -e brauchen entfällt unter Solaris. Unter Linux und Windows ist sie immer dann erforderlich, wenn Sie Steuerzeichen, wie \t, \a, \n und \c benutzen.

Shell Scripte können jeden beliebigen Namen haben. Sie werden beim Lesen dieses Buches feststellen, dass ich Scripte meistens nach dem Schema name.sh benenne. Dies ist meine persönliche Vorliebe, Scripte zu benennen. Einen technischen Grund gibt es dafür nicht.

Shell Scripte werden durch das x-Recht als ausführbar gekennzeichnet, nicht durch den Dateinamen.

4.9.2. Der printf Befehl

Den printf Befehl kennen Sie bereits von PHP, falls Sie das Kapitel gelesen haben.

Printf gibt einen Text auf und fügt zusätzlich Zahlen oder Texte ein. Beispiel (test2.sh):

```
#!/bin/sh
printf "Die Zahl %i gefällt mir %s \n" 13 gut
```

Der Befehl gibt anstelle des Platzhalters %i die Zahl 13 aus, und anstelle des Platzhalter %s gibt es das Wort "gut" aus. Die korrekte Reihenfolge der Argumente ist wichtig.

Am Ende habe ich ein \n geschrieben, weil printf die Texte nicht automatisch mit einem Zeilenumbruch abschliesst, wie echo es tut.

Von den vielen denkbaren Platzhaltern sind diese gebräuchlich:

- %i Dezimalzahl
- %s Text
- %f Fliesskommazahl
- %e Fliesskommazahl in Exponentialschreibweise
- %o Oktalzahl
- %x Hexadezimalzahl mit kleinen Buchstaben
- %X Hexadezimalzahl mit großen Buchstaben

Bei Dezimalzahlen können Sie die gewünschte Ausgabebreite festlegen, indem Sie die Anzahl der Zeichen mit angeben:

printf "%10d" 12

Dieser Befehl gibt die Zahl 12 rechtsbündig mit 12 Zeichen Breite aus. Es wird mit Leerzeichen aufgefüllt. Wenn die Längenangabe mit einer 0 beginnt, wird mit Nullen aufgefüllt.

Bei Fliesskommazahlen können Sie die Anzahl der Nachkommastellen festlegen:

printf "%.2f" 12.345678

Dieser Befehl gibt "12.35" aus, die Zahl wurde also korrekt aufgerundet. Leider können Sie bei Fliesskommazahlen nicht festlegen, wie viele Stellen vor dem Komma angezeigt werden. Es werden immer alle Ziffern angezeigt und die Zahl wird nicht rechtsbündig ausgerichtet.

Achten Sie darauf, dass Sie bei printf genau so viele Werte angeben, wie Platzhalter im Text stehen. Es ist nicht fest geregelt, was passiert, wenn Sie zuviele oder zu wenig Werte angeben.

4.9.3. Zeichenketten

Zeichenketten (oder Texte) schreiben Sie grundsätzlich in Anführungsstriche. Manche Befehle funktionieren meist auch ohne (z.B. echo), Sie sollten sich dennoch angewöhnen, Zeichenketten immer in Anführungsstriche einzuschliessen. Beispiel (test3.sh):

```
#!/bin/sh
echo "Hallo      Stefan"
echo  Hallo      Stefan
```

Diese beiden Befehle erzeugen nicht die gleiche Ausgabe. Im zweiten Fall werden Sie zwischen "Hallo" und "Stefan" nur ein Leerzeichen sehen, weil Sie die Anführungsstriche weggelassen haben. Tatsächlich handelt es sich nun nämlich nicht mehr um einen einzigen Text, sondern um zwei Texte. Die Shell benutzt Leerzeichen und Tabulator-Zeichen als Trennzeichen zwischen zwei Argumenten, wobei mehrere aufeinanderfolgende Trennzeichen ignoriert werden.

Wenn Sie innerhalb einer Zeichenkette ein Steuerzeichen ausgeben wollen, dass normalerweise eine andere Funktion hat, müssen Sie ein \ voranstellen (test4.sh):

```
#!/bin/sh
echo "Hallo \"Stefan\" "
```

Es zeigt

```
Hallo "Stefan"
```

an. Ein weiteres Steuerzeichen, dass Sie nur mit vorangestellem \ anzeigen können, ist das Dollar-Zeichen. Das \ Zeichen selbst bekommen Sie logischerweise mit \\.

Sie können Zeichenketten auch in einfache Anführungsstriche schreiben. Der Unterschied ist, dass innerhalb einfacher Anführungsstriche " und $ keine besondere Bedeutung mehr haben. Beispiel (test5.sh):

```
#!/bin/sh
echo "Dein Name ist $USER"
echo 'Dein Name ist $USER'
echo 'Hallo "Stefan"'
echo "Hallo 'Stefan'"
```

Der erste Befehl zeigt den aktuellen Usernamen an, eine Umgebungsvariable der Shell. Der zweite Befehl zeigt hingegen nur den Namen der Variable an. Die beiden anderen Befehle demonstrieren, dass einfache Anführungsstriche innerhalb von doppelten Anführungsstrichen keine besondere Bedeutung haben - und umgekehrt ebenfalls.

4.9.4. Variablen

Variablen können Sie setzen und wieder auslesen. Variablen in Shell Scripten speichern grundsätzlich nur Text. Wichtig ist, dass sie um den Zuweisungsoperator = keine Leerzeichen schreiben! (test6.sh):

```
#!/bin/sh
variable1="Stefan"
echo "Hallo $variable"
variable2=0123
echo $variable2
```

Beim zweiten Echo Befehl wird auch brav 0123 angezeigt, weil dies - wie bereits geschrieben -als Text behandelt wird, und nicht als Zahl. Daher bleibt die voranstehende Null auch erhalten.

Ich möchte Ihnen nun eine Eigenart beim Auslesen von Variablen zeigen (test7.sh):

```
#!/bin/sh
variable3="Hallo        Stefan"
echo "$variable3"
echo $variable3
```

Dieses Script erzeugt die Ausgabe:

```
Hallo        Stefan
Hallo Stefan
```

Dieses Beispiel demonstriert, dass der Variablenname durch den Inhalt ersetzt wird, **bevor** der Text in die einzelnen Wörter zerlegt wird. Die mehrfachen Leerzeichen werden von der Shell ignoriert. Aus dem Text "Hallo Stefan" sind nun zwei einzelne Wörter geworden und die Leerzeichen sind verloren gegangen. Wenn Sie wissen, dass in einer Variablen mehr als nur ein einziges Wort steht, sollten Sie die Variablennamen also immer in Anführungsstriche schreiben.

Variablen, die Sie in einem Shell Script setzen, sind ausserhalb des Shell-Scriptes nicht sichtbar. Der export Befehl macht Variablen extern sichtbar. Beispiel (test8.sh):

```
#!/bin/sh
Name1="Cicelle"
Name2="Monique"
echo "$Name1 $Name2"
export $Name2
```

Führen Sie das Script einmal aus, indem Sie

./test9.sh

eingeben. Die beiden Namen werden angezeigt. Geben Sie nun

echo "$Name1 $Name2"

ein. Sie sehen nur den Namen Monique, denn nur dieser wurde mittels export Befehl exportiert. Der Name Cicelle ist ausserhalb des Scriptes unsichtbar.

Eine andere Möglichkeit, Variablen zu exportieren, geht so (test9.sh):

```
#!/bin/sh
Name1="Cicelle"
Name2="Monique"
echo "$Name1 $Name2"
```

Führen Sie das Script aus, indem Sie

. ./test9.sh

eingeben. Prüfen Sie den Erfolg, indem Sie nochmal

echo "$Name1 $Name2"

eingeben. Dieses mal sind beide Namen auch ausserhalb des Scriptes bekannt geworden, denn Sie haben das Script mit dem Punkt-Befehl gestartet. Man sieht sowas nur selten, denn der Punkt-Befehl hat einen entscheidenden Nachteil: Man kann sich nicht mehr aussuchen, welche Variablen man extern bekannt machen will. Es werden einfach alle Variablen exportiert.

Wenn Sie innerhalb eines Scriptes ein anderes aufrufen - sozusagen als Unterprogramm - dann kennt dieses keine Variablen des haupt-Scriptes. Auch das haupt-Script kennt keine Variablen des Unter-Scriptes, ausser wenn diese exportiert werden. Das klingt kompliziert, es ist aber ganz einfach. Schauen Sie sich dieses Beispiel an :

```
#!/bin/sh
# Diese Datei heisst test10.sh
Name1="Rafael"
echo "Der erste Name ist $Name1."
./test11.sh
echo "Der zweite name ist $Name2."


#!/bin/sh
# Diese Datei heisst test11.sh
echo "--- Beginn Unterprogramm"
echo "Der erste Name ist $Name1."
Name2="Alexander"
echo "Der zweite name ist $Name2."
echo "--- Ende Unterprogramm"
```

Speichern Sie sich diese beiden Scripte ab und setzen Sie wie gehabt mit **chmod a+x test*.sh** das Ausführrecht. Geben Sie dann ein:

./test10.sh

Sie erhalten dann folgende Bildschirmausgabe:

```
Der erste Name ist Rafael.
--- Beginn Unterprogramm
Der Erste Name ist .
Der zweite Name ist Alexander.
--- Ende Unterprogramm
Der Zweite Name ist .
```

Wie Sie sich bereits warscheinlich denken, können Sie $Name2 mit export exportieren, oder Sie rufen das Unterprogramm mit dem Punkt-Befehl auf, um diesen Fehler zu beheben. Dann wird Alexander auch im Hauptprogramm bekannt sein.

Abschliessend möchte ich noch etwas zu den variablen-Namen bemerken: Schreiben Sie Variablennamen nicht komplett in großen Buchstaben. Alle Variablen des Betriebssystems sind nämlich so geschrieben, und als Anwender sollten sich Ihre eigenen Variablennamen eindeutig davon unterscheiden. So kann es Ihnen nicht passieren, aus Versehen eine Variable des Betriebssystems zu verändern.

Ein Beispiel: Angenommen Sie schreiben aus Versehen **PS1="Hallo"**, weil Sie keine Ahnung hatten, dass diese Variable vom Betriebssystem benutzt wird... versuchen Sie es, die Auswirkung ist harmlos, aber sicher überraschend.

Geben Sie

PS1="Hallo"

ein und bestaunen Sie das Ergebnis. Ihre Eingabeaufforderung hat sich verändert! Wenn Sie kleine Buchstaben verwendet hätten, wäre das nicht passiert.

Falls Sie sich eine Liste aller gerade gesetzten Shell-Variablen ansehen wollen, dann geben Sie einmal

env | more

ein. Einige Variablen erscheinen Ihnen sicher als nützlich.

4.9.5. Ein/Ausgabe Umleitung in Dateien

Unter Ein/Ausgabe Umleitung versteht man den Mechanismus, der die Eingabe oder die Ausgabe eines Programmes in ein anderes Programm, eine Datei oder eine Variable umleitet.

Die Eingabe ist normalerweise die Tastatur, wärend für die Ausgabe normalerweise der Bildschirm benutzt wird. Doch das ist änderbar:

echo "Hallo" > test.txt

Dieser Befehl zeigt nicht das Wort "Hallo" an, sondern schreibt es in die Datei test.txt. Schauen Sie nach!

echo "Fachmann" >> test.txt

Dieser Befehl hängt ein weiteres Wort an die bereits vorhandene Datei an. Der doppelte Pfeil weist die Shell an, keine neue Datei anzulegen, sondern an die vorhandene Datei anzuhängen - falls vorhanden. Wenn die Datei noch nicht existiert, wird eine neue angelegt.

Auch die Eingabe ist umlenkbar. Dafür benutzen Sie einen Pfeil im umgekehrter Richtung. Der Befehl read liest eine Zeile von der Tastatur und speichert Ihre Eingabe in eine Variable:

read eingabe
echo "$eingabe"

Sie könnten die Eingabe zum Beispiel in eine Datei umleiten:

read eingabe < test.txt
echo "$eingabe"

Nun liest der read Befehl nicht mehr von der Tastatur, sondern er liest die erste Zeile aus der Textdatei aus.

4.9.6. Ein/Ausgabe Umleitung in Programme

Sie können Eingaben und Ausgaben nicht nur in Dateien umleiten, sondern auch in Programme. Das geht so:

env | more

Der Befehl "env" zeigt die alle Umgebungsvariablen der Shell an. Durch die Umleitung in den more Befehl betrachten Sie die Variable seitenweise. Denn more liest von dem Eingabe und schreibt den Text seitenweise auf den Bildschirm.

4.9.7. Ein/Ausgabeumleitung in Variablen

Ausgaben kann man auch in Variablen umleiten. Ein Beispiel:

Name=`hostname`
echo $Name

Der Befehl hostname zeigt den Namen des Computers an. Durch obigen Befehl leiten Sie dessen Textausgabe in eine Variable um. Beachten Sie, dass die hier verwendeten Zeichen keine Anführungsstriche sind. Die Zeichen ' und ` sind unterschiedliche Zeichen mit völlig unterschiedlicher Bedeutung. Das Zeichen `wird Backtick oder Backquote genannt.

Sie können die Ausgabe von Befehlen auch in Texte einfliessen lassen, allerdings nur, wenn der Text in doppelten Anführungsstrichen steht:

echo "Hallo `hostname`"
echo 'Hallo `hostname`'

Das erste Beispiel funktioniert, während das zweite Beispiel nicht funktioniert.

Es gibt keine direkte Möglichkeit, die Eingabe eines Befehls in eine Variable umzuleiten. Doch man kann sich mit einem echo Befehl behelfen:

Hilfe=`man tar`
echo "$Hilfe" | more

Die Manual Page vom Programm tar wird in die Variable $Hilfe geschrieben. Mittels echo Befehl geben Sie die Variable aus und leiten den Text in den more Befehl um.

4.9.8. Mehrere Befehle in einer Zeile

Sie können mehrere Befehle in eine Zeile schreiben. Je nach Schreibweise, werden diese dann unterschiedlich ausgeführt.

echo "Hallo" ; echo "Rita" ; echo "viel Freude"

Diese drei echo Befehle werden der Reihe nach ausgeführt. Sie erhalten also die Bildschirmausgabe:

```
Hallo
Rita
viel Freude
```

Ein anderes Beispiel:

rm /tmp/test.txt && echo "Die Datei wurde gelöscht"

Zuerst wird versucht, die Datei /tmp/test.txt zu löschen. Nur wenn das erfolgreich war, erscheint die Erfolgsmeldung. Der Echo Befehl wird also bedingt ausgeführt.

Noch ein Beispiel:

rm /tmp/test.txt || echo "Die Datei konnte nicht gelöscht werden"

Zuerst wird versucht, die Datei /tmp/test.txt zu löschen. Wenn das nicht erfolgreich war, erscheint die Fehlermeldung.

An dieser Stelle möchte ich erklären, was genau "erfolgreich" und "nicht erfolgreich" ist.

Jedes Programm und jeder Befehl (Befehle sind Programme) liefern am Ende einen Exit Code an die Shell zurück. Das ist eine Zahl im Bereich 0 bis 255. Die 0 wird als erfolgreich gewertet, alle anderen Werte sind Fehlercodes.

Die Zahlen 1-255 haben bei jedem Programm eine andere Bedeutung. Bei den meisten Befehlen finden Sie eine Erklärung der möglichen Exit-Codes in deren Manual Page.

4.9.9. Kontrollstrukturen

Kontrollstrukturen ermöglichen Ihnen, die Befehle eines Shell Scriptes nicht einfach nur sequentiell von oben nach unten abzuarbeiten, sondern Wiederholungen oder bedingte Programmabschnitte einzubauen.

4.9.9.1. if...then...else...fi

Mit der Kontrollstruktur if...then...else...fi führen Sie Programmabschnitte bedingt aus, wenn ein Vergleich erfolgreich war oder ein Testbefehl 0 zurück gibt. Die Schreibweise ist:

```
if Befehl ; then
  Programmabschnitt1
fi
```

oder so:

```
if Befehl ; then
  Programmabschnitt1
else
  Programmabschnitt2
fi
```

Wenn der Befehl erfolgreich war, dann wird der Programmabschnitt1 ausgeführt. Wenn er nicht erfolgreich war, wird der Programmabschnitt2 ausgeführt, falls vorhanden.

Sehr selten sieht man diese erweiterte Variante, in der mehrere Alternative Befehle getestet werden:

```
if Befehl1 ; then
  Programmabschnitt1
elif Befehl2 ; then
  Programmabschnitt2
else
  Programmabschnitt3
fi
```

Wenn der Befehl1 erfolgreich war, wird der Programmabschnitt1 ausgeführt. Amsonsten wird der Befehl2 ausgeführt. Wenn Befehl2 erfolgreich war, wir Programmabschnitt2 ausgeführt. Amsonsten wird Programmabschnitt3 ausgeführt. Der elif-Abschnitt kann mehrmals wiederholt werden.

Ein konkretes Beispiel (test12.sh):

```
#!/bin/sh
if rm /tmp/test.txt ; then
  echo "Das Löschen war erfolgreich"
  echo "Alles Ok"
else
  echo "Fehler:"
  echo "Die Datei konnte nicht gelöscht werden"
fi
```

Und nun folgt ein Beispiel für die erweiterte Version (test13.sh):

```
#!/bin/sh
if rm /tmp/test.txt ; then
  echo "Das Löschen von test.txt war erfolgreich"
elif  rm /tmp/test2.txt ; then
  echo "Das Löschen von test2.txt war erfolgreich"
elif  rm /tmp/test3.txt ; then
  echo "Das Löschen von test3.txt war erfolgreich"
else
  echo "Keine der drei Dateien konnte gelöscht werden"
fi
```

Manche Leute schreiben das Schlüsselwort "then" lieber in eine eigene Zeile, dann entfällt das Semikolon:

```
#!/bin/sh
if rm /tmp/test.txt
then
  echo "Das Löschen war erfolgreich"
fi
```

4.9.9.2. Vergleiche

Wenn Sie einen Programmabschnitt bedingt ausführen wollen, möchten Sie dies oft tun, nachdem Sie zwei Texte oder Zahlen miteinander verglichen haben. Dazu benutzen Sie den **test** Befehl oder die eckigen Klammern **[ ]**.

Eine Manual Page zu den eckigen Klammern werden sie nicht finden, denn dahinter steckt in Wirklichkeit der test Befehl.

["Hallo" = "Hallo"]
echo $?

Liefert den Wert 0, weil der Vergleich erfolgreich ist. Dieser Befehl ist identisch mit:

test "Hallo" = "Hallo"
echo $?

Der echo $? Befehl zeigt den Exit-Code des vorherigen Befehls an. $? ist eine spezielle Variable, auf die ich später nochmal zurück komme.

De meisten Programmierer bevorzugen die eckigen Klammern, weil er sich vom Rest des Shell-Scriptes optisch gut abzeichnet. Achten Sie auf die Leerzeichen! Hier ist es wichtig, dass Sie keine Leerzeichen weglassen, sonst passiert nicht das, was Sie erwarten.

Texte Vergleichen

["Text1" = "Text2"]	0, wenn beide Texte gleich sind
["Text1" != "Text2"]	0, wenn beide Texte unterschiedlich sind
["Text1" < "Text2"]	0, wenn Text1 alphabetisch vor Text2 kommt
["Text1" > "Text2"]	0, wenn Text1 alphabetisch nach Text2 kommt
[-z "Text"]	0, wenn der Text leer ist
["Text"]	0, wenn der Text nicht leer ist

Bei der alphabetischen Reihenfolge stehen die Buchstaben A-Z vor a-z, entsprechend der ASCII Tabelle.

Zahlen vergleichen

[Zahl1 -eq Zahl2]	0, wenn beide Zahlen gleich sind
[Zahl1 -ne Zahl2]	0, wenn die Zahlen nicht gleich sind
[Zahl1 -lt Zahl2]	0, wenn Zahl1 kleiner als Zahl2 ist
[Zahl1 -gt Zahl2]	0, wenn Zahl1 grösser als Zahl2 ist
[Zahl1 -le Zahl2]	0, wenn Zahl1 kleiner oder gleich Zahl2 ist
[Zahl1 -ge Zahl2]	0, wenn Zahl1 grösser oder gleich Zahl2 ist

Sie können nur Integer-Zahlen vergleichen, das sind Zahlen ohne Nachkommastellen.

Die Eckigen Klammern oder auch der test Befehl machen nur Sinn, wenn man sie zusammen mit Kontrollstrukturen, wie z.B. if...then...else...fi benutzt. Ein Beispiel (test13.sh):

```
#!/bin/sh
Zahl = 12
if [ $Zahl -gt 5 ] ; then
   echo "Die Zahl $Zahl ist grösser als 5"
else
   echo "Die Zahl $Zahl ist nicht grösser als 5"
fi
```

Schauen Sie sich das Ergebnis an. Ändern Sie Zahl=12 in Zahl=3 um und vergleichen Sie das Ergebnis mit der vorherigen Ausgabe.

Und hier ist noch ein Beispiel (test14.sh):

```
#!/bin/sh
Name = "Manuela"
if [ $Name = "Manuela" ] ; then
   echo "Der Name $Name ist Manuela"
else
   echo "Der Name $Name ist nicht Manuela"
fi
```

Dateinamen prüfen

[-r "Dateiname"]	0, wenn die Datei lesbar ist
[-w "Dateiname"]	0, wenn die Datei beschreibbar ist
[-x "Dateiname"]	0, wenn die Datei ausführbar ist
[-f "Dateiname"]	0, wenn es eine normale Datei ist
[-d "Dateiname"]	0, wenn es eine Verzeichnis ist
[-L "Dateiname"]	0, wenn es ein symbolische Link ist
[-e "Name"]	0, wenn der Name als Datei, Verzeichnis oder Link existiert

Dazu ein Beispiel (test15.sh):

```
#!/bin/sh
if [ -f "/tmp/test.txt" ] ; then
   echo "Die Datei /tmp/test.txt existiert"
fi
```

4.9.9.3. Mehrere Vergleiche verknüpfen

Mehrere Vergleiche können logisch mit UND und ODER verknüpft werden. Auch die Negation gibt es.

["Text1" = "Text2" -a "Text3" = "Text4"]	-a ist eine UND Verknüpfung
["Text1" = "Text2" -o "Text3" = "Text4"]	-o ist eine ODER Verknüpfung
[! "Text1" = "Text2"]	! ist die negation. Sie macht wenig Sinn, denn man könnte auch != anstelle von = schreiben.

Verknüpfungen dieser Art werden nur sehr selten gebraucht. Ein sinnvolles Anwendungsbeispiel (test16.sh):

```
#!/bin/sh
echo "Bitte geben Sie eine Zahl ein"
read Zahl
if [ $Zahl -lt 10 -o $Zahl -gt 100 ] ; then
   echo "Die Zahl ist leiner als 10 oder grösser als 100"
else
   echo "Die Zahl ist im Bereich 10 bis 100"
fi
```

Eine eher gebräuchlichere Variante, mehrere Vergleiche zu verknüpfen is die folgende, obwohl sie etwas langsamer ist:

["Text1" = "Text2"] && ["Text3" = "Text4"]	&& ist eine UND Verknüpfung
["Text1" = "Text2"] || ["Text3" = "Text4"]	|| ist eine ODER Verknüpfung

Dazu ein Beispiel (test17.sh):

```
#!/bin/sh
echo "Bitte geben Sie eine Zahl ein"
read Zahl
if [ $Zahl -lt 10 ] || [ $Zahl -gt 100 ] ; then
   echo "Die Zahl ist leiner als 10 oder grösser als 100"
else
   echo "Die Zahl ist im Bereich 10 bis 100"
fi
```

4.9.9.4. case...esac

Wenn Sie viele if und elif Befehle verketten, kann der Programmtext schnell unübersichtlich werden. Ein schlechtes Beispiel is test18.sh:

```
#!/bin/sh
echo "Geben Sie eine Zahl ein"
read Zahl
if [ $Zahl -eq 1 ] ; then
   echo "Sie haben eine Eins eingegeben"
elif [ $Zahl -eq 2 ] ; then
   echo "Sie haben eine Zwei eingegeben"
elif [ $Zahl -eq 3 ] ; then
   echo "Sie haben eine Drei eingegeben"
elif [ $Zahl -eq 4 ] ; then
   echo "Sie haben eine Vier eingegeben"
else
   echo "Sie haben eine andere Zahl eingegen"
fi
```

Sie könnten dieses Script auch mit der case...esac Anweisung schreiben. Oft sieht das übersichtlicher aus (test19.sh):

```
#!/bin/sh
echo "Geben Sie eine Zahl ein"
read Zahl
case $Zahl in
   1)   echo "Sie haben eine Eins eingegeben" ;;
   2)   echo "Sie haben eine Zwei eingegeben" ;;
   3)   echo "Sie haben eine Drei eingegeben" ;;
   4)   echo "Sie haben eine Vier eingegeben" ;;
   *)   echo "Sie haben etwas anderes eingegeben" ;;
esac
```

Vergessen sie die doppelten Semikolons nicht! Wenn Sie mehrere Befehle pro passende Bedingung ausführen wollen, können Sie dies so schreiben (test20.sh):

```
#!/bin/sh
echo "Geben Sie eine Zahl ein"
read Zahl
case $Zahl in
   1)   echo "Sie haben eine Eins eingegeben"
        echo "Eins gefaellt mir"
        ;;
   2)   echo "Sie haben eine Zwei eingegeben"
        echo "Zwei ist auch ganz nett"
        ;;
esac
```

Dieses Beispiel demonstriert, dass die doppelten Semikolons auch in einer eigenen Zeile stehen dürfen und dass Sie den Abschnitt *) weglassen dürfen.

Sie können auch mehrere Vergleichswerte als ODER Funktion verknüpfen (test21.sh):

```
#!/bin/sh
echo "Geben Sie eine Zahl ein"
read Zahl
case $Zahl in
   1|2)    echo "Sie haben eine Eins oder Zwei eingegeben" ;;
   3|4|9)  echo "Sie haben eine Drei oder Vier oder Neun eingegeben" ;;
esac
```

Die case...esac Anweisung funktioniert auch mit Texten. Schreiben Sie diese wie gehabt immer in Anführungsstriche, um Probleme mit Leerzeichen zu vermeiden (test22.sh):

```
#!/bin/sh
echo "Geben Sie ein Wort ein"
read Wort
case "$Wort" in
   "rot"|"gruen")  echo "Sie haben rot oder grün eingegeben" ;;
   "gelb")         echo "Sie haben gelb eingegeben" ;;
   *)              echo "Sie haben etwas anderes eingegeben" ;;
esac
```

4.9.9.5. while...do...done

Die while Schleife führt einen Block von Befehlen wiederholt aus, solange eine Bedingung erfüllt ist oder ein Testbefehl 0 zurückliefert. Ein Beispiel (test23.sh):

```
#!/bin/sh
while read Text ; do
   echo "Sie haben $Text eingegeben"
done
```

Solange der read Befehl erfolgreich war, wird der echo Befehl wiederholt. Hierbei ist es nützlich zu wissen, wann read denn nicht erfolgreich ist, nämlich wenn Sie Strg-D drücken.

Auch bei der while Schleife können Sie test oder die eckigen Klammern benutzen (test24.sh):

```
#!/bin/sh
echo "Geben Sie etwas ein, oder nur Enter zum beenden"
read Text
while [ "$Text" ] ; do
   echo "Danke"
   read Text
done
```

Sie können hier beliebig viele Zeilen Text eingeben. Doch wenn Sie einfach nur Enter drücken (also eine Leerzeile), wird die Schleife beendet.

4.9.9.6. for...in...do...done

Die for Anweisung führt einen Programmabschnitt mehrmals aus, und zwar für jedes Element in einer Liste. Anhand des folgenden Beispiels kann man dies nachvollziehen (test25.sh):

```
#!/bin/sh
for Farbe in rot gelb gruen blau ; do
  echo "Die Farbe ist $Farbe"
done
```

Dieses Script erzeugt die Ausgabe:

```
Die Farbe ist rot
Die Farbe ist gelb
Die Farbe ist gruen
Die Farbe ist blau
```

4.9.9.7. break, continue und exit

Die Anweisung break unterbricht eine laufende Schleife. Es werden keine weiteren Wiederholungen mehr durchgeführt (test26.sh):

```
#!/bin/sh
echo "Geben Sie etwas ein, oder nur Enter zum beenden"
while true ; do
   read Text
   if [ -z "$Text" ] ; then
      break;
   fi
   echo "Danke"
done
```

Dieses Script ist funktionell fast mit test24.sh identisch. Es wurde nur ein anderer Lösungsansatz verfolgt.

Die while Schleife wiederholt sich zunächst unendlich oft, denn der Befehl **true** liefert immer eine 0 (was Erfolgreich bedeutet). Mehr macht der true Befehl nicht. Das Gegenteil ist übrigens der **false** Befehl -was für eine Überraschung!

Die if Anweisung prüft, ob der Text leer ist. Wenn er leer ist, wird die Schleife abgebrochen.

Die continue Anweisung erzwingt die nöchste Schleifenwiederholung und sorgt dafür, dass die restlichen Befehle der aktuellen Schleife nicht mehr ausgeführt werden (test27.sh):

```
#!/bin/sh
echo "Geben Sie etwas ein, oder Strg-D zum beenden"
while read Text ; do
   if [ -z "$Text" ] ; then
      continue;
   fi
   echo "Danke"
done
```

Wenn Sie dieses Script ausführen, können Sie beliebig viele Zeilen eintippen - auch Leerzeilen. Doch ein Dankeschön erhalten sie nur für Zeilen, die nicht leer sind.

Die exit Anweisung beendet das gesamte Shell-Script mit sofortiger Wirkung. Sie können einen Exit-Code mit angeben, wenn Sie möchten. Wenn Sie den Exit-Code weglassen, wird 0 genommen. Beispiel (test28,sh):

```
#!/bin/sh
echo "Geben Sie etwas ein"
read Eingabe
if [ "$Eingabe" = "doof" ] ; then
   echo "Ich bin nicht doof!"
   exit 1
fi
echo "Ok, danke"
```

Nach der Ausführung dieses Scriptes können Sie den Exit-Code mit **echo $?** kontrollieren.

4.9.10. Funktionen

Funktionen sind kleine Unterprogramme, die Sie mehrmals an unterschiedlichen Stellen aufrufen können. Ein Beispiel (test29.sh):

```
#!/bin/sh
function Fehler()
{
    echo "Es ist ein Fehler aufgetreten:"
    echo "$1"
}
echo "Geben Sie bitte eine 3 ein"
read Zahl
if [ $Zahl -ne 3 ]; then
    Fehler "Falsche Zahl"
else
   echo "Danke"
fi
```

Wenn Sie das Script ausführen und eine andere Zahl, als 3 eingeben, erscheint diese Ausgabe:

```
Geben Sie bitte eine 3 ein
5
Es ist ein Fehler aufgetreten:
Falsche Zahl
```

Die Funktion zeigt nicht nur einen festen Text an, sondern auch einen Variablen Text (Falsche Zahl), den Sie vom Hauptprogramm übergeben bekommen hat.

Innerhalb der Funktion wird der erste Übergebene Text über die Variable $1 angesprochen. Wenn es mehrere Texte gäbe, würden diese entsprechend fortlaufend nummeriert werden (test30.sh):

```
#!/bin/sh
function Test()
{
    echo "Die Übergebenen Texte sind:"
    ecfho "$1 + $2 + $3 + $4"
}
Test "Hallo" "Saskia" "meine" "Schwester"
```

Dieses Script erzeugt die Ausgabe:

```
Die Übergebenen Texte sind:
Hallo + Saskia + meine + Schwester
```

4.9.11. Spezielle Variablen

Ein paar spezielle Variablen haben Sie bereits kennen gelernt. Ich möchte Sie ihnen noch einmal in Zusammenhang mit anderen vorstellen:

$0	ist der eigene Name des Scriptes
$1, $2 ... $9	sind die Argumente einer Funktion oder eines Scriptes
$#	ist die Anzahl der Argumente
"$@"	sind alle Argumente als Liste einzelner Texte
"$*"	sind alle Argumente als ein einziger langer Text
$*	oder $@ sind alle Argumente, zerlegt in Worte
$?	ist der Exit-Code des letzten Befehls
$RANDOM	Zufallszahl im Bereich 0-32767 (nur bash Shell)

Wie man die Variablen $1 bis $9 benutzt, haben Sie bereits in Zusammenhang mit den Funktionen gesehen. Sie können die Variablen auch benutzen, um Argumente beim Scriptaufruf zu lesen (test31.sh):

```
#!/bin/sh
echo "Sie haben $# Argumente eingegeben"
echo "Das Script heisst $0"
echo "Das erste Argument ist $1"
echo "Alle Argumente Zusammen sind $*"
```

Starten Sie das Script, indem Sie den Befehl

./test31.sh Eins Zwei Drei

eingeben. Die Bildschirmausgabe ist dann:

```
Sie haben 3 Argumente eingegeben
Das Script heisst ./test31.sh
Das erste Argument ist Eins
Alle Argumente Zusammen sind Eins Zwei Drei
```

Im folgenden Beispiel möchte ich Ihnen den Unterschied zwischen "$@" und "$*" demonstrieren (test32.sh):

```
#!/bin/sh
for Wort in "$@"; do
    echo "-- $Wort"
done
echo ""
for Wort in "$*"; do
    echo "-- $Wort"
done
```

Die erste for-Schleife gibt jedes Wort einzeln aus, "$@" hat also eine Liste mit drei Wörtern erzeugt. Die zweite for-Schleife gibt aber nur eine Zeichenkette aus, alle Drei Wörter sind zu einem einzigen Text verschmolzen:

./test32.sh Eins Zwei Drei

```
-- Eins
-- Zwei
-- Drei

-- Eins Zwei Drei
```

./test32.sh "Eins 1" "Zwei 2" "Drei 3"

```
-- Eins 1
-- Zwei 2
-- Drei 3

-- Eins 1 Zwei 2 Drei 3
```

Wenn Sie die beiden Ausgaben miteinander vergleichen, bemerken Sie, dass der Text "Eins 1" nicht auseinandergerissen wird.

Wenn Sie $@ oder $* ohne Anführungsstriche Schreiben, bekommen Sie in beiden Fällen eine Liste mit einzelnen Worten. In diesem Fall wird "Eins 1" auch auseinandergenommen (test33.sh):

```
#!/bin/sh
for Wort in $*; do
    echo "-- $Wort"
done
```

./test33.sh "Eins 1" "Zwei 2" "Drei 3"

```
-- Eins
-- 1
-- Zwei
-- 2
-- Drei
-- 3
```

Es gibt als drei unterschiedliche Möglichkeiten, alle Argumente gleichzeitig auszulesen.

4.9.12. Mehr als neun Argumente

Sie haben gerade gelernt, dass die Variablen $1 bis $9 die Argumente represäntieren, die Sie einer Funktion oder einem Shell-Script beim Aufruf übergeben.

Tatsächlich ist die Anzahl der Argumente jedoch nicht auf 9 beschränkt. Die Grenze von 9 umgehen Sie mit dem Shift Befehl (test34.sh):

```
#!/bin/sh
echo "Das erste Argument ist $1"
shift
echo "Das zweite Argument ist $1"
shift
echo "Das dritte Argument ist $1"
shift
echo "Das vierte Argument ist $1"
```

Der Shift Befehl löscht das erste Argument und verschiebt alle anderen um eine Position nach links. Aus $2 wird also $1, aus $3 wird $2, und so weiter. Den Shift Befehl können sie beliebig oft benutzen, und so kommen Sie auch an Argumente heran, die über $9 hinaus gehen.

4.9.13. Arithmetische Ausdrücke mit expr

Shell Skripte können keine mathemaischen Ausdrücke berechnen. Der Ausdruck

Ergebnis=1 + 2

funktioniert so nicht. Mathematische Ausdrücke muessen Sie mit dem Hilfsprogramm expr berechnen:

```
Ergebnis=`expr 1 + 2`
echo $Ergebnis
```

Leider können Sie mit expr nur Integer-Zahlen (ohne Nachkommastellen) berechnen. Die folgenen Ausdrücke werden von expr verstanden:

Zahl1 + Zahl2	Addition
Zahl1 - Zahl2	Subtraktion
Zahl1 * Zahl2	Multiplikation
Zahl1 / Zahl2	Division
Zahl1 % Zahl2	Rest einer Division (Modulus)
substr Text Start Anzahl	Liefert Anzahl Buchstaben vom Text beginnend bei Start "substr Hallo 2 3" liefert "all"
index Text Buchstaben	Liefert die Position wo als erstes einer der Buchstaben steht "index Hallo ab" liefert "2"
length Text	Liefert die Länge des Textes

Vergessen Sie die Leerzeichen nicht und beachten Sie, dass der Stern in der Shell eine besondere Bedeutung hat. Sie müssen daher immer ein Backslash \ vor den Stern schreiben. Das gilt auch für Klammern.

In Ausdrücken können Sie Klammern benutzen, um die Rangfolge festzulegen.

```
Ergebnis=`expr \( 1 + 2 \) \* 3`
echo $Ergebnis
```

```
9
```

```
Text="Hallo"
Laenge=`expr length "$Text"`
echo $Laenge
```

```
5
```

```
expr 6 \* 5
```

```
30
```

Der letzte Befehl zeigt, wie Sie das Ergebnis einfach nur anzeigen, anstatt es in eine Variable zu speichern. Man sieht sowas nur selten, denn meistens will man das Ergebnis in irgend einer Form weiter verarbeiten, z.B. in einer if...then...fi Kontrollstruktur.

4.9.14. Fehlermeldungen

Sie haben bereits gelernt, wie Sie die Ausgabe von Programmen in Variablen, Dateien oder in andere Programme umleiten. Diese Umleitungen wirken sich aber nicht auf Fehlermeldungen aus.

Ein Beispiel:

rm xxxaaabbbccc > /dev/null

Der Befehl soll eigentlich die Datei xxxaaabbbccc löschen. Da die Datei aber gar nicht existiert, erscheint eine Fehlermeldung.

Nun haben Sie ja die Ausgabe nach /dev/null umgeleitet, aber dennoch erscheint die Fehlermeldung auf dem Bildschirm.

Das liegt daran, dass es in Wirklichkeit zwei unterschiedliche Ausgabekanäle gibt. Der Kanal 1 ist für normale Ausgaben gedacht, während der Kanal 2 für Fehlermeldungen genutzt wird. Die Ausgabeumleitung im obigen Befehl wirkt sich aber nur auf Kanal 1 aus.

Falls Sie sich jetzt fragen, was denn /dev/null für eine Datei sein soll: Das ist keine wirkliche Datei, sondern ein Gerätetreiber, der einfach alles verschluckt und nichts damit anfängt. Sie können /dev/null sozusagen als "Mülleimer" für Ausgaben benutzen, die Sie nicht sehen haben wollen.

Wir haben aber immer noch das Problem, dass die Fehlermeldung trotz Umleitung erscheint. Die Lösung geht so:

rm xxxaaabbbccc > /dev/null 2>&1

Durch den Anhang 2>&1, der übrigens ganz am Ende stehen muss, teilen Sie der Shell mit, dass der Fehlerkanal 2 genau dorthin umgeleitet werden soll, wo auch der Ausgabekanal 1 hin geht. Dieses mal erscheint die Fehlermeldung also nicht mehr.

Das dies auch bei Umleitung in Variablen funktioniert, zeigt dieses Beispiel:

Ergebnis=`rm xxxaaabbbccc 2>&1`
echo $Ergebnis

Falls Sie einmal selbst in einem Shell-Script eine Fehlermeldung anzeigen möchten, die auf Kanal 2 ausgegeben wird, tun Sie das so (test35.sh):

```
#!/bin/sh
echo "Das ist normaler Text"
echo "Fehler: Irgend etwas stimmt nicht" > /dev/stderr
```

Testen Sie das Script:

Ergebnis=`./test35.sh`
Echo $Ergebnis

Der normale Text steht nun in der Variable Ergebnis, doch die Fehlermeldung erscheint direkt auf dem Bildschirm und ist nicht in die Variablen geschrieben worden.

4.9.15. Nützliche Hilfsprogramme

Ich möchte Ihnen nun ein paar weitere nützliche Hilfsprogramme vorstellen, die man in Shell Scripten häufig benutzt. Dabei geht es mir nicht darum, alle Funktionen dieser Programme zu beschreiben, sondern nur diejenigen, die ich in Zusammenhang mit SMS Anwendungen für wichtig halte.

Wenn Sie sich tiefer gehend informieren wollen, sollten Sie die Manual Pages der Programme lesen.

4.9.15.1. cat

Der Befehl cat zeigt den Inhalt einer Textdatei an:

cat test.txt

Sie können die Ausgabe in eine Variable umleiten, um den Inhalt der Textdatei in die Variable zu bekommen:

Variable=`cat test.txt`

Wenn Sie den Dateinamen weglassen, liest cat von der Tastatur, bis Sie <Strg-D> drücken:

Variable=`cat`

4.9.15.2. cut

Der Befehl cut liest von der Eingabe zeilenweise und schneidet einen bestimmten Teil davon heraus. Sie können wählen, zwischen

-c Bereich	Die Zeichen im Bereich
-f Bereich	Die Spalten im Bereich
-d ','	Legt das Trennzeichen für Spalten fest.

Die Bereichsangabe können Sie unterschiedlich schreiben:

3	Nur das dritte Zeichen oder die dritte Spalte
3-5	Die Zeichen oder Spalten drei bis fünf
-5	Die Zeichen vom Beginn bis fünf
1,3,5	Die Zeichen an Position 1, 3 und 5

Wenn Sie dem cut Befehl einen Dateinamen übergeben, liest er aus der Datei, ansonsten liest er von der Tastatur, bis Sie <Strg-D> drücken. Natürlich könenn Sie sowohl die Eingabe, als auch die Ausgabe mit den Pfeilen > < und dem Pipe-Zeichen | umleiten. Ein paar Beispiele:

cut -c 3-5
Hallihallo
<Strg-D>

Ergibt die Ausgabe:

```
lli
```

Angenommen, Sie haben eine Textdatei mit diesem Inhalt,

```
Marius
Mariechen
Liesa
Robert
```

und Sie geben dann

cut -c 1-3 test.txt

ein. Dann bekommen Sie diese Ausgabe:

```
Mar
Mar
Lie
Rob
```

Die Optionen -f und -d benutzt man meistens kombiniert:

cut -f 2,4 -d ','
Eins,Zwei,Drei,Vier,Fünf
<Strg-D>

Dies erzeugt die Ausgabe:

```
Zwei,Vier
```

Sie haben also die Felder 2 und 4 herausgefiltert. Beachten Sie, dass die Ausgabe das selbe Trennzeichen benutzt, wie die Eingabe.

4.9.15.3. sed

Sed ist ein sehr umfangreiches Programm, mit dem man Texte verändern kann. Es gibt eigene Bücher für sed, so umfangreich sind die Funktionen. Ich möchte Ihnen hier nur ein paar wenige Anwendungsbeispiele von sed zeigen, die ich häufig benutze.

Wie cat und cut liest auch sed wahlweise aus einer Textdatei oder von der Tastatur, bis Sie <Strg-D> drücken. Die Eingabe und die Ausgabe lässt sich wiederum umleiten.

Sed verändert die Quelldatei nicht, sondern zeigt nur deren Inhalt in veränderter Form an.

Angenommen, Ihre Textdatei (test.txt) hat diesen Inhalt:

```
Hallo
das
Wetter
ist
schön
und
ich
muss
lernen
```

So lesen Sie aus einer Textdatei nur bestimmte Zeilen aus, während Sie die übrigen Zeilen verwerfen:

sed '2,4p' --silent test.txt

```
das
Wetter
ist
```

2,4p bedeutet also: Gib nur die Zeilen 2 bis 4 aus. Wenn Sie nur eine Zeile haben wollen, entfällt das Komma im Befehl:

sed '2p' --silent test.txt

Der nächste Befehl gibt den ganzen Text aus, löscht aber die erste Zeile:

sed '1d' test.txt

Sed kann auch Wörter suchen und ersetzen:

sed 's/schön/schlecht/g' test.txt

Das Suchwort "schön" ist ein regulärer Ausdruck. Im Übernächsten Kapitel erkläre ich Ihnen, was genau reguläre Ausdrücke sind. Lesen Sie es durch - es lohnt sich!

4.9.15.4. grep

Der grep Befehl liest die Eingabe zeilenweise und gibt davon nur die Zeilen wieder aus, in denen ein gesuchtes Wort (genauer: regulärer Ausdruck) vorkommt. Die Eingabe ist wie gewohnt wahlweise eine Datei oder die Tastatur, sie lässt sich auch umleiten. Ebenso kann man die Ausgabe umleiten. Wenn Sie von der Tastatur eingeben, beenden Sie den Vorgang durch die Tastenkombination <Strg-D>.

Angenommen, die haben eine Textdatei mit diesem Inhalt (test.txt):

```
Im Winter kommt der Schnee.
Im Sommer ist es dafür zu Warm.
Ich singe gerne Lieder, ganz laut.
Wohin gehst Du?
Der Mann sieht merkwürdig aus.
```

Wenn Sie jetzt **grep 'der' test.txt** eingeben, erscheint diese Ausgabe:

```
Im Winter kommt der Schnee.
Ich singe gerne Lieder, ganz laut.
```

Die zweite Zeile erscheint deswegen, weil in "Lieder" das Wort "der" enthalten ist. Die Zeile "Der Mann ..." erscheint nicht, denn das D ist dort groß geschrieben.

Mit der Option -i sorgen Sie dafür, dass nicht zwischen großen und kleinen Buchstaben unterschieden wird:

grep -i 'der' test.txt

```
Im Winter kommt der Schnee.
Ich singe gerne Lieder, ganz laut.
Der Mann sieht merkwürdig aus.
```

Mit der Option -c zählen Sie einfach nur die Anzahl der gefunden Zeilen. Beispiel (test36.sh):

```
#!/bin/sh
Anzahl=`grep -c 'der' test.txt`
if [ $Anzahl -gt 2 ]; then
   echo "Es wurden mehr als 2 Zeilen gezählt, nämlich $Anzahl"
else
  echo "Es wurden weniger oder genau 2 Zeilen gezählt"
fi
```

Mit der Option -l lassen Sie sich nur die Dateinamen anzeigen, in denen das gesuchte Wort gefunden wurde. Sie können also auch mehrere Dateinamen als Argument angeben.

Mit der Option -v lassen Sie sich alle Zeilen anzeigen, in denen das gesucht Wort **nicht** vorkommt.

Was genau reguläre Ausdrücke sind, erfahren Sie im nächsten Kapitel.

4.9.15.5. Reguläre Ausdrücke

Reguläre Ausdrücke sind Suchbegriffe, nach denen in Textengesucht werden kann. Sie sind aber nicht nur auf einfache Wörter beschränkt. Reguläre Ausdrücke werden bei der Shell- Script- Programmierung oft zusammen mit den Befehlen sed und grep benutzt.

Ich möchte Ihnen einige nützliche reguläre Ausdrücke am Beispiel des grep Befehls zeigen, sie funktionieren aber auch genau so als Suchwort beim Befehl sed.

a-z, A-Z, 0-9	sucht nach dem entsprechenden Buchstaben
^	ist der Zeilenanfang
$	ist das Zeilenende
\<	ist ein Wortanfang
\>	ist ein Wortende
.	ist ein beliebiges Zeichen
[abc]	sucht nach einem Buchstaben a, b oder c
[a-z]	sucht nach einem Buchstaben von a bis z
a*	sucht nach keinem oder beliebig vielen a
a+	sucht nach mindestens einem oder mehr a (nur egrep)
|	ist eine Oder-Verknüpfung (nur egrep)
()	verwendet man zum Gruppieren (nur egrep).

Achtung: Die letzten drei Ausdrücke kennt nur der egrep Befehl, eine erweiterte Version von grep!

Wenn Sie nach einem Zeichen suchen wollen, dass in grep oder egrep eine besondere Funktion hat, müssen Sie ein \ voranstellen, z.B. sucht \. nach dem Punkt und nicht nach irgend ein beliebiges Zeichen.

Legen Sie sich die Textdatei test.txt an, mit diesem Inhalt an:

```
Im Winter kommt der Schnee.
Ich singe gerne Lieder, ganz laut.
Wohin gehst Du?
Der Mann sieht merkwürdig aus.
Mein Name ist Dr. Irene Meier.
Ihr Name ist Frau Dr.Dr. Liebig.
Brrrrrrrr, ist mir kalt.
Beim nächsten Ton ist es 08:23:56.
Bären sind liebe Tiere.
Bumm, das hat gekracht.
Brrrrumm, und noch einer.
```

Auf diese Textdatei wollen wir nun einige reguläre Ausdrücke anwenden.

grep 'i' test.txt

Zeigt alle Zeilen an, in denen ein I vorkommt. Aber

grep '^I' test.txt

zeigt nur die Zeilen an, die mit einem großen I beginnen. Der Unterschied ist, dass die Zeile mit " Frau Dr. Irene" nur vom ersten grep Befehl gefunden wird.

grep '\.$' test.txt

zeigt alle Zeilen an, die mit einem Punkt enden. Die Zeile mit dem Fragezeichen fällt also heraus.

grep '..:..:..' test.txt

zeigt alle Zeilen mit einer Uhrzeit an

grep 'er\>' test.txt

zeigt alle Zeilen an, in denen ein Wort vorkommt, dass mit "er" endet.

grep '\<....\>' test.txt

zeigt alle Zeilen an, in denen ein Wort mit vier Buchstaben steht.

grep 'Br*umm' test.txt

zeigt alle Zeilen an, in denen ein Wort mit B beginnt, dann kommen beliebig viele r (vielleicht auch keins) und dann umm. Also ist die Ausgabe:

```
Bumm, das hat gekracht.
Brrrrumm, und noch einer.
```

egrep 'Br+umm' test.txt

zeigt aber nur die Zeile mit "Brrrrumm" an, denn wegen dem + Zeichen wird nun mindestens ein r verlangt.

egrep '(Dr\.)+' test.txt

findet "Dr." und "Dr.Dr.". Dieses Beispiel demonstriert den Nutzen von Klammern. Da + bezieht sich jetzt nämlich nicht nur auf einen einzigen Buchstaben, sondern auf alles, was innerhalb der Klammern steht

egrep 'Meier|Liebig' test.txt

sucht nach dem Namen Meier oder Liebig.

grep '[TF][or]' test.txt

zeigt alle Zeilen an, in denen ein großes T oder ein großes F gefolgt von einem kleinen o oder r vorkommt.

4.9.15.6. Leerzeichen wegschneiden

Manchmal haben Sie in einer Variablen einen Text stehen, der mit unerwünschten Leerzeichen beginnt oder endet. Wie bekommen Sie diese weg?

Ganz einfach so:

```
a="        Hallo"
echo "$a"
b=`echo $a`
echo "$b"
```

Diese Befehle erzeugen die Ausgabe:

```
          Hallo
Hallo
```

Der erste echo Befehl demonstriert, dass in der Variable a der Text mitsamt den ungewollten Leerzeichen steht. Danach zweckentfremden wir den echo Befehl mit einer Ausgabeumleitung in die Variable b, um die Leerzeichen wegzubekommen. Der dritte echo Befehl beweist den Erfolg dieser Aktion.

Warum funktioniert das so?

Zur Erklärung zerlegen wir zuerst mal die dritte Befehlszeile in

b=`...`

und

echo $a

Interessant ist dabei echo $a. Hier habe ich ganz bewusst keine Anführungsstriche um die Variable "$a" geschrieben - wie man es normalerweise immer tun sollte. Dadurch bewirkt man, dass die Shell den Text in $a in einzelne Worte zerlegt und alle überflüssigen Leerzeichen wegschneidet. Genau das wollen wir ja ausnahmsweise so haben.

4.10. Awk Programmierung

Awk ist eine Ergänzung zur Shell Scriptsprache. Die Stärken von Awk liegen darin, Tabellarische Textdateien einzulesen und zu verarbeiten. Awk wird oft verwendet, um Textdateien zu konvertieren, Werte zu berechnen und Reports zu generieren.

Awk gibt es für fast jedes Betriebssystem. Es ist auf fast allen Unix Systemen standardmäßig installiert.

Unter Solaris verwenden Sie bitte die neue Version nawk. Die ältere Version kann nicht alles, was ich hier beschreibe und sie meldet Fehler meisten in nur schwer verständlicher Form.

Betrachten Sie diese Anleitung als Einstiegshilfe. Awk hat viele Features, die ein eigenes Buch Wert sind. Ich möchte Ihnen hier nur die wichtigsten zeigen.

Die Homepage von gnu-awk bietet ein umfangreiches englisches Buch im HTML Format kostenlos zum Download oder online Lesen an: http://www.gnu.org/software/gawk/gawk.html

4.10.1. Wie man awk startet

Awk benötigt immer ein Script und eine oder mehrere Quelldateien. Die folgenden drei Befehle sind gleichwertig:

awk -f Scriptdatei Textdatei

awk -f Scriptdatei < Textdatei

cat Textdatei | awk -f Scriptdatei

Mehrere Textdateien gibt man so an:

awk -f Scriptdatei Textdatei1 Textdatei2 Textdatei3 ...

Wenn das Script sehr klein ist, kann man es auch als Argument angeben, anstatt es in eine separate Script-Datei zu speichern:

awk 'Befehle' Textdatei

4.10.2. Erste awk Beispiele

Bevor ich Ihnen die awk Scriptsprache erkläre, möchte ich Ihnen ein paar einfache Beispiele zeigen. Da erste Beispiel liest eine Textdatei ein und gibt von jeder Zeile das zweite Wort aus:

awk '{print $2}' Textdatei

Wenn Sie mit unterschiedlichen Textdateien herumspielen, werden Sie bemerken, dass awk mehrfache Leerzeichen und Tabulatoren zusammenfasst. Awk entfernt auch führende Leerzeichen aus allen Zeilen. Wenn die Textdatei so aussieht:

```
Stefan          Frings
Michaela Meier
   Gundula Geier
```

kommt hinten dies heraus:

```
Frings
Meier
Geier
```

Das zweite Beispiel gibt von einer Textdatei alle Zeilen aus, die das Wort Hallo beinhalten.

awk '/Hallo/ {print $0}' Textdatei

Das dritte Beispiel gibt die nur dritte Zeile der Textdatei aus, alle anderen werden unterdrückt:

awk 'NR==3 {print $0}' Textdatei

Die Aufgaben aus diesen einfachen Beispielen hätte man auch mit anderen Befehlen lösen können. Besonders, wenn Aufgaben komplexer werden, wird awk allerdings eindeutig die bessere Wahl sein.

4.10.3. Grundsätzliche Syntax von awk Scripten

Awk liest Textdateien zeilenweise ein und führt für jede Zeile eine Liste von Befehlen aus. Das ganze awk Script wird für jede einzelne Textzeile wiederholt durchlaufen. Jeder Befehl besteht aus zwei Teilen, der Bedingung und der Aktion.

Bedingung {Aktion}

Die Bedingung legt fest, welche Textzeilen durch die Aktion betroffen sein sollen. Ich möchte auf die Beispiele des vorherigen Kapitels zurück kommen:

Im Beispiel 1 wurde keine Bedingung angegeben. Die Aktion **print $2** betrifft also alle Zeilen der Textdatei.

Im Beispiel 2 wurde als Bedingung **/Hallo/** angegeben. Das ist ein Regulärer Ausdruck (wie bei grep), der besagt, dass alle Zeilen mit dem Wort Hallo betroffen sind.

Im Beispiel 3 wurde als Bedingung **NR==3** angegeben. Es ist nur die Zeile betroffen, deren Nummer==3 ist. NR ist eine Variable, die für die Zeilennummer steht.

Erstellen Sie nun mit Ihrem bevorzugten Texteditor ein awk Script. Es soll test1.awk heißen und so aussehen:

```
NR==1 { print "Die erste Zeile ist:" $0 }

NR>1 { print "Danach kommt:" $0;
       anzahl=anzahl+1;
     }

/Hallo/ { print "Hallo ebenfalls.";
          hallo=hallo+1;
        }

END { print "Nach Zeile 1 kamen " anzahl " Zeilen.";
      print "Hallo wurde " hallo " mal gezählt.";
    }
```

Dieses Script hat vier bedingte Aktionen. Sie sehen hier, wie man mehrere Aktionen in eine Scriptdatei schreibt und Sie sehen auch, dass eine Aktion durchaus aus mehr als nur einem einzelnen Befehl bestehen kann.

Die erste Aktion gilt nur für die erste Textzeile. Sie gibt einen Kommentar aus, und dahinter die ganze Zeile.

Die zweite Aktion gilt für alle Zeilen, außer der ersten. NR bedeutet Record-Number oder einfacher gesagt Zeilennummer. Auch diese Aktion gibt einen Kommentar aus. Außerdem zählt sie die Anzahl der Vorkommnisse.

Die dritte Aktion gilt nur für die Zeilen, in denen Hallo steht. Anstatt sie erneut auszugeben, gibt diese Aktion einfach nur "Hallo ebenfalls." aus und zählt die Vorkommnisse.

Die vierte Aktion wird nur ganz zum Schluss ausgeführt, wenn die Textdatei bis zum Ende eingelesen wurde. Dafür sorgt die Bedingung END. Zum Schluss werden die beiden Zählergebnisse ausgegeben.

Wenn Sie Aktionen vor der allerersten Textzeile ausführen möchten, verwenden Sie die Bedingung BEGIN.

Erstellen Sie nun eine Textdatei mit mehreren Zeilen. Wenigstens eine soll das Wort Hallo beinhalten. Führen Sie dann das Script aus, um es zu testen:

awk -f test1.awk Textdatei

Sie können auch mehrere Textdateien nacheinander abarbeiten. In diesem Fall wird die Bedingung END nur einmal ganz am Ende wirksam. Auch die Bedingung BEGIN würde nur einmal wirksam werden.

4.10.4. Feldtrenner

Awk trennt Textzeilen normalerweise anhand von Leerzeichen. Mehrfache Leerzeichen werden als eins zusammengefasst. Tabulatoren sind mit Leerzeichen identisch. Beispiel:

Sie haben die Textdatei test1.txt mit diesem Inhalt:

```
Stefan Frings 12 10 74 ,Programmiert SMS Server Tools
Bill Gates 23 05 52 ,War der reichste Mann der Welt
Golden Gate 01 24 65 ,Ist eine Brücke
```

Angenommen, Sie wollen nur die Namen haben, dann schreiben Sie dieses Script:

```
{print $1,$2}
```

Awk wird jede Zeile einlesen und in Wörter zerlegen. Der Print Befehl gibt dann die ersten beiden Wörter aus. Das Komma beim Print Befehl legt fest, dass zwischen die beiden Wörter ein Trennzeichen ausgegeben werden soll. Das Trennzeichen ist ein Leerzeichen, was Sie sicher nicht überrascht hat. Die Ausgabe sieht so aus:

```
Stefan Frings
Bill Gates
Golden Gate
```

Ohne dieses Trennzeichen erscheinen die Namen direkt hintereinander, also so:

```
{print $1 $2}

StefanFrings
BillGates
GoldenGate
```

Sie können auch Texte dazwischen einfügen:

```
{print "Vorname: " $1 " Nachname: " $2}


Vorname: Stefan Nachname: Frings
Vorname: Bill Nachname: Gates
Vorname: Golden Nachname: Gate
```

Das Komma fügt normalerweise ein Leerzeichen ein. Sie können auch andere Zeichen als Standard-Trennzeichen festlegen:

```
BEGIN {OFS="++"}
{print $1,$2}
```

Zu Beginn legt dieses Script "++" als Trennzeichen fest. OFS bedeutet Output-Field-Separator. Die Ausgabe sieht dann so aus:

```
Stefan++Frings
Bill++Gates
Golden++Gate
```

Sie haben nun gelernt, wie Trennzeichen in Ausgaben verändert werden. Doch auch für die Eingabe können Trennzeichen verändert werden. Awk ist hier nicht nur auf Leerzeichen und Tabulatoren beschränkt.

Verwenden Sie die obige Testdatei test1.txt mit diesem Script:

```
BEGIN {FS=","}
{print $2}
```

Sie erhalten als Ausgabe das Zweite Feld hinter dem Komma:

```
Programmiert SMS Server Tools
War der reichste Mann der Welt
Ist eine Brücke
```

Leerzeichen werden also nun nicht mehr als Trennzeichen betrachtet, sondern nur noch Kommas. Im Fall von Eingabe-Trennzeichen können Sie nicht nur einzelne Zeichen benutzen, sondern richtige reguläre Ausdrücke, wie bei grep.

```
BEGIN {FS="[ ,]"}
{print $1,$2,$7}
```

Hier werden sowohl Leerzeichen als auch Kommas als Trennzeichen verwendet. Die Zeichenfolge " ," wird als doppeltes Trennzeichen betrachtet, weil sie zweimal auf den regulären Ausdruck passt. Ergebnis:

```
Stefan Frings Programmiert
Bill Gates War
Golden Gate Ist
```

Wenn die Zeichenfolge nur als ein Trennzeichen betrachtet werden soll, muß lediglich der Reguläre Ausdruck verändert werden:

```
BEGIN {FS="[ ,]+"}
{print $1,$2,$6}
```

Der Kommentar beginnt mit Feld $6, weil " ," nicht mehr als doppeltes Trennzeichen gezählt wird. Schauen Sie im Kapitel über grep nach, um mehr über reguläre Ausdrücke zu erfahren.

4.10.5. Eingebaute Variablen

Awk kennt ein paar eingebaute Variablen, die Sie verwenden können. Sie haben bereits OFS und FS kennen gelernt.

OFS	Ausgabe-Feldtrenner. Print fügt dieses Zeichenfolge anstelle von Kommas ein. Vorgabewert ist ein Leerzeichen.
FS	Feldtrenner. Zeilen werden mit diesem regulären Ausdruck in Felder zerlegt. Vorgabewert ist ein oder mehrere Leerzeichen oder Tabulatoren.
ORS	Ausgabe Datensatztrenner. Print schließt jede Zeile mit diesem Zeichen ab. Vorgabewert ist ein Zeilenumbruch.
RS	Datensatztrenner. Eingabezeilen werden durch dieses Zeichen getrennt. Vorgabewert ist ein Zeilenumbruch.
OFMT	Ausgabeformat für Nummern.
NF	Anzahl von Feldern in der aktuellen Zeile.
NR	Nummer der Zeile, bezogen auf den Anfang der ersten Datei.
FNR	Nummer der Zeile bezogen auf den Anfang der aktuellen Datei.
FILENAME	Name der aktuellen Eingabedatei.
RSTART	Anfang einer Zeichenkette, gehört zum Befehl match().
RLENGTH	Länge einer Zeichenkette, gehört zum Befehl match().
$0	Die aktuelle Zeile.
$1	Das erste Feld der aktuellen Zeile.
$2	Das zweite Feld der aktuellen Zeile.
$3	Das dritte Feld der aktuellen Zeile.

4.10.6. Variablen an der Kommandozeile übergeben

Manchmal möchten Sie Variablen an der Kommandozeile übergeben. Das kommt besonders oft vor, wenn Sie awk innerhalb eines Shell-Scriptes benutzen. Ein simples Beispiel:

```
#!/bin/sh
echo "Wir haben"
awk -v fruechte=36 -v personen=12 'BEGIN {print fruechte/personen}'
echo "Äpfel pro Kind."
```

oder so

```
#!/bin/sh
aepfel=36
kinder=12
echo "Wir haben"
awk -v fruechte=$aepfel -v personen=$kinder 'BEGIN {print fruechte/personen}'
echo "Äpfel pro Kind."
```

oder auch so

```
#!/bin/sh
aepfel=36
kinder=12
ergebnis=`awk -v fruechte=$aepfel -v personen=$kinder 'BEGIN {print fruechte/personen}'`
echo "Wir haben $ergebnis Äpfel pro Kind."
```

All diese Beispiele erzeugen (fast) die gleiche Ausgabe. Das letzte Beispiel hat nur einen Echo Befehl und erzeugt daher eine einzeilige Ausgabe.

Sie sehen, wie man mit Hilfe der Option -v Variablen von außen setzen kann, um sie dann innerhalb des awk Scriptes zu benutzen.

Sie haben sicher bemerkt, dass hier gar keine Eingabedatei eingelesen wird. Hinter der BEGIN Bedingung steht einfach die Mathematische Berechnung. Da es keine Bedingung gibt, die auf Textzeilen angewendet wird, darf ausnahmsweise auf die Angabe einer Textdatei verzichtet werden.

4.10.7. Operatoren

Die vorherigen Kapitel haben Ihnen bereits mathematische Additionen und Divisionen vorgeführt. Awk kennt die folgenden Operatoren:

+	Addition
-	Subtraktion
*	Multiplikation
/	Division
%	Modulus
^	Exponentialfunktion
++	Erhöhen um 1
--	Verringern um 1

Awk kennt diese Zuweisungen:

=	Zuweisung
+=	Hinzufügen zu
-=	Abziehen von
*=	Multiplizieren um
/=	Dividieren durch
%=	Modulus durch
^=	Exponentialfunktion

Beispiele:

a=3 Ergibt 3

a+=5 Ergibt 8, ist identisch mit a=a+5

b=a+6 Ergibt 14

b++ Ergibt 15, ist identisch mit b=b+1

b/=3 Ergibt 5, ist identisch mit b=b/3

Awk kennt die folgenden Vergleichsoperatoren:

==	gleich
!=	nicht gleich
>	größer als
<	kleiner als
>=	größer oder gleich
<=	kleiner oder gleich
~	beinhaltet regulären Ausdruck /RE/
!~	beinhaltet regulären Ausdruck /RE/ nicht

Beispiele:

if ($1=="Stefan") Wenn das erste Wort Stefan ist, dann ...

if ($1>5) Wenn die erste Zahl größer als 5 ist, dann ...

if ($1~/fan/) Wenn das erste Wort "fan" enthält, dann ...

Logische Operatoren setzte man in Awk ein, um mehrere Vergleiche zu kombinieren:

&&	Und
||	Oder
!	Nicht

Beispiel:

if (a==3 && b==4) Wenn a==3 und b==4 ist, dann ...

4.10.8. Bedingungen mit if

Mit dem Schlüsselwort if können Sie einen oder mehrere Befehle Bedingt ausführen. If ist eine Alternative zu den Bedingungen aus Kapitel 4.10.3, die flexibler verwendbar ist.

```
if (Bedingung)
  Befehl;
else
  Befehl;
```

oder auch:

```
if (Bedingung) {
  Befehle;
  Befehle;
  Befehle;
} else {
  Befehle;
  Befehle;
  Befehle;
}
```

Wenn Sie mehr als einen Befehl bedingt ausführen möchten, schreiben Sie geschweifte Klammern davor und dahinter. Sie bilden damit Blöcke. Der else-Abschnitt kann weggelassen werden.

Beispiel:

if (a==3)

```
  print "Die Variable a ist jetzt 3";
```

4.10.9. Schleifen mit while

Schleifen benutzt man, wenn ein Befehl oder ein Block von Befehlen mehrmals ausgeführt werden soll. Die while Schleife kann auf zwei Unterschiedliche Arten verwendet werden:

```
while (Bedingung)
  Befehl;


do
  Befehl;
while (Bedingung)
```

Die beiden Varianten Unterscheiden sich nur darin, ob die Bedingung bei jeder Wiederholung vor oder nach dem befehl geprüft wird. Beispiel:

```
a=0
while (a<10) {
  print "Die Variable a ist jetzt" a;
  a++;
}
```

Diese Schleife zählt von 0 hoch bis 9.

4.10.10. Schleifen mit for

Von der Programmiersprache C hat awk sich die for Schleife abgeguckt. Sie ist hauptsächlich für Schleifen gedacht, bei denen ein Wert hoch oder runter gezählt wird.

```
for (Vorbereitung; Bedingung; Schritt)
  Befehl;
```

Das Beispiel aus dem Vorherigen Kapitel könnte auch so aussehen:

```
for (a=0; a<10; a++)
  print "Die Variable a ist jetzt" a;
```

In Zusammenhang mit Arrays gibt es noch eine weitere Form der for-Schleife:

```
for (Indexvariable in Arrayname)
  Befehl;
```

Auf diese Variante gehe ich in einem späteren Kapitel ein.

4.10.11. Break und continue

Mit dem Befehl break können Sie eine Schleife abbrechen.

```
a=0;
while (a<10) {
  print "Die Variable a ist jetzt" a;
  a++;
  if (a>5)
    break;
}
```

Diese Schleife läuft nicht bis zur 9 durch, sondern nur bis zur 5, weil dann der break Befehl ausgeführt wird. Break wird verwendet, um schleifen frühzeitig abzubrechen.

Continue bricht den aktuellen Schleifendurchlauf ab und führt die nächste Schleifenwiederholung durch.

```
a=0;
while (a<10) {
  print "Die Variable a ist jetzt" a;
  a++;
  if (a>5)
    continue;
  print "a ist noch nicht größer als 5";
}
```

4.10.12. Next und exit

Im Gegensatz zu break und continue beeinflussen diese beiden Befehle den Ablauf des gesamten awk Scriptes.

Next stoppt die Bearbeitung der aktuellen Textzeile. Awk liest die nächste Textzeile ein und führt die Befehle von oben nach unten für diese neue Zeile aus.

Exit bricht das awk Script sofort ab. Sie können hinter exit eine Zahl angeben, welche als Return-Wert an die Shell übergeben wird. Shell-Scripte können diesen Wert dann auswerten, falls gewünscht.

Exit oder exit 0 wird üblicherweise für erfolgreiches Programmende verwendet, während andere Werte einen Fehler bedeuten. Das ist zwar keine Vorschrift, aber es hat sich so eingebürgert.

4.10.13. Sonderzeichen in print

Sie haben mit print bereits Texte und variablen ausgegeben. Print kennt die noch zwei Sonderzeichen:

\n	Zeilenumbruch
\t	Tabulator
\00xx	Beliebiges Zeichen als Oktalzahl

Print kann seine Ausgaben in Dateien schreiben, anstatt auf die Standard-Ausgabe (welche meist der Bildschirm ist).

```
print "Hallo" > "Dateiname";
print "Hallo" >> "Dateiname";
```

Die zweite Version hängt an eine vorhandene Datei an, während die erste Version eine neue Datei anlegt und ggf. vorhandene Datei löscht.

Print kann außerdem Unterprogramme starten und seine Ausgabe an dieses Programm übergeben.

```
print "Stefan\nMarkus\nJessica\nOlivia" | "sort";
```

Der obige Befehl schreibt vier Namen und übergibt diese an den Unix Befehl sort. Sort sortiert die Namen alphabetisch.

4.10.14. Printf und sprintf

Printf verwendet man, um eine Zeichenkette ohne abschließenden Zeilenumbruch auszugeben:

```
printf("Zeichenkette");
```

Man kann printf auch verwenden, um Variablen in einem besonderen Format auszugeben:

```
printf("Zeichenkette",Variable,Variable,Variable,...);
```

Innerhalb der Zeichenkette müssen Platzhalter stehen, an deren Stelle dann die Variablem eingefügt werden. Es gibt diese Platzhalter:

%d	Eine Integer-Zahl
%f	Eine Fließkommazahl
%c	Ein Zeichen
%s	Eine Zeichenkette

Beispiel:

printf("%f Euro",12.5);

Man kann die die Länge der Felder festlegen, wo die Variablen eingefügt werden. Die Werte werden dann rechtsbündig ausgegeben:

printf("%6f Euro",12.5);

Auch linksbündige Ausgabe ist möglich:

printf("%-6f Euro",12.5);

Ein etwas komplexers Beispiel:

printf("%-20s %4.2f Euro\n","Inkjet paper",12.5);

Hier wird der Titel "Inkjet paper" linksbündig ausgegeben und auf 20 Zeichen Länge aufgefüllt. Der Preis wird rechtsbündig auf 4 Stellen aufgefüllt ausgegeben und mit genau 2 Nachkommastellen. Am Ende kommt noch ein Zeilenumbruch.

Sprintf funktioniert wie printf, nur wird die Ausgabe in eine Variable geschrieben, anstatt auf den Bildschirm:

variable=sprintf("%f Euro",12.5);

4.10.15. Arrays

Arrays sind Variablen, die viele Werte beinhalten können. Jeder Wert innerhalb des Arrays hat einen Namen oder eine Nummer zuzüglich dem Wert selbst.

Beispiele:

```
fruechte["apefel"]=36;
fruechte["birnen"]=20;
fruechte["kirschen"]=340;
...
print "Wir haben " fruechte["aepfel"] " Äpfel";
print "Wir haben " fruechte["birnen"] " Birnen";
print "Wir haben " fruechte["kirschen"] " Kirschen";
...
```

Einfacher geht es mit der for Schleife:

```
fruechte["apefel"]=36;
fruechte["birnen"]=20;
fruechte["kirschen"]=340;
...
for (name in fruechte)
  print "Wir haben " fruechte[name] " name;
```

Arrays kann man sehr gut gebrauchen, wenn man die Anzahl von Fehlercodes in einer Protokolldatei zählen möchte. Angenommen, die Protokolldatei hat dieses Format:

```
05.04.2005 12:38 Error 12
05.04.2005 12:39 Trying to read configuration file
05.04.2005 12:55 Error 18
05.04.2005 12:58 Starting Internet communication
05.04.2005 12:59 Error 17
```

Dann können Sie mit dem folgenden Awk Script die Häufigkeit der einzelnen Fehlercodes zählen:

```
/Error/ {errors[$4]++}

END {for (code in errors)
       print "Fehlercode " code " wurde " errors[code] " mal gezählt.";
    }
```

4.10.16. Eingebaute Funktionen

Awk hat viele eingebaute Funktionen. Die gebräuchlichsten numerischen Funktionen sind diese:

int(x)	Rundet die Zahl x ab
log(x)	Natürlicher Logarithmus von x
exp(x)	Natürlicher Exponent von x
sin(x)	Sinus von x in radians.
cos(x)	Cosinus von x in radians.
rand()	Zufallszahl zwischen 0 und 1
srand()	Initialisiert den Zufallszahlengenerator. Dieser Befehl sollte einmal in der BEGIN Bedingung jedes Programmes verwendet werden, das rand() benutzt.

Viel häufiger werden Sie Zeichenketten-Funktionen verwenden:

tolower(s)	Gibt den Text s in Kleinbuchstaben aus.
toupper(s)	Gibt den Text s in Großbuchstaben aus.
length(s)	Länge der Zeichenkette.
substr(s,start,laenge)	Gibt einen Teil der Zeichenkette s zurück. Der Teil beginnt bei start und hat eine bestimmte Länge. Wenn man die Länge nicht angibt, wird der gesamte Rest zurückgegeben.
match(s,r)	Vergleicht die Zeichenkette s mit dem regulären Ausdruck r. Es werden die Variablen RSTART und RLENGTH gesetzt. Wenn der Ausdruck nicht gefunden wurde, ist RSTART 0. Match gibt die Position oder 0 als Ergebnis zurück.
sub(r,s,ziel)	Sucht im Ziel nach dem regulären Ausdruck r und ersetzt den ersten gefundenen Abschnitt durch die Zeichenkette s. Wenn kein Ziel angegeben wird, wird $0 verwendet.
gsub(r,s,ziel)	Wie sub, ersetzt aber alle gefunden Abschnitte.
index(s,suchwort)	Sucht ein Wort in der Zeichenkette s und gibt die gefundene Position zurück. Das Ergebnis ist 0, wenn das Wort nicht gefunden wurde.
split(s,array,separator)	Zerlegt die Zeichenkette s in einzelne Felder. Jedes Feld wird ein Element im Array. Wenn kein Separator angegeben ist, wird FS verwendet.

Achtung: Reguläre Ausdrücke dürfen zur besseren Kennzeichnung auch zwischen zwei Schrägstriche geschrieben werden, anstatt zwischen doppelte Anführungsstriche.

Funktionen für Ein/Ausgabe:

close(dateiname)	Schließt eine Datei
getline variable < dateiname	Liest eine Zeile aus der angegebenen Datei. Wenn man keinen Dateinamen angibt, wird die aktuelle Eingabedatei benutzt. Wenn keine Variable angegeben wird, dann wird $0 verwendet.
Befehl | getline Variable	Führt einen Befehl aus und schreibt dessen Ausgabe in die Variable. Wenn keine Variable angegeben wird, dann wird $0 verwendet.
system	Führt einen externen Unix Befehl aus und gibt den Status zurück.

Sonstige Funktionen:

delete array[element]	Löscht ein Element aus einem Array

Beispiele:

```
text="Hallo Bruderchen";
gsub(/Bruder/,"Brüder",text);
```

Ergibt: "Hallo Brüderchen"

```
text="GemiSChT gESchrieBEn";
```

```
print tolower(text);
```

Ergibt "gemischt geschrieben";

```
text="Fehlercode [45]";
match(text,/\[[0-9]+\]/);
if (RSTART>0)
  print substr(text,RSTART+1,RLENGTH-2);
```

Ergibt: 45

Match sucht nach einer Zahl mit mindestens einer Ziffer, die innerhalb von eckigen Klammern steht. Die Position und Länge wird in RSTART und RLENGTH gespeichert.

Der Reguläre Ausdruck setzt sich so zusammen [0-9] sucht nach einer Ziffer und \[sucht nach einer Klammer. Das + bedeutet soviel, wie "mindestens 1 mal".

Mit substr wird dann nur die Zahl selbst ohne die Klammern herausgenommen und dann mit print angezeigt.

4.10.17. Eigene Funktionen

Eigene Funktionen können am Anfang des Scriptes deklariert werden:

```
function verdoppeln(a) {
  return a*2;
}

{print verdoppeln($1)}
```

Dieses Script liest eine Textdatei ein, in der jede Zeile mit einer Zahl beginnen muss. Diese Zahlen werden verdoppelt und dann angezeigt.

4.10.18. Awk Beispiel für SMS Anwendungen

Wie sie sehen, kann Awk einiges, was die Shell nicht kann. Die meisten Funktionen von Awk werden Sie vermutlich nicht brauchen. Nun stellt sich die Frage, warum ich Awk überhaupt erkläre.

Sie können mit Awk manche einfache Aufgaben elegant lösen, was mit der Shell nur sehr umständlich möglich wäre.

An dieser Stelle greife ich etwas vor. Auch wenn Sie die Funktionsweise der SMS Server Tools nicht nicht ganz kennen, möchte ich Ihnen ein sehr konkretes Anwendungsbeispiel geben.

Angenommen, Ihre Kunden können per SMS einen Witze bestellen, indem Sie das Wort "Witz" und eine Artikelnummer per SMS an Ihren Server senden. Sie empfangen nun viele SMS Nachrichten, von denen sicher einige keine gültige Bestellung erhalten. Ihr Programm soll folgende Aufgaben erfüllen:

- Verwerfen aller ungültigen Bestellungen
- Extrahieren der 5-Stelligen Artikelnummer
- Versenden des Artikels (Witz).

Sie könnten nun grep verwenden, um zu prüfen, ob die empfangene SMS das Wort "Witz" enthält und ob eine Artikelnummer enthalten ist. Doch grep kann Ihnen nicht die Artikelnummer extrahieren. Auch der Befehl cut ist wenig hilfreich, denn Sie wissen nicht, wo die Artikelnummer steht. Der Kunde könnte sie an den Anfang schreiben, oder an das Ende. Er könnte Leerzeichen dazwischen schreiben und er könnte auch andere überflüssige Wörter in die SMS schreiben. Beispiel:

"Hallo, ich möchte den Witz Nummer 12345 bekommen".

Prinzipiell ist das eine gültige Bestellung, denn sie enthält sowohl das Wort "Witz" als auch die Artikelnummer. Ich möchte Ihnen nun zeigen, wie sie ein Shell-Script und Awk kombinieren, um solche Bestellungen zu verarbeiten.

Ihre Ware (die Witze) liegt in Form von Textdateien vor, etwa so:

/var/spool/sms/witze/12345.txt
/var/spool/sms/witze/23456.txt
/var/spool/sms/witze/98765.txt

Empfangene SMS werden von dem SMS Server Tools als Dateien abgelegt, die so aussehen:

```
From: +491721234567

Hallo, ich möchte den Witz Nummer 12345 bekommen
```

Wenn Die SMS Server Tools eine SMS Empfangen, speichern Sie die Nachricht ab und rufen dann ein Eventhandler Script auf. $1 ist dann immer das Wort "RECEIVED" und $2 ist der Dateiname, unter dem die SMS gespeichert wurde.

Ihr Eventhandler Script sieht so aus:

```
#!/bin/sh
if [ "$1" != "RECEIVED" ]; then
  exit
fi

from=`formail -zx From: < $2`
text=`formail -I "" <$2 | sed -e"1d"`

if echo "$text" | grep -i "witz"; then
  smsfile=`mktemp /var/spool/sms/outgoing/send_XXXXXX`
  echo "To: $from" >> $smsfile
  echo "" >> $smsfile
  number=`echo "$text" | awk '

  {
    if (match($0,/[0-9][0-9][0-9][0-9][0-9]/)
      print substr($0,RSTART,RLENGTH)
  }

'`
  if [ -f "/var/spool/sms/witze/$number.txt" ]; then
    cat "/var/spool/sms/witze/$number.txt" >> $smsfile
  else
    echo "Sorry, falsche Artikelnummer" >> $smsfile
  fi
fi
```

Falls Sie Solaris benutzen, ersetzen Sie awk bitte wieder durch nawk.

Das Script extrahiert mit Hilfe von formail zuerst die Telefonnummer des Absenders und den Nachrichtentext. Dann prüft es mit grep, ob das Schlüsselwort "Witz" enthalten ist. Der Parameter -i sorgt dafür, dass Groß/Kleinschreibung egal ist. Nur, wenn das Wort Witz gefunden wurde, wird die Nachricht weiter bearbeitet.

Mit mktemp wird eine neue SMS-Datei erzeugt. Mktemp ersetzt die xxxxxx durch eine Zufallszahl. Mit echo Befehlen wird die Telefonnummer des Kunden und eine Leerzeile in die SMS-Datei geschrieben.

Anschließend wird die Artikelnummer mit awk aus der Bestellung extrahiert. Der nächste if-Befehl prüft, ob ein entsprechender Artikel (Textdatei) existiert. Anschließend wird entweder der bestellte Text oder eine Fehlermeldung in die SMS-Datei geschrieben.

Die SMS Server Tools werden die Datei automatisch senden.

Ich möchte die Zeile mit dem awk Befehl genauer erklären:

```
number=`echo "$text" | awk '{if (match($0,/[0-9][0-9][0-9][0-9][0-9]/) print substr($0,RSTART,RLENGTH)}'`
```

```
number=`...`
```

Sorgt dafür, dass das Ergebnis des Befehls in der Variable $number gespeichert wird. So können wir die Artikelnummer später sehr einfach in den Dateinamen einfügen.

```
echo "$text" | awk ...
```

Der Echo Befehl sendet den Text der Bestellung nach awk. Wir geben awk also keine Eingabedatei, sondern verwenden stattdessen eine Pipe. Das Prinzip habe ich in Kapitel 4.10.1 gezeigt.

```
if (match($0,/[0-9][0-9][0-9][0-9][0-9]/) ...
```

Der match Befehl sucht nach einem Regulären Ausdruck. In diesem Fall wird nach genau 5 Ziffern im Bereich 0-9 gesucht. Nur, wenn eine 5-Stellige Zahl gefunden wurde, wir der nächste print Befehl ausgeführt.

```
print substr($0,RSTART,RLENGTH)
```

Hier wird dann genau die gefundene Zahl ausgegeben. Der match() Befehl hat die Position und die Länge in den Variablen RSTART und RLENGTH gespeichert, so dass wir mit dem Befehl substr() den entsprechenden Teil aus der ganzen Textzeile extrahieren können.

5. Die SMS Server Tools

Die SMS Server Tools sind ein Programmpaket zum Senden und Empfangen von SMS.

Wie der Name bereits sagt, dient das Programmpaket weniger als buntes User Interface, sondern es ist eher zum Aufbau von Servern gedacht, die automatisch funktionieren.

Das Programmpaket wird von dem Autor dieses Buches entwickelt. Er bekommt dabei Unterstützung von zahlreichen Hobbyprogrammierern und kommerziellen Anwendern.

Es wird vergleichsweise schnell weiter entwickelt. Wenn Sie an der jeweils neuesten Version interessiert sind, sollten Sie meine Homepage besuchen:

http://www.meinemullemaus.de

Neue Versionen kündige ich immer unter

http://www.freshmeat.net

an. Dort können Sie ihre eMail Adresse in eine Verteiler-Liste eintragen. Sie erhalten dann bei der Veröffentlichung jeder neuen Version eine eMail. Ich eMail Adresse kann ich selbst nicht einsehen, sie wird von den Betreibern von www.freshmeat.net vertraulich behandelt. Ich aknn lediglich nachschauen, wieviele Einträge in der Liste stehen.

Das Programm ist bereits seit einiger Zeit in Debian Linux enthalten.

Dieses Kapitel ist im Grunde eine Übersetzung der englischen Original-Dokumentation, jedoch mit einigen zusätzlichen Details.

5.1. Überblick

Diese Einleitung gibt Ihnen eine Übersicht, was die SMS Server Tools können und wie sie funktionieren.

Eine Anmerkung zu Abkürzungen:

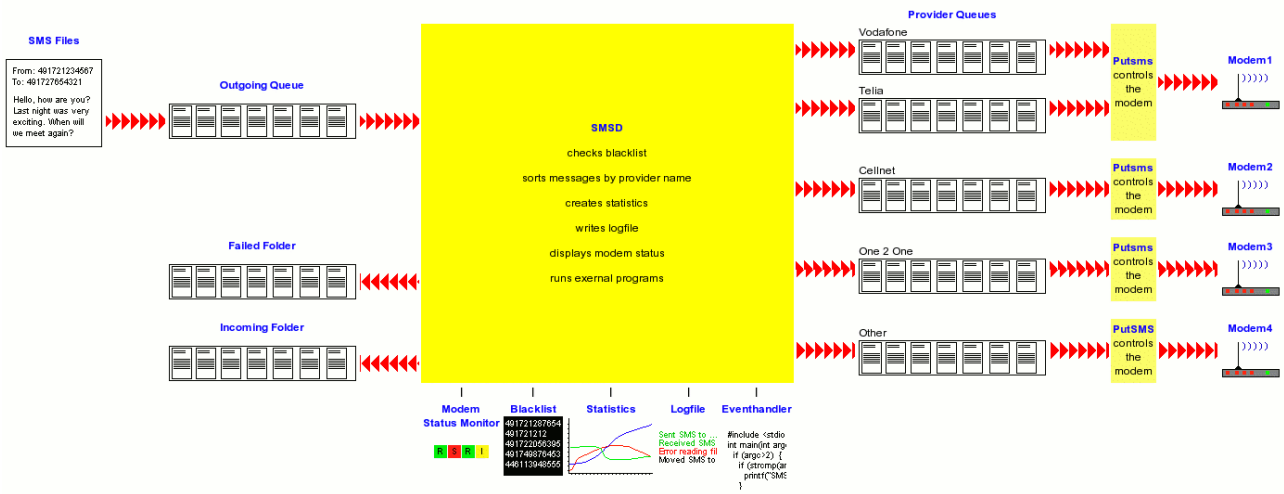
SMS bedeutet Short Message Service und bezeichnet den gesamten Dienst. Man kann also keine SMS versenden, dennoch wird die Abkürzung von den meisten Handybesitzern falsch verwendet - selbst die Werbung gebraucht diese Abkürzung falsch.

SM bedeutet Short Message. Man sendet also keine SMS, sondern eine SM. Viele Leute vermeiden diese Abkürzung, weil SM auch Sado-Masochismus bedeutet.

SMSC bedeutet Short Message Service Center. Das ist eine SMS Zentrale, die für die Zwischenspeicherung und Verteilung von SM beim Netzbetreiber zuständig ist. So wie jedes Telefongespräch durch eine Vermittlungsstelle geht, und jeder Brief durch die Post, geht jede SM durch wenigstens ein SMSC.

Unter SIM Karte versteht man die kleine Chipkarte, die Sie in jedes Handy und jedes GSM Modem stecken müssen, um das sich beim Netzbetreiber zu identifizieren. Ohne SIM Karte verweigern die Geräte jede Funktion. Handies und GSM Modems speichern nicht nur Telefonbücher auf die SIM Karte, sondern auch SMS. Auf der SIM Karte ist auch die eigene Rufnummer (MSISDN) gespeichert.

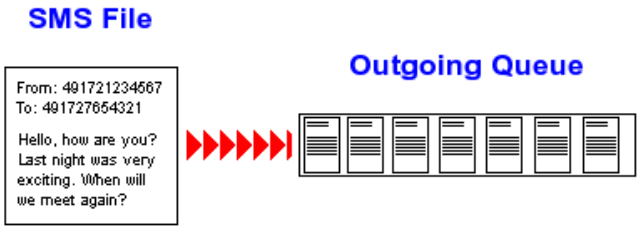
Das folgende Bild zeigt die wichtigsten Bestandteile der SMS Server Tools und wie diese untereinander verknüpft sind.



Das Bild zeigt, wie die SMS Server Tools SM senden. In diesem Fall werden fünf britische Telefon-Netzbetreiber unterstützt und es werden vier GSM Modems zum Versand und Empfang benutzt.

Eine Erklärung folgt ab der nächsten Seite.

Das Senden von SMS beginnt, indem man ein SMS File anlegt.



In dieser Textdatei steht wenigstens die Telefonnummer des Empfängers und der Nachrichtentext.

Sie können auch den Namen oder die Telefonnummer des Absenders in die Datei schreiben. Diese tauchen dann später in Protokolldateien (Logfiles) wieder auf.

Nachdem Sie die Datei erstellt haben, verschieben Sie sie in die ausgehende Warteschlange (Outgoing Queue). Das is ein einfaches Verzeichnis auf der Festplatte.

Sie können die Datei auch direkt in diesem Verzeichnis erstellen, wenn das schnell geht. Das Hauptprogramm sms untersucht die Dateien und prüft, ob sie innerhalb einer Sekunde grösser werden, oder nicht. Wenn eine Datei innerhalb einer Sekunde ihre Grösse beibehalten hat, geht sms davon aus, dass sie fertig ist und jetzt gesendet werden kann.

Falls Sie Dateien direkt in der ausgehenden Warteschlange erstellen, stellen Sie sicher, dass dies zügig vonstatten geht, damit die Nachricht nicht irrtümlich zu früh gesendet wird, bevor sie fertig erstellt wurde.

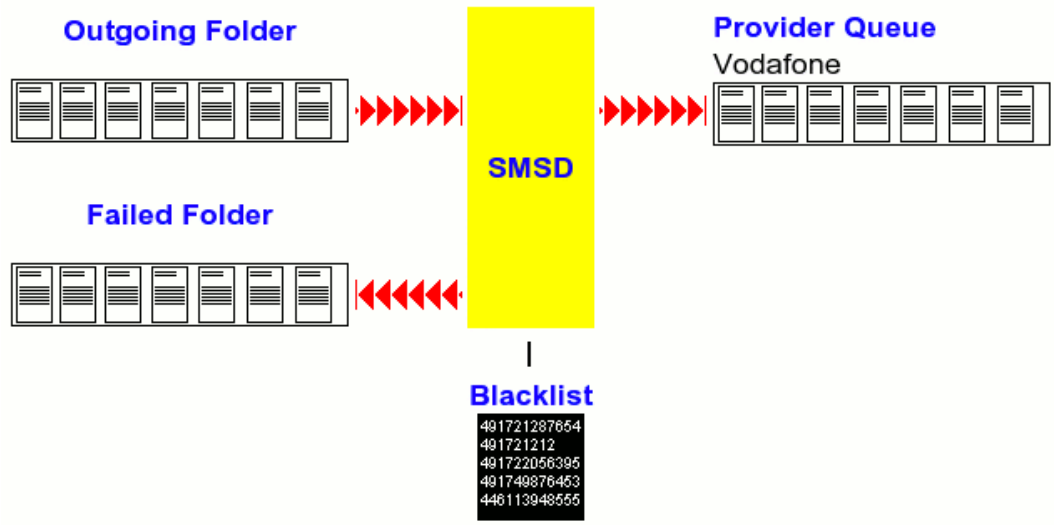
Das Dateiformat der SMS Files ist sehr einfach. Es ähnelt dem Dateiformat von eMails unter Unix. Zuerst kommt kein Kopf mit Absender, Empfänger und evtl. weiteren Informationen. Sie dürfen beliebig viele Zusatzinformationen in den Kopf schreiben. Smsd ignoriert alles, womit es nichts anfangen kann.

Dann kommt eine Leerzeile und danach der Text. Bei binären SM, z.B. Klingeltönen oder Logos, gilt die selbe Regel. Die Binärdaten beginnen unmittelbar nach der Leerzeile.

Im obigen Bild sehen Sie ein Beispiel für ein gültiges SMS File.

Die ausgehende Warteschlange erlaubt es Ihnen, viele SMS Files abzulegen, die dann nacheinander gesendet werden. Sie brauchen nicht nach dem Speichern jeder einzelnen Datei zu warten, bis sie tatsächlich gesendet wurde.

Der nächste Schritt ist, das SMS File auf Gültigkeit zu überprüfen. Diese Aufgabe nimmt Ihnen das Hauptprogramm ab.



Es liest alle Dateien in der ausgehenden Warteschlange (Outgoing Queue) ein und vergleicht die Empfängernummern mit der schwarzen Liste (Blacklist). Das ist eine Textdatei, in der alle Telefonnummern eingetragen werden, die keine SM bekommen dürfen. Die Blackliste kann auch Vorwahlen enthalten, um ganze Nummernbereiche zu sperren.

Alternativ gibt es auch eine weisse Liste (Whitelist). Hier tragen Sie alle Zielrufnummern ein, die SM bekommen dürfen. Wenn keine weisse Liste existiert, darf jeder eine SM bekommen, ausser diejenigen, die in der schwarzen Liste stehen. Wenn es sowohl eine weisse als auch eine schwarze Liste gibt, erhalten nur diejenigen Kurzmitteilungen, die in der weissen Liste stehen und nicht in der schwarzen. Die schwarze Liste hat Vorrang.

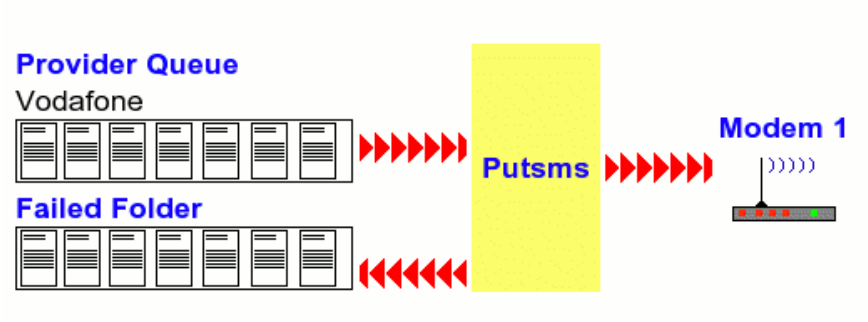
Wenn in dem SMS File ein Netzanbieter (Provider) Name angegeben ist, wird die Datei nun in die Warteschlange dieses Netzanbieters (Provider Queue) verschoben. Wenn der Netzanbieter jedoch unbekannt ist, vergleicht sms die Telefonnummer des Empfängers mit einer Liste in der Konfigurationsdatei sms.conf und ordnet die Kurzmitteilung anhand der Vorwahl dem richtigen Netzanbieter zu.

Anmerkung: Mittlerweilen existiert das PPRTU Projekt, welches es den mobilen Telefonkunden erlaubt, den Netzanbieter zu wechseln ohne eine neue Telefonnummer zu bekommen. Die Vorwahl in der Telefonnummer ist also kein eindeutiges Anzeichen mehr, zu welchem Netzanbieter die Telefonnummer gehört. Für sms ist das unkritisch, aber eine falsche Zuordnung kann zu erhöhten Gebühren für den SMS Versand führen und Statistiken verfälschen.

Wenn im SMS File weder der Netzanbieter angegeben wurde, noch die Vorwahl bekannt ist, wird die SMS als ungültig betrachtet und in die fehlgeschlagene Warteschlange (Failed Queue) verschoben. Die Nachricht wird dann nicht gesendet.

Wenn Sie die fehlgeschlagenen Nachrichten nicht aufbewahren wollen, können Sie das tun, indem Sie in der Konfigurationsdatei sms.conf einfach keine fehlgeschlagene Warteschlange angeben.

Jetzt wird die Kurzmitteilung gesendet.



Jeder Netzanbieter Warteschlange (Provider Queue) ist wenigstens ein Modem zugeordnet. Ein Programmteil von sms (Modem Task) liest die SMS Files aus den Netzanbieter Warteschlangen aus und sendet Sie mit Hilfe des Unterprogrammes Putsms.

Sie können mehrere Modems dem selben Netzanbieter zuordnen - was in den Beispielbildern hier nicht gezeigt wird. Dadurch erhöhen Sie die Geschwindigkeit, mit der viele Nachrichten gesendet werden. Ausserdem bleibt Ihr System nicht stecken, wenn eines von mehreren Modems ausfällt, denn die restlichen Modem arbeiten fleissig weiter.

Um zu verhindern, dass eine Nachricht von mehreren Modems gleichzeitig gesendet wird, benutzt sms sogenannte Sperrdateien (Lockfiles). Eine Sperrdatei ist eine Datei, die genau so heisst, wie das SMS File, es hat aber zusätzlich die Endung ".LOCK". Die Existenz einer Sperrdatei signalisiert den anderen Programmteilen, dass diese Nachricht bereits in Arbeit ist.

Für Neugierige schreibt sms seine Prozessnummer in die Datei. Der Betriebssystem-Kern gibt jedem Programm eine Prozessnummer, die Sie z.B. mit dem ps Befehl anzeigen lassen können. Eine technische Funktion hat die Nummer für sms jedoch nicht.

Die Lockfiles ermöglichen es Ihnen auch, mehrere Computer zum Versand von Kurzmitteilungen aufzustellen. Alle Computer sollten dann gemeinsam auf die selben Warteschlangen zugreifen. Fällt ein Computer aus, dann gerät Ihr System nicht ins stocken, denn die anderen Computer senden weiterhin.

Wenn das Senden einer Kurzmitteilung trotz einer Wiederholung fehlgeschlagen ist, wird die Datei in die fehlgeschlagene Warteschlange verschoben.

Wenn das Senden von unterschiedlichen Kurzmitteilungen durch ein Modem drei mal nacheinander fehlschlägt, geht sms davon aus, dass das Modem defekt ist. In diesem Fall wird das Modem für eine einstellbare Zeit (z.B. eine Stunde) blockiert. Dadurch werden die verbleibenden Kurzmitteilungen über die anderen Modems gesendet - falls vorhanden. Sie tun also gut daran, jedem Netzanbieter wenigstens zwei Modems zuzuordnen.

Wie Sie am großen Gesamtbild zu Beginn dieses Kapitels sehen können, ist es auch möglich, ein Modem für mehrere Netzanbieter zugleich zu benutzen. In der Konfigurationsdatei bestimmen Sie für jedes Modem anhand einer Liste, welche Netzanbieter Warteschlangen es bedienen soll. Dabei werden die Netzanbieter gemäss Ihrer Reihenfolge priorisiert behandelt. Ein Beispiel mit zwei Modems:

Modem 1 ist für Telekom D1 und Vodafone D2. Im Modem steckt eine SIM Karte von Telekom D1.

Modem 2 ist für Vodafone D2 und Telekom D1. Im Modem steckt eine SIM Karte von Vodafone D2.

Das Modem 2 wird vorzugsweise Nachrichten für D2 senden. Erst wenn die Warteschlange für D2 leer ist, sendet das Modem aushilfsweise Nachrichten für D1.

Umgekehrt verhält sich das Modem 1. Es sendet vorzugsweise Nachrichten für D1 und erst wenn diese Warteschlange leer ist, sendet es Nachrichten für D2.

Worin liegt der Nutzen dieser Konfiguration?

Auf diese Frage gibt es mehrere Antworten.

Falls das Modem 1 (für D1) ausfällt, hilft das Modem2 aus. So ist sichergestellt, dass Ihr System weiterhin arbeitet, auch wenn ein Modem defekt ist. Umgekehrt gilt das auch für Modem 2.

Falls sie sehr viele Nachrichten für D2 versenden, aber nur sehr wenige für D1, hilft das Modem 1 aus und sendet ebenfalls D2 Nachrichten. Sie verbessern die Geschwindigkeit. Umgekehrt gilt das auch für Modem 2.

Sie sind vor Netzausfällen eines einzelnen Netzanbieters geschützt, denn das andere Modem wird fortfahren, Kurzmitteilungen zu senden.

Sie sparen Geld, denn wenn Sie Kurzmitteilungen zum "eigenen" Netzanbieter schicken, ist das etwas billiger und sie erhalten bei einigen Verträgen sogar einen saftigen Mengenrabatt. Das Modem mit der D1 SIM Karte soll also vorzugsweise an D1 Empfänger senden und das andere Modem soll vorzugsweise an D2 Empfänger senden.

Der Punkt 4 ist als der Grund, warum Kurzmitteilungen von sms überhaupt nach Netzanbieter vorsortiert werden. Ohne diese Vorsortierung könnten Sie die Mengenrabatte und günstigeren Preise nicht optimal ausnutzen.

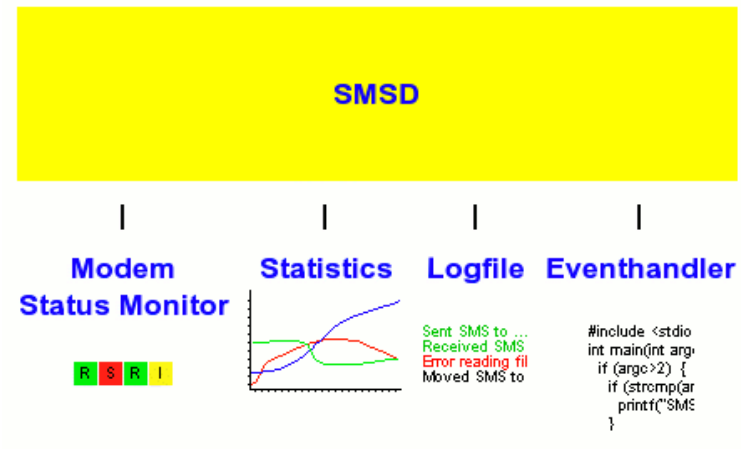
Nun möchte ich Ihnen erklären, wie der Empfang von SM funktioniert.

Der Programmteil von sms, der das Modem steuert (Modem Task), ruft regelmässig empfangene Kurzmitteilungen ab. Dazu benutzt er das Unterprogramm Getsms. Die empfangenen Nachrichten werden in der eingehenden Warteschlange (Incoming Queue) abgelegt.

Sie können in der Konfigurationsdatei von sms festlegen, welche Modems Kurzmitteilungen empfangen sollen und was Priorität hat: Senden oder Empfangen.

Bei den meisten Modems müssen Sie die Priorität auf Empfangen einstellen, denn wenn der Posteingang voll ist, können die meistem Modems leider keine Nachricht senden. Probieren Sie es im Zweifelsfall aus.

Weitere Funktionen:



Nebenbei, aber für die meisten Anwender dennoch sehr wichtig, kann sms noch einiges mehr. Zum Beispiel kann es einen Modem-Statusmonitor anzeigen. Dieser Monitor zeigt an, welche Modems gerade senden, empfangen, blockiert sind oder nichts zu tun haben.

Smsd sammelt Statistiken über die Anzahl gesendeter und empfangener SM, sowie über die Auslastung der Modems. Diese Statistik hilft Ihnen z.B. bei der Entscheidung, ob Sie die Leistung Ihres System durch hinzufügen weiterer Modems sinnvoll erhöhen können, oder nicht. Mit einer Tabellenkalkulation wie Microsoft Excel oder KSpread können Sie aus diesen tabellarischen Statistiken eindrucksvolle Graphen erstellen.

Wenn Sie sms stoppen oder killen, verlieren Sie keine Statistikdaten. Dies ist eine Besonderheit in den SMS Server Tools. Die meisten Programme verlieren alle gesammelten Zahlen des aktuellen Zeitabschnittes (z.B. Stunde), wenn Sie das laufende Programm abbrechen. Smsd hingegen sichert die Statistiken in eine temporäre Datei, um sie beim nächsten Start korrekt aufzuzeichnen.

In Protokolldateien (Logfiles) zeichnet sms alles Aktionen auf, die es ausgeführt hat. Sie können die Detailstufe einstellen. Vielleicht möchten Sie nur Fehlermeldungen protokollieren oder jedes kleinste Detail sehen, z.B. die Modembefehle.

Smsd kann auf den Bildschirm protokollieren (Standard Ausgabe), in eine Datei schreiben oder den Syslog Dämon von Unix zum Aufzeichenn von Protokollen benutzen. Sie können dies frei konfigurieren.

Ereignis-Programme (Eventhandler) können beim Auftreten bestimmter Ereignisse gestartet werden. z.B. wenn eine Kurzmitteilung gesendet oder empfangen wurde. Die Eventhandler ermöglichen Ihnen das Andocken anderer Programme, womit Sie die Funktionen von sms ohne großen Aufwand erheblich erweitern können. Ein typisches Anwendungsbeispiel ist z.B. eine Gebührenabrechnung mittels SQL Datenbank.

Seit Version 1.10 kann sms sogar die Gültigkeit von SMS Files in der ausgehenden Warteschlange (Outgoing Queue) von externen Programme prüfen lassen, bevor sie gesendet werden. Wie Sie im Laufe dieses Buches feststellen werden, liegt in den Ereignis-Programmes ein elementarer Vorteil von den SMS Server Tools im Vergleich zu anderen ähnlichen Programmen.

Alarmhandler sind eigentlich auch Ereignis-Programme. Sie werden aufgerufen, wenn Fehler aufgetreten sind. Alarm Handler können Beispielsweise benutzt werden, um eine Störmeldung per eMail zu versenden oder eine Signalleuchte einzuschalten.

5.2. Installation und Konfiguration

Wie man die SMS Server Tools unter unterschiedlichen Betriebssystemen installiert, habe ich im Kapitel 4.2 erklärt. Zur Erinnerung wiederhole ich hier nochmal die wesentlichen Befehle:

make -s

make -s install

Der erste Befehl überstetzt die Programmquelltexte in ein ausführbares Programm, entsprechend Ihrem Betriebssystem. Der zweite Befehl installiert das Programm. Die folgende Liste zeigt Ihnen, welche Dateien wohin installiert werden, bezogen auf Version 1.15. Neuere Versionen haben möglicherweise zusätzliche Dateien:

/usr/local/bin/smsd	Das Hauptprogramm
/usr/local/bin/putsms	Unterprogramm zum Senden
/usr/local/bin/getsms	Unterprogramm zum Empfangen
/usr/local/bin/sendsms	Shell Script. Kann benutzt werden, um an der Eingabeaufforderung manuell eine SM zu senden.
/usr/local/bin/sms2html, sms2unicode, unicode2sms	Shell Scripte zum Konvertieren von SMS Dateien.
/etc/smsd.conf	Vorlage für Konfigurationsdatei, muss auf jeden Fall individuell angepasst werden.
/etc/init.d/sms	Startup-Script, kann benutzt werden, um smsd zu starten und zu stoppen, insbesondere beim Booten.
/var/spool/sms/incoming	Warteschlange für empfangene SM
/var/spool/sms/outgoing	Warteschlange für ausgehende SM
/var/spool/sms/checked	Warteschlange für geprüfte ausgehende SM

Wenn Sie die SMS Server Tools später durch eine neuere Version aktualisieren, werden Ihre Scripte und Konfigurationsdateien nicht überschrieben.

Smsd wird durch eine einzige Konfigurationsdatei eingerichtet. Dies ist normalerweise die Datei /etc/smsd.conf. Sie können auch eine andere Konfigurationsdatei benutzen, dann müssen Sie diese aber mit der Option -c angeben. Beispiel:

smsd -c /etc/smsd.conf.test

Die Vorlage für die Konfigurationsdatei zeigt Ihnen, wie Sie die SMS Server Tools für ein Modem einrichten. Die Datei ist für einen schnellen Einstieg vorgesehen und kurz gehalten.

Im Verzeichnis smstools/examples befindet sich eine weitere Konfigurationsdatei mit allen erdenklichen Optionen.

Ich möchte Sie nun nach und nach in die vielen Einstellungen einweisen, indem ich mit einer einfachen Konfigurationsdatei beginne.

5.2.1. Einfache Konfigurationsdatei

Eine einfache Konfigurationsdatei sieht so aus (/etc/smsd.conf):

```
devices = GSM1

[GSM1]
device = /dev/ttyS0
incoming = yes
pin = 1111
```

Wenn Sie diese Datei mit der Vorlage smsd.conf.full vergleichen, werden Sie sehen, dass ich hier viele Zeilen ausgelassen habe. Das ist in Ordnung, denn smsd benutzt bei allen fehlenden Angaben Standardwerte. Ich werde später auf die anderen Zeilen eingehen.

Für Anfänger ist es sicher besser, zunächst mit einer vereinfachten Konfigurationsdatei zu beginnen, wie oben abgebildet.

Sie können beliebig viele Kommentare an jeder Stelle einfügen. Markieren Sie jede Kommentarzeile mit einem Raute Zeichen, wie oben gezeigt.

Mit "devices" legen Sie fest, wieviele GSM Modems Sie benutzen wollen und wie diese heißen. Verwenden Sie keine Sonderzeichen und keine Leerzeichen (außer den Unterstrich).

Wenn Sie mehrere Modems benutzen, geben Sie jedem Modem einen Namen und listen Sie diese auf - mit Kommata getrennt.

Dann kommt ein Konfigurationsabschnitt für jedes Modem, in diesem Fall nur ein Modem.

Hinter "device" gibt man den Namen der seriellen Schnittstelle an. Unter Linux heißt der erste serielle Anschluss /dev/ttyS0 und der zweite heisst /dev/ttyS1. Unter Windows heissen sie /dev/com1und /dev/com2. Solaris nennt den ersten seriellen Anschluss /dev/cuaa und den zweiten /dev/cuab.

Mit der Einstellung "incoming = yes " oder "incoming = no" bestimmen Sie, ob dieses Modem auch Nachrichten empfangen soll.

SIM Karten sind normalerweise mit einer PIN gesichert. Durch die Option "pin" teilen Sie diese den SMS Server Tools mit. Bei Handies brauchen Sie diese Option nicht, denn dort geben Sie die PIN immer über die eingebaute Tastatur ein, wenn Sie das Handy einschalten.

Ich möchte Ihnen empfehlen, die PIN der SIM Karte zu deaktivieren, denn dann laufen die SMS Server Tools etwas schneller. Legen Sie die SIM Karte dazu in ein Handy ein und deaktivieren Sie die PIN mit dem Menü des Handy. Die Bedienungsanleitung des Handies sollte erklären wie das geht.

5.2.2. Mehrere Modems und Netzanbieter

Nun zeige ich Ihnen, wie Sie die SMS Server Tools für mehrere Modems und mehrere Netzanbieter einrichten. Betrachten Sie dieses Kapitel als Beispiel, sie müssen sich nicht 100% daran halten. Experimentieren Sie ggf. mit anderen Zuordnungen von Modem zu Netzanbieter um für Ihre Anwendung die optimale Leistung zu erzielen.

```
devices = GSM1, GSM2, GSM3, GSM4, GSM5
[queues]
D1 = /var/spool/sms/D1
D2 = /var/spool/sms/D2
O2 = /var/spool/sms/O2
EPLUS = /var/spool/sms/EPLUS
QUAM = /var/spool/sms/QUAM
MOBILCOM = /var/spool/sms/MOBILCOM

[provider]
D1 = 49160, 49170, 49171, 49175, 49151
D2 = 491520, 49162, 49172, 49173, 49174
O2 = 49176, 49179, 49159
EPLUS = 49163, 49177, 49178, 49157
QUAM = 49150
MOBILCOM = 49156

[GSM1]
# mit D1 Karte
device = /dev/ttyN000
incoming = yes
queues = D1

[GSM2]
# mit D1 Karte
device = /dev/ttyN001
incoming = yes
queues = D1

[GSM3]
# mit D2 Karte
device = /dev/ttyN002
incoming = yes
queues = D2

[GSM4]
# mit D2 Karte
device = /dev/ttyN003
incoming = yes
queues = D2

[GSM5]
# mit E-Plus Karte
device = /dev/ttyN004
incoming = yes
queues = O2, EPLUS, QUAM, MOBILCOM
```

In dieser Konfigurationsdatei fehlen die PIN Nummern, sie müssen also die PIN's der fünf SIM Karten deaktivieren.

Die fünf Modems sind an ein Digi Etherlite angeschlossen, denn kein normaler Computer hat für fünf Modems ausreichend serielle Anschlüsse. Die Namen der seriellen Anschlüsse habe ich der Installationsanleitung des Digi Etherlite entnommen.

Die ersten beiden Modems bedienen die Netzbetreiber-Warteschlange für D1. Das dritte und vierte Modem bedient die Netzbetreiber-Warteschlange für D2. Das letzte Modem ist für die anderen Netzbetreiber vorgesehen.

Bei D1 und D2 ergibt sich somit eine höhere Leistung und Schutz vor Ausfall eines einzelnen Modems. Bei den anderen Netzbetreibern verzichten wir aus Kostengründen auf ein zweites Modem, denn dort ist der Kundenkreis viel kleiner.

Natürlich können Sie auch für die anderen Netzbetreiber zwei Modems benutzen. In diesem Beispiel wollte ich Ihnen lediglich demonstrieren, dass man die Anzahl der Modems pro Netzbetreiber beliebig variieren kann.

Ins Ausland werden keine Kurzmitteilungen gesendet. Wenn Sie das doch wollen, dann fügen Sie einfach eine OTHER Queue hinzu und lassen Sie wenigstens ein Modem darauf zugreifen.

```
[queues]
...
OTHER = /var/spool/sms/OTHER
[provider]
...
OTHER = 0,1,2,3,4,5,6,7,8,9
```

Wenn smsd eine Nachricht findet, die zu keinem Provider passt, dann passt Sie aber auf jedenfall zu OTHER und wird von dem Modem (oder den Modems) gesendet, das dieser Warteschlange zugewiesen ist.

5.2.3. Alle Konfigurationseinstellungen

Nun, da Sie die Grundlagen der Konfiguration kennengelernt haben, können Sie sich mit den anderen Einstellungen befassen. Auf den folgenden Seiten erkläre ich Ihnen alle Einstellmöglichen der Datei smsd.conf. Lesen Sie dieses Kapitel bitte aufmerksam durch, denn hier verbergen sich viele nützliche Einstellungen, die Sie nicht verpassen sollten.

Die Konfigurationsdatei besteht aus vier Abschnitten:

- Haupteinstellungen
- Warteschlangen
- Netzbetreiber
- Eine oder mehrere Modemeinstellungen

Die Schreibweise der Datei entspricht den meisten Konfigurationsdateien unter Windows.

Abschnitte beginnen mit [Abschnittname], innerhalb der Abschnitte werden mehrere Variablen mit dem Gleichheitszeichen gesetzt:

```
Variable = Wert
```

Listen sind mehrere Werte, durch Kommata getrennt:

```
Variable = Wert1, Wert2, Wert3
```

Die gesamte Konfigurationsdatei unterscheidet zwischen großen und kleinen Buchstaben, ausser bei booleanschen Werten. Booleansche Werte sind:

```
yes, on, true, 1,
no, off, false, 0
```

Sie können diese beliebig miteinander mischen.

5.2.3.1. Haupteinstellungen

Der Hauptabschnitt hat keinen Namen und beginnt ganz oben in der Konfigurationsdatei.

devices = Liste mit Modemnamen

Legt die Anzahl und Namen der Modems fest. Verwenden Sie nur Buchstaben, Ziffern und den Unterstrich "_" im Namen. Die Modemnamen tauchen in Statistiken, Dateinamen, Protokoldateien und Ereignis-Programmen auf.

Sie muessen die Modems hier nicht so nennen, wie sie wirklich heissen. Erfinden Sie einfach Namen, die Ihnen gefallen. Haupsache, sie sind eindeutig.

spool = Verzeichnisname

Hier geben Sie das Verzeichnis für die ausgehende Warteschlange an. Der Vorgabewert ist /var/spool/sms/outgoing.

checked = Verzeichnisname

Hier geben Sie das Verzeichnis an, in welches geprüfte ausgehende Nahrichten verschoben werden. Der Vorgabewert ist /var/spool/sms/checked. Diese Variable hat keine Bedeutung, wenn Sie [provider] und [queue] Verzeichnisse festlegen.

failed = Verzeichnisname

Hier geben Sie das Verzeichnis an, in dem fehlgeschlagene SMS Files aufbewahrt werden. Wenn Sie keine Angabe machen, werde fehlgeschlagene Dateien einfach gelöscht.

incoming = Verzeichnisname

Legt fest, wo empfangene Nachrichten abgespeichert werden. Der Vorgabewert ist /var/spool/sms/incoming.

sent = Verzeichnisname

Wenn Sie gesendete Nachrichten aufbewahren wollen, geben Sie dieses Verzeichnis an.

mypath = Verzeichnisname

Geben Sie hier das Verzeichnis an, wo getsms und putsms installiert wurden, also normalerweise /usr/local/bin. Das ist auch der Vorgabewert, wenn Sie keine Angabe machen.

logfile = Dateiname oder Zahl

Name der Protokolldatei. Wenn Sie keinen Namen angeben, wird sys Syslog Dämon von Unix verwendet bzw. das Ereignisprotokoll von Windows.

Wenn Sie einen Dateinamen angeben, werden alle Informationen mit Zeitstempel und Programmname in die Datei geschrieben.

Sie können auch eine Zahl angeben, diese wird dann als Dateihandle interpretiert. Sie können solche Nummern nur zusammen mit Shell Ein-/Ausgabeumleitung benutzen. Die 1 ist die Standard-Ausgabe also normalerweise der Bildschirm. Durch Ein-/Ausgabeumleitung können Sie diese auch in Dateien oder andere Programme umleiten.

loglevel = Zahl

Legt die Detailstufe der Protokolldatei fest. Der Vorgabewert ist 4, wenn Sie eine Protokolldatei verwenden oder 7, wenn Sie syslog verwenden. Der Wert 0 bedeutet, dass gar nichts protokolliert wird. Es werden alle Informationen protokolliert, deren Stufe kleiner oder gleich der Zahl ist:

7	debug	Alle Details werden protokolliert, sogar die Modembefehle.
6	info	Informationen werden protokolliert. Das Logfile enthält viele Details, aber nicht die Modembefehle, die bei der Fehlersuche benötigt werden.
5	notice	Informationen, wenn eine Nachricht gesendet oder empfangen wurde, sowie Informationen über alle nicht normalen Ereignisse.
4	warning	Warnung, wenn das Programm ein Problem hat, z.B. eine Nachricht nicht senden kann.
3	error	Fehlermeldungen, wenn das Programm ein temporäres Problem hat, z.B. ein Modem mit ERROR antwortet oder eine Datei nicht gelesen werden kann.
2	critical	Kritische Fehler. Wenn das Programm ein permanentes Problem hat, z.B. das Senden mehrfach fehlschlägt oder die Zugriffsrechte auf Warteschlangen nicht ausreichen.
1		Diese Stufe ist für den Betriebssystem-Kern reserviert und wird von den SMS Server Tools nicht benutzt.

Die Nummern gelten für Linux und vermutlich auch für alle anderen Betriebssysteme. 100% sicher bin ich mir da allerdings nicht.

Wenn Sie den Syslog Dämon benutzen oder das Windows Ereignisprotokoll, können Sie die Detailstufe sowohl hier, als auch in der Konfiguration des Betriebssystems reduzieren.

alarmhandler = Programmname

Gibt ein Programm an, dass bei jedem Alarm gestartet werden soll. Es darf auch ein Shell-Script sein. Geben Sie das Programm mit vollem Pfad an. Alarmhandler werden später detailliert erklärt. Wenn Sie keinen Alarmhandler brauchen, lassen Sie diese Angabe weg.

alarmlevel = Zahl

Legt fest, bei welchen Stufen ein Alarmhandler gestartet wird. Sie können Werte von 2 bis 5 benutzen:

5	notice	Informationen, wenn eine Nachricht gesendet oder empfangen wurde, sowie Informationen über alle nicht normalen Ereignisse.
4	warning	Warnung, wenn das Programm ein Problem hat, z.B. eine Nachricht nicht senden kann.
3	error	Fehlermeldungen, wenn das Programm ein temporäres Problem hat, z.B. ein Modem mit ERROR antwortet oder eine Datei nicht gelesen werden kann.
2	critical	Kritische Fehler. Wenn das Programm ein permanentes Problem hat, z.B. das Senden mehrfach fehlschlägt oder die Zugriffsrechte auf Warteschlangen nicht ausreichen.

delaytime = Zahl

Legt fest, wie viele Sekunden sich smsd schlafen legt, wenn eine Warteschlange leer ist. Nach Ablauf dieser Zeit prüft smsd erneut, ob neue SMS Files in der Warteschlange stehen

Je grösser diese Zahl ist, um so geringer wird die Festplatte und der Prozessor durch smsd belastet. Wenn Sie Wert auf maximale Leistung legen, stellen Sie eine Sekunde ein und benutzen Sie diesen Computer nicht gleichzeitig für andere Programme.

Der Vorgabewert ist 10 Sekunden.

errorsleeptime = Zahl

Wenn ein Modem mit ERROR antwortet, kann das Programm eine zusätzliche Pause einlegen. Manche Modems brauchen dies, bevor Sie neue Befehle annehmen können.

Der Vorgabewert ist 10 Sekunden.

blocktime = Zahl

Wenn das Versenden von drei aufeinander folgenden Kurzmitteilungen fehlschlägt, geht smsd davon aus, dass das Modem defekt ist. Das Programm blockiert dieses Modem dann solange, wie sie hier angeben (in Sekunden). So verhindern Sie, dass sich Ihr Computer sinnlos mit einem defekten Modem beschäftigt und das Senden einer SM nach der anderen fehlschlägt. Der Vorgabewert ist 3600 Sekunden, also eine Stunde. Erst nach Ablauf dieser Zeit versucht smsd erneut, das Modem zu benutzen.

eventhandler = Programmname

Immer, wen eine Kurzmitteilung gesendet oder empfangen wurde, oder wenn das Senden fehlgeschlagen ist, kann smsd ein externes Programm aufrufen. Geben Sie das Programm mit vollem Pfad an. Ereignisprogramme werden später detailliert erklärt.

stats = Verzeichnis

Diese Einstellung legt fest, in welches Verzeichnis smsd Statistiken speichern soll. Wenn Sie diese Angabe weglassen, werden keine Statistiken geschrieben. Wie die Statistik-Dateien benannt werden und wie deren Inhalt aussieht, wird später erklärt.

stats_interval = Zahl

Hiermit legen Sie fest, in welchen Zeitabständen Statistik-Dateien geschrieben werden sollen. Der Wert ist in Sekunden, 3600 bedeutet also, dass Statistiken stündlich geschrieben werden. Smsd stellt sicher, dass die Zeitpunkte der Dateien immer gleich sind, egal wann sie smsd gestartet haben. Beim Wert 3600 werden Statistiken also immer zu jeder vollen Stunde geschrieben, und nicht 60 Minuten nach dem Programmstart.

blacklist = Dateiname

Legt den Dateinamen der schwarzen Liste fest. Wenn Sie diese Angabe weglassen, wird keine schwarze Liste verwendet.

whitelist = Dateiname

Legt den Dateinamen der weissen Liste fest. Wenn Sie diese Angabe weglassen, wird keine weisse Liste verwendet.

checkhandler = Programmname

Zusätzlich zu den schwarzen und weissen Listen können Sie auch externe Programme benutzen, um zu entscheiden, ob eine Nachricht gesendet werden darf, oder nicht.

Bevor smsd eine Nachricht in die Netzbetreiber-Warteschlangen sortiert, startet es den Checkhandler und übergibt diesem als einziges Argument den Dateinamen des fraglichen SMS Files. Wenn der Checkhandler mit Exitcode 0 endet, wird die Nachricht gesendet, ansonsten wird die nicht gesendet.

Wenn Sie keinen Checkhandler brauchen, lassen Sie diese Angabe weg.

autosplit = 0, 1, 2 oder 3

0	Automatisches Teilen zu langer Textnachrichten ist deaktiviert.
1	Nachrichten werden geteilt, ohne Nummerierung
2	Nachrichten werden geteilt, mit Nummerierung
3	Nachrichten werden als verkettete Nachrichten gesendet, die das empfangende Handy wieder zusammenfügt (kann nicht jedes Handy)

Binärdateien und Unicode Nachrichten (Klingeltöne, Logos, etc.) werden niemals geteilt und dürfen maximal 140 Bytes groß sein.

receive_before_send = boolean

Wenn Sie diese Variable auf yes stellen, stellt smsd sicher, dass wenigstens ein Speicherplatz auf der SIM Karte leer ist, bevor eine Nachricht gesendet wird. Manche Modem brauchen dies, weil sie keine Nachrichten senden können, wenn die SIM Karte voll ist.

Durch diese Option bewirken Sie, dass smsd immer abwechselnd eine Nachricht sendet und empfängt. Ohne diese Option sendet smsd vorzugsweise alle Nachrichten, bevor empfangene Nachrichten ausgelesen werden.

5.2.3.2. Warteschlangen

Die SMS Server Tools sortieren ausgehende Nachrichten in einzelne Warteschlangen für die jeweiligen Netzbetreiber.

Dadurch bekommen Sie die Möglichkeit, durch Verwendung unterschiedlicher SIM Karten von unterschiedlichen Netzanbietern, Kosten zu sparen.

Diese Funktion ist optional, sie können den kompletten [queues] Abschnitt weglassen, wenn Sie keine Sortierung benötigen.

Nach den Haupteinstellungen definieren Sie in der Konfigurationsdatei smsd.conf die Namen der Warteschlangen und die zugehörigen Verzeichnisnamen auf der Festplatte.

[queues]
Name = Verzeichnisname
Name = Verzeichnisname
Name = Verzeichnisname
...

Benutzen Sie nur Buchstaben, Ziffern und den Unterstrich, vermeiden Sie Sonderzeichen. Beispiel:

```
[queues]
D1 = /var/spool/sms/D1
D2 = /var/spool/sms/D2
O2 = /var/spool/sms/O2
EPLUS = /var/spool/sms/EPLUS
QUAM = /var/spool/sms/QUAM
MOBILCOM = /var/spool/sms/MOBILCOM
```

5.2.3.3. Netzanbieter

In diesem Abschnitt geben Sie die Ortsvorwahlen der einzelnen Netzbetreiber an. Die Software verwendet diese Einstellung, um Nachrichten in Warteschlangen zu sortieren.

Für jede Zeile im vorherigen Abschnitt [queues] muss es genau eine gleichnamige Zeile im Abschnitt [provider] geben.

Schreiben Sie die Vorwahlen immer in internationaler Form, aber ohne das voranstehende Plus Zeichen.

[provider]
Name = Liste mit Vorwahlen
Name = Liste mit Vorwahlen
Name = Liste mit Vorwahlen

Beispiel:

```
[provider]
D1 = 49160, 49170, 49171, 49175, 49151
D2 = 491520, 49162, 49172, 49173, 49174
O2 = 49176, 49179, 49159
EPLUS = 49163, 49177, 49178, 49157
QUAM = 49150
MOBILCOM = 49156
```

5.2.3.4. Modemeinstellungen

Nach den Netzanbieter-Vorwahlen folgen ein oder mehrere Modemeinstellungen. Für jedes angeschlossene Modem muessen Sie einen passenden Abschnitt mit Modemeinstellungen in die Konfigurationsdatei schreiben.

Orientieren Sie sich dabei an der Zeile

devices = Liste mit Modemnamen

aus dem Hauptabschnitt. Für jedes dort aufgelistete Modem sucht smsd einen entsprechenden Abschnitt mit Modemeinstellungen.

Es ist erlaubt, mehr Modemeinstellungen in die Datei zu schreiben, als Sie wirklich benutzen. Wenn Sie Beispielsweise ein Modem zeitweise ausser Betrieb nehmen, weil es repariert werden muss, brauchen Sie nicht die ganzen Einstellungen zu löschen. Entfernen Sie das defekte Modem einfach aus der "devices" Zeile im Hauptabschnitt und belassen Sie die Modemeinstellungen einfach, wie sie sind.

Smsd interessiert sich nur für Modems, die in der "devices" Zeile aufgelistet sind.

Die folgenden Einstellungen können Sie für jedes einzelne Modem vornehmen:

[Modemname]

Jeder Abschnitt mit Modemeinstellungen wird durch den Modemnamen eingeleitet. Dahinter folgen zusammenhängend alle Einstellungen für das genannte Modem.

init = Zeichenfolge

Hier können Sie einen zusätzlichen Modembefehl eingeben, der vor dem Senden oder Empfangen vom Modem ausgeführt wird. Im einfachsten Fall brauchen Sie diese Einstellung nicht.

Wenn Sie hier "ATE0" hinschreiben, unterdrücken Sie das "echo" aller Modembefehle, dass vom Modem erzeugt wird. Das Programm läuft dann minimal schneller und die Protokolldatei (mit Loglevel 7) sieht übersichtlicher aus, weil dort nicht jeder Modembefehl doppelt erscheint.

Manche GSM Modems, insbesondere Handies speichern empfangene Kurzmitteilungen vorzugsweise im Internen Speicher (RAM) anstatt auf der SIM Karte, versuchen aber lediglich, von der SIM Karte zu lesen. Das ist nicht so gut, denn so können Sie natürlich die empfangenen SM nicht mit dem Computer auslesen.

Wenn Sie dieses Problem bemerken, initialisieren Sie Ihr Modem mit

init = ATE0+CPMS="SM" oder

init = ATE0+CPMS="ME"

So befehlen Sie dem Modem, dass es alle empfangenen Kurzmitteilungen auf der SIM Karte (SM) oder im internen Speicher (ME) speichern UND lesen soll. Je nach Gerät führt nur eine der beiden Varianten oder beide zum Erfolg.

Um mit Siemens Modems Status-Reports empfangen zu können, müssen Sie diese mit **AT+CNMI=2,0,0,2,1** initialisieren.

device = Schnittstellen-Name

Hier geben Sie den Namen der seriellen Schnittstelle an, an die das Modem angeschlossen ist.

Unter Linux heist der erste serielle Anschluss /dev/ttyS0 und der zweite heisst /dev/ttyS1. Unter Windows heissen sie /dev/com1und /dev/com2. Solaris nennt den ersten seriellen Anschluss /dev/cuaa und den zweiten /dev/cuab.

incoming = boolean (oder Zahl)

Wenn Sie hier "yes" oder "1" schreiben, liest smsd regelmässig empfangene Kurzmitteilungen aus dem Modemspeicher aus. Der Vorgabewert ist "no".

Bitte beachten Sie, dass manche Modems nicht senden können, wenn der Empfangsspeicher voll ist. Wenn Sie keine Nachrichten empfangen wollen, und dieses Problem haben, lassen Sie die betroffenen SIM Karte vom Netzbetreiber in eine schwarze Liste eintragen. Sie bekommen dann nie wieder SM an diese SIM Karte zugestellt.

Das Auslesen empfangener SM hat beim Wert "yes" oder "1" geringere Priorität, als das Senden von SM. Smsd liest empfangene SM nur aus, während es nichts zu senden gibt.

Wenn Sie hier "2" schreiben, verhält es sich anders herum. Dann hat das Empfangen Vorrang und smsd sendet nur, wenn es keine weitere empfangene Kurzmitteilung gibt.

queues = Liste mit Warteschlangen Namen

Durch diese Zeile legen Sie fest, welche Warteschlangen das Modem bedienen soll. Beziehen Sie sich hier auf die Namen, die Sie im [queues] und [provider] Abschnitt gewählt haben.

pin = Zahl

Geben Sie hier die Geheimzahl der SIM Karte an. Sie können die Geschwindigkeit des Programmes etwas erhöhen, indem Sie die PIN mit einem Handy deaktivieren und diese Einstellung dann weglassen, denn dann braucht smsd nicht mehr zu prüfen, ob die PIN bereits eingegeben wurde oder ob sie eingegeben werden muss. Bei Handies entfällt diese Angabe sowieso, denn dort müssen Sie die PIN grundsätzlich über die Tastatur des Handies eingeben.

mode = Modus

Nahezu alle GSM Modems werdem im Modus "new" betrieben, was daher auch die Standardvorgabe ist. Bei anderen Modems brauchen Sie diese Einstellung:

old	Falcom A1 oder andere sehr alte GSM Modems aus GSM Phase 1
new	Alle GSM Modems aus Phase 2 oder neuere
ascii	Für Testzwecke. Sie können im Ascii Modus nur 140 Zeichen Text senden. Das Programm kommuniziert im Klartext mit dem Modem, anstatt Text und Rufnummern im PDU Format zu verschlüsseln. Manche Modems unterstützen diesen Modus nich

smsc = Nummer

Jedes GSM Modem und jedes Handy verwendet automatisch die SMS Zentrale, die der Netzbetreiber vorgibt. Diese Voreinstellung befindet sich auf der SIM Karte. Sie können auch jedes andere SMSC verwenden, indem Sie dessen Rufnummer hier angeben. Die Benutzung dieser Einstellung macht das Programm etwas langsamer, weil es diese Nummer dem Modem extra mitteilen muss.

Schreiben Sie die Nummer im internationalen Format, aber ohne das vorangestellte Plus Zeichen, z.B.: 491722270000

baudrate = Zahl

Mit dieser Zeile legen Sie die Übertragungsgeschwindigkeit der seriellen Schnittstelle fest. Alle mir bekannten Modems funktionieren mit dem Vorgabewert 19200 Bits pro Sekunde. Wenn Ihr Modem nicht reagiert, versuchen Sie es mit 9600. Bei Infrarot-Schnittstellen hat diese Einstellung keine Bedeutung, weil dort die Übertragungsrate vom Betriebssystem automatisch vorgegeben wird.

rtscts = boolean

Wenn Sie hier "no" schreiben, verzichten die SMS Server Tools auf Hardware Handshake Leitungen. Manche Handies und viele nicht originale Anschlusskabel sind billig konstruiert - man hat dort die Handshake Leitungen einfach weggelassen.

Wann immer es geht, sollten Sie Hardware Handshake unbedingt benutzen, also auf "yes" stellen oder diese Zeile ganz weglassen. Hardware Handshake stellt sicher, dass der Computer nicht zu schnell mit dem Modem "spricht".

cs_convert = boolean

Der Vorgabewert ist "yes" und bedeutet, dass die SMS Server Tools die Zeichensätze aller Kurzmitteilungen in den Zeichensatz ISO-8859-15 übersetzen. Unter Windows wird dieser Zeichensatz "Ansi" genannt.

Diese Übersetzung benötigen Sie, um SMS unter Windows oder Unix richtig darstellen zu können und um umgekehrt Text korrekt als SMS senden zu können.

Wenn Sie die Einstellung "no" verwenden, werden die Texte unverändert übertragen, bis auf eine Ausnahme: Das @ Zeichen wird in den ASCII Code 183 übersetzt, denn im GSM Alphabet hat es den Code 0, und der Code 0 markiert das Ende eines Textes in nahezu jedem Windows und Unix Programm.

report = boolean

Wenn Sie diese Einstellung auf "yes" stellen, fordert das Modem für jede gesendete Kurzmitteilung einen Statusbericht bei der SMS Zentrale an. Nicht jede SMS Zentrale unterstützt diese Funktion - in Deutschland sollten Sie damit aber kein Problem haben.

Viel häufiger sind die Modems selbst schuld, wenn das Empfangen von Statusberichten nicht funktioniert. Viele Modems von Siemens, Digicom und Wavecom können keine Statusberichte empfangen. Und alle mir bekannten Handies empfangen zwar Statusberichte, aber sie können diese nur auf dem Display anzeigen und nicht über die serielle Schnittstelle vom Computer auslesen lassen.

Falcom Modems haben sich in dieser Hinsicht als absolut zuverlässig erwiesen.

eventhandler = Programmname

Immer, wenn eine Kurzmitteilung gesendet oder empfangen wurde, oder wenn das Senden fehlgeschlagen ist, kann smsd ein externes Programm aufrufen. Geben Sie das Programm mit vollem Pfad an. Ereignisprogramme werden später detailliert erklärt.

Wenn Sie sowohl im Hauptabschnitt, als auch hier einen Eventhandler angeben, dann hat dieser hier Vorrang. Die SMS Server Tools starten immer nur einen Eventhandler, und zwar vorzugsweise den Modem-spezifischen.

5.3. SMS Server Tools starten/stoppen

In diesem Kapitel lernen Sie, wie Sie die SMS Server Tools korrekt starten und stoppen.

Das Hauptprogramm der SMS Server Tools heisst smsd. Alle anderen Programme werden von smsd als Unterprogramme benutzt. Sie werden die Unterprogramme also niemals direkt benutzen.

Die SMS Server Tools beinhalten ein Start/Stop Script, welches Sie benutzen können, um das Programm korrekt zu starten und zu stoppen. Bei der Installation wird es als /etc/init.d/sms oder /sbin/init.d/sms installiert - je nach Betriebssystem Version.

Zu Starten geben Sie ein:

/etc/init.d/sms start

Dabei werden zuerst alle alten Sperrdateien gelöscht und dann wird smsd im Hintergrund gestartet. Mit dem Befehl

ps -ef | grep sms

können Sie sich davon überzeugen.

Zum Stoppen geben Sie ein:

/etc/init.d/sms stop

Das Script findet die Prozessnummer von smsd heraus und beendet das Programm dann mittels kill Befehl.

Die folgenden Zeilen, die ich etwas nach rechts eingerückt habe, gelten nur für Unix, also Solaris und Linux:

Wenn Sie die SMS Server Tools automatisch beim Hochfahren des Servers starten möchten, legen Sie sich eine Verknüpfung im entsprechenden Runlevel-Verzeichnis an. Die Runlevel Verzeichnisse heissen /etc/init.d/rc0.d bis rc6.d. Bei manchen Linux Versionen ist es /sbin/init.d/rc0.d bis rc6.d.

Nun müssen Sie zuerst herausfinden, welcher Runlevel der richtige ist, schliesslich stehen 7 zur Auswahl. Im Kapitel 4.2 hatte ich bereits erwähnt, wie das geht, ich möchte es dennoch an dieser Stelle noch einmal wiederholen.

Geben den Befehl

grep :initdefault /etc/inittab

ein und merken Sie sich die Zahl zwischen zwei Doppelpunkten, meistens eine 3. Diese Zahl sagt Ihnen, in welchem Verzeichnis Sie eine Verknüpfung zu dem Start/Stop-Script anlegen müssen, also z.B. in /etc/init.d/rc3.d.

Gehen Sie in dieses Verzeichnis hinein ung geben Sie dort ein:

ln -s /etc/init.d/sms S82sms

Nun werden die SMS Server Tools beim Hochfahren des Servers automatisch gestartet.

Und das Herunterfahren brauchen Sie sich keine Gedanken zu machen. Das Betriebssystem bricht alle laufenden Programme beim Herunterfahren ab, kurz bevor sich der Rechner aus schaltet bzw. neu startet. Für die SMS Server Tools ist das völlig akzeptabel. Ein "sauberes" Beenden ist bei den SMS Server Tools nicht notwendig.

Windows NT, 2000, XP Anwender können die SMS Server Tools als Dienst automatisch beim Hochfahren starten lassen. Installieren Sie den Dienst mit folgendem Befehl, wenn sie das möchten:

cygrunsrv --install smsd --path /usr/local/bin/smsd.exe --type auto --neverexits --shutdown --env "CYGWIN=server" --env PATH=/usr/local/bin:/usr/bin:/bin" --desc "SMS Server Tools"

Starten Sie nun den Dienst durch einmalige Eingabe von **net start smsd** oder indem sie die Systemsteuerung benutzen.

Das Programm smsd kann Ihnen auch einen Statusmonitor anzeigen, während es läuft. Um diesen zu aktivieren, starten Sie smsd mit der Option -s. Im Kapitel 5.8 erfahren Sie, wie Sie diese Option benutzen.

5.4. Kurzmitteilungen senden

Die einfachste Art, eine Kurzmitteilung zu senden, geht mit Hilfe des Scriptes sendsms. Sie wenden es so an:

sendsms +491721234567 "Hallo, wie geht es Dir?"

oder

sendsms +491721234567
"Hallo, wie geht es Dir?"

Das Script kann auch mit Eingabeumleitung umgehen. Zum Beispiel so:

echo "Das Datum ist `date` " | sendsms +491721234567

Der echo Befehl erzeugt einen Text und fügt die Ausgabe vom date Befehl in den Text ein. Die Ausgabe dieses echo Befehls wird dann in sendsms umgeleitet.

Wenn Sie einen Blick in das Script sendsms werfen, werden Sie sehen, dass es gar nicht viel tut. Es legt nämlich nur eine Textdatei mit der Telefonnummer und dem Text in der ausgehenden Warteschlange an:

```
FILE=`mktemp /var/spool/sms/outgoing/send_XXXXXX`
echo "To: $DEST" >> $FILE
echo "" >> $FILE
echo "$TEXT" >> $FILE
```

In der ersten Zeile wird eine temporäre Datei erstellt und ihr Name in die Variable FILE gespeichert. Mktemp ist ein Standard-Unix Befehl, der die XXXXXX durch eine zufällige Buchstaben/Ziffernkombination ersetzt aber dabei auf Eindeutigkeit achtet. So bekommt jede gesendete Nachricht garantiert einen eigenen eindeutigen Dateinamen.

Die nachfolgenden echo Befehle schreiben die Telefonnummer, eine Leerzeile und den Text in die Datei.

Genau das können Sie auch selbst tun, und zwar mit jedem beliebigen selbstgeschriebenen Script oder auch mit einem ganz normalen Texteditor.

Eine korrekt geschrieben Datei sieht so aus:

```
To: 491721234567

Hallo
```

Vergessen Sie die Leerzeile nicht und denken Sie stets daran, die Telefonnummer im internationalen Format, zu schreiben. Im Gegensatz zur Konfigurationsdatei smsd.conf dürfen Sie hier alle Telefonnummern auch mit einem + beginnen lassen. Smsd entfernt es gegebenenfalls automatisch.

Speichern Sie solche SMS Dateien unter beliebigem Namen in das Verzeichnis /var/spool/sms/outgoing ab. Smsd interessiert sich nicht für den Dateinamen, sondern nur für den Inhalt, darum dürfen Sie jeden beliebigen Dateinamen wählen.

Sie sollten aber auf jeden Fall darauf achten, dass jede Kurzmitteilung einen eigenen eindeutigen Dateinamen bekommt. Sonst überschreiben Sie womöglich ungewollt eine noch nicht gesendete Kurzmitteilung, und diese geht dann verloren.

5.5. Kurzmitteilungen empfangen

Sie können Kurzmitteilungen nicht aktiv empfangen. Die SMS Zentrale (SMSC) schickt Ihnen einfach welche, sobald Ihr SGM Modem empfangsbereit ist. Sie müssen die empfangenen Nachrichten nur noch auslesen.

Diesen Job nehmen Ihnen die SMS Server Tools automatisch ab, sofern Sie in die Konfigurationsdatei in den Modemeinstellungen

```
incming = yes
```

geschrieben haben. Die empfangenen Nachrichten legt smsd alle in das Verzeichnis für die hereinkommende Warteschlange ab, das ist normalerweise /var/spool/sms/incoming.

Falls es für Sie wichtig ist, wann genau die Nachricht empfangen wurde oder welches Modem sie empfangen hat, dann schauen Sie doch einfach in die Datei hinein. Alle diese Informationen stehen nämlich in jeder empfangenen Datei:

```
From: 491721234567
From_SMSC: 491722270333
Sent: 00-02-21 22:26:23
Received: 00-02-21 22:26:29
Subject: GSM1

Hallo, wie geht es Dir?.
```

Die erste Zeile benennt den Absender der Nachricht. Die zweite Zeile zeigt an, welche SMS Zentrale Ihnen diese Nachricht zugestellt hat. Es folgen die Zeitstempel, wann die Nachricht vom Absender losgeschickt wurde und wann sie von Ihrem Computer empfangen wurde.

Die Subject Zeile benennt das Modem, welches die Nachricht empfangen hat. Der Autor des Programmes hat diese Zeile so genannt, weil man dann eine SMS sehr einfach als eMail weiter leiten kann.

5.6. SMS Dateiformat

SMS Dateien ähneln dem Aufbau von eMails. Sie beginnen mit einem Vorspann (Header) und enden mit dem Text. Dazwischen steht eine Leerzeile zur Trennung der beiden Abschnitte:

```
Vorspann

Text
```

Wie bereits erwähnt ist der Dateiname völlig beliebig, jedoch sollten Sie darauf achten, dass jede Nachricht einen eindeutigen Namen hat, der nicht mehrfach vorkommt.

Was genau alles im Vorspann stehen kann, hängt davon ab, was genau gesendet werden soll oder was Sie empfangen haben. Diese Details erfahren Sie auf den nächsten Seiten.

Die Reihenfolge der Angaben im Vorspann spielt keine Rolle. Manche Header enthaöten booleansche (binäre) Werte. In diesen Fällen bedeutet yes, true, on und 1 alle das gleiche, während alles andere als false (off) interpretiert wird.

Smsd ignoriert alle unbekannten Informationen im Vorspann. Sie können also beliebige Notizen hinzufügen, z.B.:

```
Anmerkung: Nur ein Test
```

Smsd kennt das Wort "Anmerkung:" nicht und ignoriert daher die gesamte Zeile.

Zeilenumbrüche dürfen wahlweise im Unix Format oder im Dos Format gespeichert werden. Bei empfangenen Nachrichten speichern die SMS Server Tools immer im Unix Format.

Der Unterschied zwischen Unix und Dos Zeilenumbrüchen ist:

Unix speichert einen Zeilumbruch mit einem Steuerzeichen, Code 10 (Dezimal).

Dos speichert einen Zeilumbruch mit zwei Steuerzeichen, Code 13 und Code 10.

Falls Sie unter Windows versuchen, empfangene SM mit Notepad zu lesen, werden SIe enttäuscht sein. Notepad kann keine Textdateien im Unix Format lesen. Nahezu jeder andere Texteditor kann es jedoch, z.B. Wordpad, Word und Editpad.

5.6.1. Texte senden

Wenn Sie eine SM senden, müssen Sie wenigstens die Telefonnummer des Empfängers und natürlich den Text angeben. Eine einfache Datei sieht so aus:

```
To: +491721234567

Hallo, wie geht es Dir?
```

Wie bereits erwähnt, müssen Sie alle Telefonnummern im internationalen Format schreiben. Das vorangestellte Plus Zeichen dürfen Sie dabei weglassen.

Wenn Sie möchten, dürfen Sie noch weitere Angaben in den Vorspann einfügen:

```
From: Stefan
To: 491721234567
Flash: yes

Hallo, wie geht es Dir?
```

From: Name oder Telefonnummer des Absenders. Dieses Feld hat derzeit keine Funktion für das Programm.

To: Telefonnummer des Empfängers im internationalen Format, aber ohne das + Zeichen. Wenn Sie eine Nachricht an eine Servicenummer senden wollen (z.B um einen Klingelton zu bestellen), schreiben Sie ein „s“ davor.

Flash: Booleanscher Wert. Wenn yes, dann erscheint die Nachricht sofort auf dem Display des Empfängers. Die meisten Telefone können das, aber leider nicht alle. Solche Nachrichten werden auch Alarm-SMS oder Klasse-0 SMS genannt.

Alphabet: Zeigt dem Programm, welcher Zeichensatz verwendet wurde. Mögliche Werte sind:

> **ISO** oder **Latin** oder **ANSI**: Normaler 8 Bit Zeichensatz, auch Ansi oder Latin-1 genannt. Alle drei Namen sind gleichwertig. Wenn der Text länger als 160 Zeichen ist, kann das Programm die Nachricht automatisch aufteilen.
>
> **GSM:** 7 Bit Zeichensatz, wie in der GSM Spezifikation beschrieben. Allerdings mit einer Ausnahme: Das @ Zeichen wird doch den Code 0xB7 dargestellt. Wenn der Text länger als 160 Zeichen ist, kann das Programm die Nachricht automatisch aufteilen.
>
> **UCS** oder **Chinese** oder **Unicode**: UCS2 Zeichensatz, maximal 70 Zeichen. Alle drei Namen sind gleichwertig. Der Vorspann muss immer mit einem 8 Bit Zeichensatz geschrieben werden, aber der Text danach muß in diesem Fall mit einem 16 Bit Zeichensatz, nämlich Unicode (big endian) geschrieben werden. Schauen Sie mal in das Verzeichnis scripts, dort werden Sie ein paar praktische Programme zur Konvertierung solche Nachrichten finden. Mehr Detail siehe weiter unten.
>
> Das Programm prüft nur die ersten 3 Zeichen dieses Feldes, so daß Wörter wie ISO-8859-15 oder UCS-2 genau so gut funktionieren.

Sie können die folgenden Zeilen verwenden, um Einstellungen aus der Konfigurationsdatei zu übergehen:

SMSC: Telefonnummer der SMS Zentrale

Provider: oder **Queue:** Name des Netzanbieters, kann verwendet werden, um den normalen Sortieralgorithmus gemäß [provider] und [queues] in der Konfigurationsdatei zu übergehen. Beide bewirken das Gleiche.

Report: Booleanscher Wert, legt fest, ob ein Statusreport erwünscht wird.

Autosplit: Booleanscher Wert, legt fest, ob das Programm größere Nachrichten als 160 Zeichen aufteilen soll.

5.6.2. Binärdateien senden

Binärdateien sind Nachrichten mit 8-Bit Daten. Wie diese auf dem Display des Empfänger angezeigt werden, hängt ganz vom Handy ab. Viele Handies kennen zum Beispiel besondere Dateiformate für Klingeltöne und Betreiberlogos.

Die SMS Server Tools haben keine besonderen Funktionen für Klingeltöne und Betreiberlogos. Sie können Binärdateien 1:1 an jedes Handy senden, völlig egal, welchen Inhalt die Dateien haben.

Wenn Sie Klingeltöne selbst komponieren möchten oder Bilder auf dem Computer in Handy-lesbare Betreiberlogos umwandeln möchten, benötigen Sie dazu andere Software. Die SMS Server Tools unterstützen Sie in keiner Weise beim Erzeugen solcher Dateien.

Wenn Sie sich weitergehend zu diesem Thema informieren wollen, sollten Sie sich die Spezifikationen der Binärdaten-Formate der gewünschten Handies besorgen. Jeder Hersteller hat da seine eigenen Dateiformate.

Binäre Kurzmitteilungen dürfen maximal 140 Bytes lang sein. Smsd zerlegt zu große Binärdateien nicht autmatisch!

Eine SMS Datei mit binärem Inhalt sieht so aus:

```
To: 491721234567
Alphabet: binary
UDH: true

gs2389gnsakj92fs2ujtiegbhewqu2ir9jsdgufh3
gjeruqgh87zt243htgerugsqh
3gert3245
```

Die Angabe "**Alphabet:**binary" teilt smsd mit, dass dies eine Kurzmitteilung mit 8 Bit Inhalt sein soll.

Die Angabe "**UDH:** true" oder "UDH: false" legt fest, wie das UDH Flag in der Kurzmitteilung eingestellt werden soll. Für die SMS Server Tools und für die SMS Zentrale hat das UDH Flag keine Bedeutung, doch manche Handies verlangen explizit, dass dieses Bit gesetzt oder eben nicht gesetzt sein soll.

Wenn Sie unsicher sind, setzen Sie es auf "yes", denn dies ist für die meisten Handies (z.B. Nokia) die richtige Einstellung. Wenn Sie diese Angabe weglassen, geht smsd automatisch von "true" aus.

Wie bei gesendeten Textnachrichten können Sie auch hier weitere Angaben im Vorspann machen, nämlich Absender, Netzanbieter und das Flash-Flag.

Die Binärdaten beginnen unmittelbar hinter der Leerzeile und enden am Dateiende.

5.6.3. Unicode Texte senden

Unicode ist ein 16-bit Zeichensatz, der neben lateinischen Buchstaben auch Chinesich, Arabisch und viele andere Sprachen beinhaltet. Er wird auch UCS2 genannt. Unicode gibt es in zwei Varianten, und zwar „big endian“ und „little endian“. Im GSM netz wird die „big endian“ variante verwendet. Unicode Nachrichten sind auch 70 Zeichen begrenzt.

Unicode Nachrichten sind prinzipiell genau so aufgebaut, wie binäre Nachrichten. Der Header ist in lateinischen Buchstaben geschrieben (iso8859-1), dann kommt eine Leerzeile und dann kommt der Nachrichtentext in Unicode „big endian“ Format:

```
To: 491721234567
Alphabet: UCS2

gs2389gnsakj92fs2ujtiegbhewqu2ir9jsdgufh3
gjeruqgh87zt243htgerugsqh
3gert3245
```

Damit Sie empfangene Unicode Nachrichten einfacher lesen können, habe ich zwei Scripte geschrieben. Das Script sms2html wandelt eine empfangene SMS Dateien in HTML um. Das Script sms2unicode wandelt eine empfangene SMS Datei in eine reine Unicode Textdatei um:

```
sms2html received.sms > new.html
sms2unicode received.sms > new.html
```

Für die umgekehrte Richtung habe ich das Script unicode2sms geschrieben. Es wandelt eine reine Unicode Textdatei in eine SMS Datei um:

```
unicode2sms message.txt > send.sms
```

Windows XP Usern empfehle ich das Programm Notepad. Es kann diese Unicode Textdateien korrekt anzeigen und erstellen.

5.6.4. Empfangene Texte

Empfangene Textnachrichten sehen so aus:

```
From: 491721234567
From_SMSC: 491722270333
Sent: 00-02-21 22:26:23
Received: 00-02-21 22:26:29
Alphabet: ISO8859-15
Subject: GSM1

This is the Text that I sent with my mobile phone.
```

Neue Programmversionen erzeugen eventuell einen grösseren Vorspann mit weiteren Informationen. Bedenken Sie dies, wenn Sie Programme schreiben, die den Vorspann auslesen.

Die "**From:**" Zeile gibt an, wer die Nachricht gesendet hat. Dies muss nicht unbedingt eine Telefonnummer sein, es gibt auch Nachrichten mit einem Absendernamen anstatt einer Telefonnummer. Handies und GSM Modems senden Nachrichten immer mit numerischer Absenderkennung.

Die "**From_SMSC:**" Zeile sagt Ihnen, welche SMS Zentrale Ihnen diese Nachricht zugestellt hat. Wie nützlich Sie diese Information finden, können Sie sich selbst ausdenken.

Als nächstes folgen zwei Zeitstempel. "**Sent:**" enthält den Zeitpunkt, wann die Nachricht gesendet wurde. Dieses Feld wird von der SMS Zentrale ausgefüllt. Verlassen Sie sich nicht allzusehr auf diesen Zeitstempel, denn die Uhr der SMS Zentrale könnte falsch eingstellt sein.

Hinter "**Received:**" schreiben die SMS Server Tools, wann Ihr Computer diese Nachricht empfangen hat. Die Genauigkeit dieser Angabe hängt also von der Uhr Ihres Computers ab.

Die "**Subject**:" Zeile benennt das Modem, welches die Nachricht empfangen hat. Je nach Anwendung kann diese Information völlig unwichtig sein oder elementar wichtig.

5.6.5. Empfangene Binärdateien

Empfangene Binärdateien sehen prinzipell genau so aus, wie empfangene Textnachrichten. Anstelle des Textes werden eben die Binärdaten in die Datei gespeichert und der Header enthält zusätzlich die beiden Angaben "Alphabet:binary" und "UDH: true" oder "UDH: false":

```
From: 491721234567
From_SMSC: 491722270333
Sent: 00-02-21 22:26:23
Received: 00-02-21 22:26:29
Subject: GSM1
UDH: true
Alphabet: binary

gs2389gnsakj92fs2ujtiegbhewqu2ir9jsdgufh3
gjeruqgh87zt243htgerugsqh
3gert3245
```

Wie bei gesendeten Binärdateien beginnen die Binärdaten unmittelbar hinter der Leerzeile und enden am Dateiende.

5.6.6. Empfangene Statusberichte

Wenn Sie beim Senden einer SM einen Statusreport beim SMSC beantragen, bekommen Sie für jede gesendete Nachricht mindestens einen solchen Bericht. In der Konfigurationsdatei smsd.conf steuern Sie dies mit der Variable "report = true".

Statusreports sind Nachrichten mit binärem Inhalt. Die SMS Server Tools übersetzen diese in lesbaren Text, um Ihnen die Auswertung zu erleichtern:

```
From: 491721234567
From_SMSC: 491722270333
Sent: 00-02-21 22:26:23
Received: 00-02-21 22:26:29
Subject: GSM1

SMS STATUS REPORT
Message_id: 117
Discharge_timestamp: 00-02-21 22:27:01
Status: 0,Ok,short message received by the SME
```

Der Vorspann hat den bekannten Aufbau. Er ist mit empfangenen Textnachrichten identisch.

Im Textabschnitt informiert Sie die erste Zeile darüber, dass es sich um einen Statusbericht handelt.

Die **Message_id** ist eine fortlaufende Nummer, die jedes GSM Modem seinen gesendeten Nachrichten vergibt. Im Statusbericht finden Sie diese Nummer wieder, damit Sie den Bericht eindeutig einer gesendeten Nachricht zuordnen können.

Doch diese Nummer birgt einen kleinen gemeinen Haken in sich. Sie beginnt nämlich bei 0 und wird fortlaufend bis 255 hoch gezählt. Die nächste SM hat dann wieder die Nummer 0. Durch den sehr kleinen Nummernkreis von 0-255 ist eine eindeutige Zuordnung nicht wirklich möglich.

Der **Discharge_Timestamp** gibt an, wann die Nachricht im Speicher der SMS Zentrale gelöscht wurde. Das tut sie immer dann, wenn sie erfolgreich ausgeliefert werden konnte oder als unzustellbar verworfen wird. Wenn die Nachricht noch nicht zugestellt wurde, aber weiterhin in der Warteschlange der SMS Zentrale verbleibt, enthält dieser Zeitstempel nur lauter Nullen.

Die letzte Zeile meldet den Status, und zwar zuerst als Zahl und danach als Text. Die SMS Server Tools melden den Text auf englisch, hier möchte ich ihn auf deutsch übersetzen:

0	Ok, die Nachricht wurde ausgeliefert
1	Ok, die Nachricht wurde warscheinlich ausgeliefert, aber das Endgerät hat den Empfang nicht bestätigt
2	Ok, die Nachricht wurde aber vom SMSC durch eine andere ersetzt
32	Versuche weiter, Netz ausgelastet
33	Versuche weiter, Endgerät ist nicht bereit
34	Versuche weiter, das Endgerät antwortet nicht
35	Versuche weiter, der Dienst wurde verweigert
36	Versuche weiter, Qualität des Dienstes ist nicht verfügbar
37	Versuche weiter, Fehler im Endgerät
64	Fehler, Prozedurfehler auf Gegenseite
65	Fehler, inkompatibles Ziel
66	Fehler, Verbindung wurde vom Endgerät verweigert
67	Fehler, nicht ermittelbar
68	Fehler, Qualität des Dienstes nicht verfügbar
69	Fehler: Keine Zusammenarbeit möglich
70	Fehler, Gültigkeitsdauer der SM abgelaufen
71	Fehler, Nachricht wurde vom Absender gelöscht
72	Fehler, Nachricht wurde vom SMSC Administrator gelöscht
73	Fehler, Die Nachricht existiert nicht
96	Fehler, Überlast
97	Fehler, Endgerät beschäftigt
98	Fehler, keine Antwort vom Endgerät
99	Fehler, Dienst verweigert
100	Fehler, Qualität des Dienstes nicht verfügbar

101	Fehler, Fehler im Endgerät

Bei allen anderen Codes ist der Text "unknown". Manche Texte mögen unsinnig erscheinen, doch dies ist eine wörtliche Übersetzung der Text, die die Spezifikation ETSI TS 100 900 vorschreibt.

Sie werden wohl am häufigsten die Codes 0 und 70 sehen.

5.6.7. Zeitstempel

In den vorherigen Seiten haben Sie mehrfach Zeitstempel gesehen. Diese haben immer folgendes Format:

YY-MM-DD HH:MM:ss

YY ist die Jahreszahl
MM ist der Monat
DD ist der Tag

HH ist die Stunde
MM sind die Minuten
ss sind die Sekunden

Alle Zahlen sind immer Zweistellig. Es kann vorkommen, dass ein Zeitstemmpel nur lauter Nullen beinhaltet, nämlich immer dann, wenn die Angabe in der Kurzmitteilung fehlt. Das sieht dann so aus.

00-00-00 00:00:00

Die SMS Server Tools verwenden diese Darstellungsart, weil die MySQL Datenbank genau die selbe Darstellung für unbekannte Datum/Uhrzeit Felder verwendet.

5.7. Modemauswahl

In diesem Kapitel möchte ich Ihnen bei der Wahl eines geeigneten GSM Modems helfen.

Die am häufigste gestellte Frage über die SMS Server Tools lautet "Welches Modem empfehlen Sie?". Die Antwort auf diese Frage ist sehr einfach, denn es gibt nicht viele GSM Modems auf dem Markt.

Alle Geräte der Marke Falcom mit serieller Schnittstelle haben sich als zuverlässig und einwandfrei herausgestellt. Als die Entwicklung der SMS Server Tools begann, gab es nur ein einziges Falcom Modem, das Modell A1. Inzwischen hat Falcom viele weitere Produkte im Angebot. Vom A1 bis zum Samba waren bisher alle Modelle einwandfrei geeignet, so daß ich davon ausgehe, daß auch künftige Modelle einwandfrei geeignet sein werden.

Das Falcom A2D und A3D kann mit einer GPS Antenne ausgestattet werden. Die Alarm-Firmware ermöglicht einen Betrieb ohne Computer und sendet Alarme, sowie GPS Positionen als SMS.

Bitte benutzen Sie keine Handies als Modem in kommerziellen SMS Anwendungen. Zum einen können Sie diese nicht ferngesteuert (z.B. durch das LAN) einschalten. Ausserdem sind Handies nicht für den Dauerbetrieb in automatischen Anwendungen gedacht. Die Zuverlässigkeit der Geräte ist in der Regel mangelhaft. Viele Handies, die als Mobiltelefon absolut zuverlässig sind, weisen beim Dauerbetrieb an einer seriellen Schnittstelle plötzlich erhebliche Mängel auf. dass sich fast jeder Problembericht auf ein Handy bezieht und kaum jemand mit GSM Modems Probleme hat, belegt dies.

Für die Auswahl eines GSM Modems sind diese Kriterien wichtig:

- Hat das Modem eine serielle Schnittstelle?
- Ist die Antenne absetzbar (FME Buchse vorhanden?)
- Unterstützt das Modem SMS Befehle nach GSM 07.05 oder GSM 07.07 Standard?
- Unterstützt das Modem den Empfang von Statusberichten?
- Kann das Modem Binärdateien (8bit SMS) senden und empfangen?
- Funktioniert das Modem zuverlässig?

5.8. Status Monitor

Die SMS Server Tools können Sie ständig über den Belegungsstatus der einzelnen GSM Modems informieren. Sie müssen smsd dazu mit der Option -s starten.

Nach dem Start zeigt smsd den Status der Modems an:

```
iiiriisssisss---
iiiriisssisss---
rriiiiissisis---
rriiiiiiiiii---
iiiiiiiiiiir---
```

Immer, wenn sich etwas ändert, erscheint eine neue Zeile. Die letzte Zeile ist dementsprechend immer die aktuelle.

Es gibt für jedes Modem einen Buchstaben. Die Modems werden in der selben Reihenfolge angezeigt, wie die in der Konfigurationsdatei in der "devices = " Zeile aufgelistet sind. Das obige Beispiel hat also 13 Modems.

s	Das Modem sendet eine SM (sending)
r	Das Modem empfängt eine SM (receiving)
i	Das Modem ist nicht aktiv (idle)
b	Das Modem ist wegen mehrfachen Fehlerversuchen blockiert (blocked)
-	Ist ein Platzhalter für nicht existente Modems.

Sie können die SMS Server Tools einfach an der Eingabeaufforderung mit

smsd -s

starten. Nach dem Start erscheinen die oben dargestellten Statusinformation in Form einfacher Textzeilen. Wenn smsd an der Eingabeaufforderun starten, können Sie übrigens wie üblich mit der Tastenkombination <Strg>-C stoppen.

Viele Anwender möchten das Programm aber sicher lieber beim Booten automatisch starten und benutzen daher vorzugsweise das Start/Stop Script. Nun darf das Programm aber keine ständigen Bildschirmausgaben machen, wenn Sie es als Hintergrundprogram starten - es soll sich eben schön ruhig im Hintergrund aufhalten.

Bei Bedarf möchten Sie dennoch den Modemstatus sehen können. Das ist auch möglich, sie brauchen nur das Start/Stop Script /etc/init.d/sms oder /sbin/init.d/sms ein klein wenig ändern:

```
...
    /usr/local/bin/smsd -s >/var/log/smsd.status &
...
```

Smsd schreibt nun die Statusinformationen in eine Textdatei mit dem Namen /var/log/smsd.status. Sie können diese Datei später jederzeit ansehen, vorzugsweise mit dem Befehl

tail -f -n1 /var/log/smsd.status

Dieser Tail Befehl zeigt ihnen fortlaufend die letzte Zeile der Protokolldatei an und wartet unendlich lange, dass die Datei grösser wird. Mit der Tastenkombination <Strg>-C können Sie die Anzeige stoppen.

Nun sollten Sie aber daran denken, dass diese Protkolldatei ständig beschrieben wird. Irgendwann wird Ihre Festplatte voll sein. Ein Cronjob kann dies verhindern, indem Sie die Protokolldatei täglich löschen. Tragen Sie diese Zeile in ihre Cron-Tabelle ein:

```
0 0 * * * cat /dev/null > /var/log/smsd.status
```

Dann werden das Statusprotokoll jede Nacht um 0:00 Uhr gelöscht, das sollte ausreichen.

Wie man mit Cron Jobs umgeht, haben Sie bereits in einem früheren Kapitel gelernt.

5.9. Ereignisprogramme

Ereignisprogramme werden von den SMS Server Tools immer dann gestartet, wenn

- eine SM empfangen wurde (RECEIVED)
- ein Statusreport empfangen wurde (REPORT)
- eine SM gesendet wurde (SENT)
- eine SM nicht gesendet werden konnte (FAILED)

Sie können Ereignisprogramme einsetzen, um auf diese Ereignisse besonders zu reagieren und dabei irgendeine Aktion auslösen. In den Anwendungsbeispielen am Ende dieses Buches werde ich Ihnen zeigen, wozu Ereignishandler gut sind.

Smsd installiert ein Beispiel-Script, welches folgendes tut: (smsevent):

- Es zeigt die Argumente auf dem Bildschirm an
- Es ändert die Zugriffsrechte für alle empfangenen SMS Dateien, so dass jeder die Dateien lesen darf
- Es leitet empfangene SM als eMail an den Administrator (root) weiter
- Es leitet alle empfangenen SM an die Telefonnummer 491721234567 weiter

Sie können dieses Script als Vorlage für eigene Programme benutzen.

Wenn die SMS Server Tools ein Ereignisprogramm aufrufen, übergeben sie dem Ereignisprogramm immer zwei oder drei Argumente:

$1	Art des Ereignisses (RECEIVED, REPORT, SENT oder FAILED)
$2	Name der SMS Datei
$3	ID-Nummer der SM (nur bei SENT).

Gesendete SMS Dateien werden nach dem Senden von der Netzanbieter Warteschlange entweder gelöscht, in die Warteschlange für gesendete Dateien verschoben oder in die Warteschlange für fehlgeschlagene Dateien verschoben. Das hängt von der Konfigurationsdatei ab und ob das Senden erfolgreich war.

Ereignisprogramme werden immer aufgerufen, bevor die Dateien verschoben oder gelöscht werden. Das zweite Argument $2 verweist demnach immer auf eine lesbare Datei in einer Netzbetreiber Warteschlange oder in der Warteschlange für empfangene Nachrichten.

Ein ganz einfaches Ereignisprogramm, das alle empfangenen SM an eine feste eMail Adresse weiter leitet, sieht z.B. so aus:

```
#!/bin/sh
if [ $1 = RECEIVED ]; then
   cat $2 | /usr/sbin/sendmail root@localhost
fi
```

5.10. Alarm Programme

Alarm Programme sind externe Programme, die von den SMS Server Tools gestartet werden, um Probleme zu melden. Betrachten Sie die Alarm Programme als Ergänzung zur Protokolldatei.

Sie könnten sich zum Beispiel eine eMail senden lassen oder eine Blitzlicht einschalten, wenn die SMS Server Tools ein Problem feststellen.

Viele große Firmen haben ein zentrales System zu Sammlung und Anzeige von Störmeldungen ihrer Maschinen. Mit Hilfe eines Alarm Programmes können Sie die SMS Server Tools sehr einfach an diese Systeme ankoppeln.

Bei welcher Wichtigkeits-Stufe Alarm Programme gestartet werden, bestimmen Sie durch die Zeile "alarmlevel = " in der Konfigurationsdatei smsd.conf.

Den Alarm Programmen werden die folgenden Argumente übergeben:

$1	Das Schlüsselwort ALARM
$2	Ein Datum im Format YYYY-MM-DD
$3	Die Uhrzeit im Format hh:mm:ss
$4	Die Wichtigkeit des Alarms (eine Ziffer)
$5	Der Name des betroffenen Modems oder SMSD
$6	Der Alarm Text (ist mit dem Text in der Protokolldatei identisch)

5.11. Schwarze und weiße Listen

Schwarze Listen sind Textdateien, in denen alle Telefonnummern aufgelistet werden, die keine SM erhalten dürfen. Sie aktivieren eine Schwarze Liste, indem Sie den Dateinamen der Liste in die Konfigurationsdatei smsd.conf eintragen:

```
blacklist = /etc/smsd.black
```

Wenn Sie keine schwarze Liste haben, darf jeder Kurzmitteilungen bekommen.

Weisse Listen sind Textdateien, in denen alle Telefonnummern aufgelistet werden, die SM empfangen dürfen.

```
whitelist = /etc/smsd.white
```

Wenn Sie eine weisse Liste haben, bekommen nur noch die dort gelisteten Telefonnummern SM zugestellt. Wenn Sie keine weisse Liste haben, darf jeder SM bekommen, abgesehen von denen, die in der schwarzen Liste stehen.

Was passiert, wenn eine Telefonnummer in beiden Listen steht? Dann ist die schwarze Liste wichtiger.

Neben Telefonnummern, dürfen die Listen auch Kommentare enthalten. Markieren Sie alle Kommentare, indem Sie ihnen ein Raute-Zeichen voran stellen. Beispiel:

```
#Dies ist die schwarze Liste
491721234567 # Der hier soll nie SM bekommen
491721111111 # will keine SM mehr bekommen. 7.4.2003
```

Die weisse und schwarze Liste sind sofort aktiv. Sie brauchen die SMS Server Tools nicht neu starten, wenn Sie die Listen verändert haben.

Wenn Ihre weisse oder schwarze Liste sehr groß ist, kann es aus Geschwindigkeitsgründen besser sein, einen Checkhandler zu benutzen, der die Gültigkeit der Telefonnummer mit einer SQL Datenbank kontrolliert. In den Anwendungsbeipspielen am Ende dieses Buches werde ich darauf näher eingehen.

5.12. Statistik Dateien

Statistik Dateien werden auf Wunsch in regelmässigen Zeitabständen von smsd geschrieben. Sie infomieren über die Auslastung der Modems sowie die Anzahl der gesendeten und empfangenen SM. Beispiel:

```
runtime,rejected
1200,1

name,succeeded,failed,received,multiple_failed,usage_s,usage_r
GSM1,20,0,1,0,80,0
GSM2,5,0,1,0,40,900
```

Diese Beispieldatei wurde von einem System mit zwei GSM Modems geschrieben. Das Dateiformat erlaubt ein einfaches Einlesen in eine Tabellenkalkulation wie z.B. Microsoft Excel oder KSpread.

Es gibt immer eine Zeile mit globalen Zählern, sowie für jedes GSM Modem eine Zeile mit Modemspezifischen Zählern.

Globale Zähler:

- Laufzeit des Programmes in Sekunden seit der letzten Statistik-Datei
- Anzahl der verweigerten SM, z.B. wegen schwarzer Telefonnummer

Zähler pro Modem:

- Name des Modems
- Anzahl erfolgreich gesendete SM
- Anzahl fehlgeschlagene SM
- Anzahl empfangene SM
- Anzahl mehrfacher Fehler (Modem blockiert)
- Anzahl der Sekunden, die das Modem mit Senden beschäftigt war
- Anzahl der Sekunden, die das Modem mit Empfangen beschäftigt war

Alle Zahlen werden auf Null zurück gesetzt, sobald sie in eine Datei geschrieben wurden. Der Dateiname wird aus dem Datum und der Uhrzeit zusammengestellt, im Format YYMMDD.hhmmss.

YY	Jahreszahl
MM	Monat
DD	Tag
hh	Stunde
mm	Minute
ss	sekunde

Die SMS Server Tools stellen sicher, dass Statistiken immer zur gleichen Zeit geschrieben werden. Wenn Sie in der Konfigurationsdatei beispielsweise festlegen, dass smsd jede Stunde eine Datei schreiben soll (3600 Sekunden), dann geschieht dies immer zur nächsten vollen Stunde, auch wenn Sie das Programm irgendwann zwischen zwei vollen Stunden gestartet haben.

Smsd geht sehr sorgsam mit Ihren Statistiken um. Im Gegensatz zu den meisten anderen Programmen, verlieren Sie keine Zahlen, wenn Sie das Programm beenden. Dies gilt auch für den Kill Befehl, solange Sie nicht die ganz grobe Methode "kill -9" benutzen.

Wenn Sie das Programm beenden, merkt es sich die aktuellen Zahlen in einer temporären Datei. Beim nächsten Mal, wenn es wieder Statistiken schreibt, addiert es diese Werte dann dazu. Ihnen gehen also keine Zähler verloren.

Bitte beachten Sie, dass neuere Programmversionen möglicherweise zusätzliche Zähler besitzen. Diese werden dann an das rechte Ende angehangen. Wenn Sie Programme oder Makros schreiben, die Statistiken auslesen, kann dies wichtig sein.

5.13. Protokollierung in SQL Datenbank

Die Protokollierung in eine SQL Datenbank ist besonders bei kommerziellen SMS Diensten nützlich. Sie ermöglicht nicht nur das Abrechnen der Dienstleitung, sondern auch die Kontrolle, ob alle SM ausgeliefert wurden.

Die Protokolldatenbank enthält eine Liste aller gesendeten und empfangenen SM. Bei gesendeten SM wird ausserdem der Status eingetragen, den die SMS Zentrale mittels Statusberichte zurückgemeldet hat.

Die Protokollierung in eine SQL Datenbank ist eigentlich keine direkte Funktion der SMS Server Tools. Sie wurde mit einem Shell Script realisiert, dass man als Eventhandler installiert. Wenn Sie andere Verknüpfungen zwischen den SMS Server Tools und einer MySQL Datenbank entwickeln möchten, können Sie dieses Script hervorragend als Vorlage benutzen, denn es ist nicht besonders kompliziert.

Dieses Shellscript heisst mysmsd und wird bei der Installation der SMS Server Tools nach /usr/local/bin kopiert. Öffnen Sie das Script bitte in Ihrem bevorzugtem Editor und passen Sie die Variablen am Dateianfang an:

```
SQL_HOST=localhost
SQL_USER=root
SQL_PASSWORD=""
SQL_DATABASE=smsd
SQL_TABLE=sms_log
```

SQL_HOST ist der Rechnername, auf dem die MySQL Datenbank installiert wurde.

SQL_USER ist der Username, mit dem sich das Script an die Datenbank anmelden soll.

SQL_PAssWORD ist das Passwort, mit dem sich das Script an die Datenbank anmelden soll. Schreiben Sie das Passwort zwischen die Anführungsstriche, dass Sie bei der Installation der Datenbank festgelegt haben.

SQL_DATABASE ist der Name der Datenbank, die von den SMS Server Tools verwendet werden soll.

SQL_TABLE ist der Name der Tabelle, in die die Protokolle eingetragen werden sollen.

Normalerweise brauchen Sie nur das Passwort und die Hostnamen anpassen. Editieren Sie die Konfigurationsdatei /etc/smsd.conf und tragen Sie dort einen Eventhandler ein:

```
eventhandler = /usr/local/bin/mysmsd
```

Kontrollieren Sie, ob die folgenden Programme installiert sind, denn das Script benötigt sie:

- mysql client
  formail
  sed
  cut

Jetzt müssen Sie die Datenbank anlegen und eine leere Tabelle, die später mit den Protokolldaten gefüllt wird. Die Tabelle soll folgende Struktur haben:

```
+-------------+----------+------+-----+---------+----------------+
| Field       | Type     | Null | Key | Default | Extra          |
+-------------+----------+------+-----+---------+----------------+
| id          | int(11)  |      | PRI | NULL    | auto_increment |
| type        | char(16) | YES  |     | NULL    |                |
| sent        | datetime | YES  |     | NULL    |                |
| received    | datetime | YES  |     | NULL    |                |
| sender      | char(32) | YES  |     | NULL    |                |
| receiver    | char(32) | YES  |     | NULL    |                |
| status      | char(3)  | YES  |     | NULL    |                |
| msgid       | char(3)  | YES  |     | NULL    |                |
+-------------+----------+------+-----+---------+----------------+
```

Dies erreichen Sie durch die Eingabe der folgenden Befehle:

```
stefan@server> mysql -u root -p
mysql> create database smsd;
Query Ok...
mysql> use smsd;
Database changed.
mysql> create table sms_log (
    -> id int auto_increment not null,
    -> primary key(id),
    -> type char(16),
    -> sent datetime,
    -> received datetime,
    -> sender char(32),
    -> receiver char(32),
    -> status char(3),
    -> msgid char(3)
    -> );
Query Ok...
```

Sie haben nun alle Vorbereitungen getroffen und können die SMS Server Tools stoppen und neu starten, um die Änderungen wirksam werden zu lassen. Ab sofort trägt smsd mit Hilfe des Scriptes mysmsd alle gesendeten und empfangenen SM in diese Tabelle ein.

Nach drei Aktionen könnte die Tabelle z.B. so aussehen:

```
+---+--------+-------------------+-------------------+-------------+-------------+-------+------+
|id |type    |sent               |received           |sender       |receiver     |status |msgid |
+---+--------+-------------------+-------------------+-------------+-------------+-------+------+
| 1 |RECEIVED|2000-02-21 22:26:23 |2002-06-06 12:16:23|491721234567 |MODEM1       |  NULL | NULL |
| 2 |SENT    |2002-06-06 12:16:34 |2002-06-06 12:16:59|somebody     |491721234567 |     0 |  117 |
| 3 |FAILED  |2002-06-06 12:16:48 |NULL               |somebody     |491721234567 |  NULL | NULL |
+---+--------+-------------------+-------------------+-------------+-------------+-------+------+
```

Die erste Reihe sagt Ihnen, dass Sie mit dem Modem MODEM1 eine SM um 22:26 Uhr empfangen haben. Sie wurde um 12:16 vom Absender 01721234567 abgeschickt.

Die zweite Reihe sagt Ihnen, dass Sie um 12:16 eine SM an 01721234567 gesendet haben und dass diese um 12:16 erfolgreich ausgeliefert wurde (status=0). Der Absendername somebody kommt von der From: Zeile in der SMS Datei.

Die dritte Reihe sagt Ihnen, dass Sie um 12:16 eine SM nach 01721234567 senden wollten, aber das Senden war erfolglos.

Nun zeige ich Ihnen beispielhaft einige SQL Befehle, mit denen Sie die Datenbank abfragen können:

Um die gesamte Tabelle abzrufen, geben Sie im mysql Client

select * from sms_log;

ein. Um alle Nachrichten an eine bestimmte Zielrufnummer zu zählen, geben Sie ein:

select count(*) from sms_log where type="SENT" AND receiver="491722056395";

Der Befehl

select * from sms_log where type="SENT" AND receiver="491721234567";

Zeigt diese dann an, anstatt sie zu zählen. Der nächste Befehl listet alle empfangenen Nachrichten auf:

select * from sms_log where type="RECEIVED";

5.14. Mysmsd erklärt

In diesem Kapitel erkläre ich Ihnen, wie das mysmsd Script funktioniert.

smsd startet das Script mysmsd jedesmal, nachdem eine SM gesendet oder empfangen wurde. Der Befehl dazu lautet:

/usr/local/bin/mysmsd SENT Dateiname ID-Nummer
/usr/local/bin/mysmsd FAILED Dateiname
/usr/local/bin/mysmsd RECEIVED Dateiname
/usr/local/bin/mysmsd FAILED Dateiname

Der Dateiname ist der Name der betroffenen SMS Datei.

```
#!/bin/sh
SQL_HOST=localhost
SQL_USER=root
SQL_PAssWORD=""
SQL_DATABASE=smsd
SQL_TABLE=sms_log

DATE=`date +"%Y-%m-%d %H:%M:%S"`
FROM=`formail -zx From: < $2 | sed 's/"//g'`
TO=`formail -zx To: < $2`
SUBJECT=`formail -zx Subject: < $2`
SENT=`formail -zx Sent: < $2`

if [ "$SQL_PAssWORD" != "" ]; then
  SQL_ARGS="-p$SQL_PAssWORD";
else
  SQL_ARGS="";
fi

SQL_ARGS="-h $SQL_HOST -u $SQL_USER $SQL_ARGS -D $SQL_DATABASE -s -e"

if [ "$1" = "FAILED" ] || [ "$1" = "SENT" ]; then

   mysql $SQL_ARGS "insert into $SQL_TABLE (type,sent,sender,receiver,msgid) values
(\"$1\",\"$DATE\",\"$FROM\",\"$TO\",\"$3\");";

elif [ "$1" = "RECEIVED" ]; then

   mysql $SQL_ARGS "insert into $SQL_TABLE (type,sent,received,sender,receiver) values
(\"RECEIVED\",\"$SENT\",\"$DATE\",\"$FROM\",\"$SUBJECT\");";

elif [ "$1" = "REPORT" ]; then

   DISCHARGE=`sed -e 1,7d < $2 | formail -zx Discharge_timestamp:`
   MSGID=`sed -e 1,7d < $2 | formail -zx Message_id:`
   STATUS=`sed -e 1,7d < $2 | formail -zx Status: | cut -f1 -d,`

   ID=`mysql $SQL_ARGS "select id from $SQL_TABLE where receiver=\"$FROM\" and type=\"SENT\" and
msgid=\"$MSGID\" order by id desc limit 1;"`

   mysql $SQL_ARGS "update $SQL_TABLE set received=\"$DISCHARGE\",status=\"$STATUS\" where
id=\"$ID\";"

fi


      DATE=`date +"%Y-%m-%d %H:%M:%S"`
```

Der Date Befehl zeigt das aktuelle Datum im angegebenen Format an. Durch Ausgabeumleitung wird das Datum in die Variable DATE geschrieben.

```
      FROM=`formail -zx From: < $2 | sed 's/"//g'`
```

$2 ist der Name der SMS Datei. Formail holt aus dieser Datei den Absendernamen oder die Absendernummer heraus. Dazu wird die Eingabe von Formail durch den Pfeil "<" in die Datei $2 umgeleitet, denn Formail kann nur von dem Eingabekanal lesen.

Der Absendername befindet sich im Vorspann hinter dem Schlüsselwort "From:".Mit sed werden unerwünschte Anführungsstriche entfernt, falls vorhanden. Das Ergebnis wird mittels Ausgabeumleitung in die Variable FROM geschrieben.

```
      TO=`formail -zx To: < $2`
```

Mit der gleichen Methode wird die Telefonnummer des Empfängers aus der SMS Datei extrahiert.

```
      SUBJECT=`formail -zx Subject: < $2`
```

Auch die Betrifft-Zeile wird mit der selben Methode entnommen. Bei empfangenen SM enthält sie den Namen des Modems, dass die SM empfangen hat.

```
      SENT=`formail -zx Sent: < $2`
```

Und wieder wird nach bekannter Methode ein Feld aus der SMS Datei extrahiert, nämlich der Zeitstempel, wann die SM gesendet wurde.

Wir haben nun die folgenden Informationen:

DATE	Aktuelles Datum mit Uhrzeit
FROM	Absender
TO	Empfänger
SUBJET	Name des Modems
SENT	Datum und Uhrzeit, wann die SM gesendet wurde

```
if [ "$SQL_PAssWORD" != "" ]; then
  SQL_ARGS="-p$SQL_PAssWORD";
else
  SQL_ARGS="";
fi
```

Die if-Kontrollstruktur prüft, ob ein Passwort gesetzt wurde. Wenn ja, dann wird in der Variable SQL_ARGS die Option gespeichert, mit der man dem MySQL Client mitteilt, welches Passwort er gefälligst benutzen soll.

Wenn kein Passwort festgelegt wurde, dann bleibt SQL_ARGS erstmal leer.

```
SQL_ARGS="-h $SQL_HOST -u $SQL_USER $SQL_ARGS -D $SQL_DATABASE -s -e"
```

Nun werden weitere Optionen für den MySQL Client zusammengestrickt, nämlich Hostname, Username, Passwort (was zuvor in $SQL_ARGS stand) und Name der Datenbank.

An dieser Stelle steht fest, wie der MySQL Client gestartet wird, um SQL Abfragen an die Protokolldatenbank zu senden, und zwar so:

```
mysql $SQL_ARGS "..."
```

Anstelle der drei Pünktchen muss nun nur noch der SQL Befehl eingesetzt werden.

Jetzt kommt es darauf an, was für ein Ereignis aufgetreten ist. Beim Senden, Empfangen und bei Status Reports muss jeweils eine anderer SQL Befehl abgesetzt werden.

```
if [ "$1" = "FAILED" ] || [ "$1" = "SENT" ]; then
    ...;
elif [ "$1" = "RECEIVED" ]; then
    ...;
elif [ "$1" = "REPORT" ]; then
    ...;
fi
```

Bei gesendeten und fehlgeschlagenen Sendeversuchen kann der gleiche SQL Befehl abgesetzt werden. Bei Empfangenen SM wird ein zweiter SQL Befehl abgesetzt und bei Statusberichten wird der dritte SQL Befehl ausgeführt. Die if-Kontrollstruktur erfährt durch $1, welches der möglichen Ereignisse aufgetreten ist.

Erfolgreich und erfolglos gesendete SM werden mit dem selben SQL Befehl in die Datenbank eingetragen:

```
insert into $SQL_TABLE (type,sent,sender,receiver,msgid)
            values ("$1","$DATE","$FROM","$TO","$3");
```

Im Quelltext des Shell Scriptes werden Sie vor jedem Anführungsstrich einen Backslash sehen, denn Anführungsstriche innerhalb der selben Anführungsstriche sind nicht erlaubt.

Es wird also eine neue Reihe in die Tabelle eingefügt, mit folgenden Werten:

type	SENT oder FAILED
sent	Aktuelles Datum
sender	Absendername oder Nummer
receiver	Empfängernummer
msgid	ID-Nummer der Nachricht

Empfangene SM werden etwas anders in die Tabelle eingetragen, nämlich so:

```
insert into $SQL_TABLE (type,sent,received,sender,receiver)
            values ("RECEIVED","$SENT","$DATE","$FROM","$SUBJECT");
```

Es wird also eine neue Reihe in die Tabelle eingefügt, mit folgenden Werten:

type	RECEIVED

sent	Sendedatum und Uhrzeit
received	Aktuelles Datum
sender	Absendername oder Nummer
receiver	Name des Modems

Status Reports enthalten noch weitere Informationen, die bisher noch nicht aus der SMS Datei extrahiert wurden. Dies wird an dieser Stelle erledigt, wo klar ist, dass es sich um einen Statusreport handelt:

```
DISCHARGE=`sed -e 1,7d < $2 | formail -zx Discharge_timestamp:`
MSGID=`sed -e 1,7d < $2 | formail -zx Message_id:`
STATUS=`sed -e 1,7d < $2 | formail -zx Status: | cut -f1 -d,`
```

Der erste Befehl ermittelt die Zeitangabe, wann das SMSC die Nahricht ausgeliefert hat. Die zweite Zeile ermittelt die ID-Nummer, auf die sich der Statusbericht bezieht. Die dritte Zeile ermittelt den Statuscode.

Neu ist hier die Verwendung von Sed. Das ist nötig, denn Formail kümmert sich nur um den Vorspann-Abschnitt innerhalb der SMS Datei. Die Statusinformationen finden Sie aber nicht im Vorspann, sondern im Textabschnitt.

Der Befehl

```
sed -e 1,7d < $2
```

liest die SMS Datei ein und übergibt sie dem Sed Programm als Eingabe. Sed entfernt davon die ersten sieben Zeilen (1,7d). Was übrig bleibt, ist der Textabschnitt der SMS Datei, und nun können wir Formail darauf loslassen, um wie gewohnt die drei interessanten Felder zu extrahieren.

Die Ausgabe von Sed wird also durch das "|" Zeichen an Formail umgeleitet.

Der dritte Befehl beinhaltet noch eine weitere Umleitung, und zwar nach

```
cut -f1 -d,
```

Der Status Code steht nämlich zusammen mit dem Statustext in einer einzigen Zeile, z.B so:

```
0,Ok,short message received by the SME
```

Der Cut Befehl zerlegt die zeile in Felder und liefert davon nur das erste Feld (-f1). Als Trennzeichen wird logischerweise das Komma angegeben (-d,).

Nun haben wir alle Statusinformationen Zusammen, und zwar:

DISCHARGE	Wann die SM ausgeliefert wurde
STATUS	Der Fehlercode oder 0 bei erfolgreicher Auslieferung
MSGID	Die ID-Nummer der Nachricht, auf die sich der Statusbericht bezieht.

Statusberichte werden nicht einfach in die Tabelle eingefügt. Für die zuvor gesendete SM gibt es ja bereits eine Zeile in der Tabelle. Unsere Aufgabe ist nun, herauszufinden, welche Zeile das genau ist, und dann den Statuscode in dieser Zeile zu ergänzen.

Wir brauchen nicht die ganze Tabelle manuell zu durchsuchen, sondern befragen einfach MySQL, welche Zeile die gesucht ist. Kriterium ist, dass die ID-Nummer der gesendeten SM mit der ID-Nummer im Statusbericht überein stimmen. Also lautet der SQL Befehl:

```
select id from $SQL_TABLE where receiver="$FROM"
            and type="SENT"
            and msgid="$MSGID"
            order by id desc limit 1;
```

Auf deutsch: Gib mir die ID aus der Tabelle sms_log, wo der Empfänger mit dem Statusbericht überein stimmt und die Nachricht gesendet wurde und die Message-ID-Nummer mit dem Statusbericht überein stimmt. Falls mehrere Zeilen in Frage kommen, gib mir nur die letzte davon.

Verwechseln Sie jetzt bitte die ID Nummern nicht. Dieser SQL Befehl fragt die ID-Nummer der Tabellenzeile ab und vergleciht die ID-Nummern der SM mit dem Statusreport.

Die ID-Nummer der Tabellenreihe hat nichts mit der Message-ID-Nummer der SM zu tun. Sie bezeichnet einfach nur die Reihennummer in der Tabelle.

Nachdem wir wissen, welche Tabellenreihe die gesendete SM hat, für die wir den Statuscode ergänzen wollen, können wir den nächsten und letzten SQL Befehl absetzen:

```
update $SQL_TABLE set received="$DISCHARGE",
            status="$STATUS"
            where id="$ID";
```

6. Sinnvolle Erweiterungen

Sie haben nun kenngelernt, welche Programme Sie zum Versand und für den Empfang von Kurzmitteilungen benötigen.

Sie können die Programme installieren und anwenden.

In diesem Kapitel zeige ich Ihnen nun, wie Sie die Funktionen der SMS Server Tools sinnvoll erweitern. Der Inhalt dieses Kapitels basiert auf Anfragen von Anwendern der SMS Server Tools. Es werden immer wieder die gleichen Fragen nach Erweiterungsmöglichkeiten gestellt.

Das Programm biete ich mit dem Basis-Funktionsumfang kostenlos an. Doch nur die Leser dieses Buches erfahren von mir, wie sie noch mehr aus dem Programmpaket heraus holen.

6.1. Wartungsfreier Dauerbetrieb

Die meisten Softwarehäuser vergessen beim Aufbau eines Server-Systems, dass der Kunde in der Regel einem wartungsfreien Betrieb erwartet. Der Kunde möchte nur Hand anlegen, wenn Hardware defekt ist oder sonst irgendeine unerwartete Störung auftritt. Daraus ergibt sich die Forderung:

Solange das System korrekt funktioniert, muss es beliebig lange so weiter laufen, ohne dass regelmässige Wartungsarbeiten anfallen.

Obwohl die Erfüllung dieses Kriteriums weder ein Problem ist, noch mit hohem Aufwand verbunden ist, wird es immer wieder ausser Acht gelassen.

Meistens scheitert der wartungsfreie Betrieb daran, dass irgendwelche temporären Dateien oder Protokolldateien langsam eine Festplattenpartition füllen. Irgendwann ist die Festplatte voll und dies wiederum führt normalerweise zu einem Funktionsausfall.

Manche Programme sind mangelhaft geschrieben. Sie stürzen spontan ab oder verbrauchen im Laufe der Zeit ganz langsam und kaum bemerkbar immer mehr Arbeitsspeicher. Dies führt dazu, dass der Server irgendwann plötzlich langsamer wird und bei völliger Erschöpfung des Speichers ausfällt.

Gelegentlich habe ich auch beobachtet, dass Programme nicht unbegrenzt lange laufen können, weil irgendwann Zähler überlaufen oder das Programm mit der Sommer/Winterzeitumstellung nicht zurecht kommt. Überhaupt ist das Verstellen der Systemuhr interessanterweise für viele Programme ein erhebliches Problem.

Abgesehen, von den Problem mit Uhr-Umstellungen waren auch die SMS Server Tools vor allen diesen Mängeln nicht verschont geblieben. Sie wurden aber vollständig behoben - bis auf ein Mangel: Die Festplatte wird irgendwann voll.

6.1.1. Protokolldateien pflegen

Für das Vollaufen der Festplatte gibt es mehrere Gründe, gegen die Sie jeweils einzeln vorgehen müssen:

- Protokolldateien werden unendlich groß
- Gesendete SM werden unendlich lange aufbewahrt
- Empfangene SM werden unendlich lange aufbewahrt
- Fehlgeschlagene SM werden unendlich lange aufbewahrt
- Die SQL Datenbank wird unendlich groß

Auf den folgenden Seiten beschreibe ich, was Sie dagegen tun können.

Über das Betriebssystem brauchen Sie sich übrigens keine Gedanken zu machen. Alle in diesem Buch genannten Betriebssysteme, also SuSE Linux, RedHat Linux, Solaris oder Windows pflegen ihre Protokolldateien automatisch und begrenzen deren Speicherplatzbedarf.

Die Programme, die Sie zusätzlich zum Betriebssystem installieren, sollten Sie aber genau unter die Lupe nehmen, insbesondere den Apache Webserver. Dessen Logfiles haben schon so manchen Server lahm gelegt.

Protokolldateien werden von den meisten Programmen unendlich lange beschrieben. So ist es auch beim Apache Webserver und bei den SMS Server Tools. Der erste Lösungsansatz ist sicherlich, diese Dateien täglich umzubenennen und neuen Dateien zu beginnen.

Doch das funktioniert so nicht!

Denn die laufenden Programme öffnen Protokolldateien in der Regel nur ein einziges mal beim Programmstart und erwarten, fortlaufend in sie hinein schreiben zu können. Daher erlaubt Windows Ihnen gar nicht erst, die Datei umzubenennen oder zu löschen, solange sie geöffnet ist.

Unter Unix verhält es sich beim umbenennen einer Datei etwas anders, aber auch nicht besser. Wenn Sie dort eine Datei umbenennen, während Sie geöffnet ist, bemerkt das laufende Programm davon nichts und schreibt munter in die gleiche Datei weiter. Die Tatsache, dass sich zwischenzeitlich deren Dateiname geändert hat, spielt keine Rolle. Ihre Datei wird also nach wie vor langsam immer grösser. Und wenn Sie die Datei einfach löschen, dann wird Sie lediglich aus dem Inhaltsverzeichnis verschwinden, aber sie existiert in Wirklichkeit weiterhin und wird immer noch wachsen.

dass meine obige Behauptung stimmt, können Sie so nachvollziehen:

Öffnen Sie zwei Shell Fenster. Geben Sie im ersten Fenster dies ein:

while true; do sleep 1; echo "Hallo"; done >>test.txt

Dies ist eine Endlosschleife, die jede Sekunde das Wort "Hallo" in die Datei test.txt schreibt. Die Datei bleibt während des gesamten Vorganges permanent geöffnet.

Beobachten Sie, dass die Datei ständig wähst, indem Sie in dem zweiten Fenster mehrmals

ls -l test.txt

eingeben.

Versuchen Sie jetzt unter Windows, diese Datei mit dem Dateimanager (Explorer) zu löschen oder umzubenennen. Windows wird es Ihnen nicht erlauben, denn die Datei ist ständig geöffnet.

Versuchen Sie jetzt unter Unix, diese Datei umzubenennen:

mv test.txt test2.txt

Kontrollieren Sie die Dateigrösse wiederum mit dem ls Befehl. Sie wird weiterhin immer grösser. Löschen Sie die Datei nun unter Unix mit dem Befehl

rm test2.txt

Die Datei erscheint nun nicht mehr im Inhaltsverzeichnis, aber die Endlosschleife schreibt weiterhin auf die Festplatte. Irgendwann wird sie voll sein. Leider gibt es keinen Befehl, mit dem Sie das jetzt sehen können. Selbst der Befehl "df -k" zeigt Ihnen nicht an, dass die Festplatte voll wird, weil er sich auf das Inhaltsverzeichnis bezieht, und in diesem existiert die Datei test2.txt ja bereits nicht mehr. Wenn Sie die Led-Anzeige der Festplatte aufmerksam beobachten, bemerken Sie vielleicht, dass sie ständig aktiv ist.

Eine andere Lösung muss also her, und die verrate ich Ihnen jetzt.

Jeder, der sich Unix Administrator nennt sollte diese Methode kennen, denn Sie ist allgemein immer anwendbar. Auch die Windows Administratoren kann ich an dieser Stelle beruhigen, denn die hier genannte Lösung funktioniert auch unter Windows innerhalb der CygWin Shell.

- Zunächst kopieren Sie die Protokolldatei in eine neue Datei um.
- Dann kopieren Sie /dev/null in die Protokolldatei hinein.

Das Gerät /dev/null liefert einfach "nichts", daher der Name. Durch das Kopieren von "nichts" in die Protokolldatei löschen Sie deren Inhalt. Die Datei existiert also weiterhin, doch sie ist nun leer. Das funktioniert sowohl unter Unix, als auch unter Windows.

In der Praxis automatisiert man diesen Vorgang durch einen Cron-Job, z.B. so:

```
10 0 * * * cp /var/log/smsd.log /var/log/smsd.log.old; cp /dev/null /var/log/smsd.log
```

Unter Windows müssen Sie den Cronjob so schreiben (bitte alles in eine einzige Zeile):

```
10 0 * * * c:\cygwin\bin\bash -c "cp /var/log/smsd.log /var/log/smsd.log.old; cp
/dev/null /var/log/smsd.log"
```

Sie müssen unter Windows **bash -c "..."** schreiben, weil es sich um CygWin Befehle handelt. Ohne diese Umschreibung kann nnCron Lite nur Dos oder Windows Befehle ausführen.

So wird die Protokolldatei jede Nacht um 0:10 Uhr gekürzt. Die alten Daten des Vortages befinden sich immer im smsd.log.old.

Sie wundern sich jetzt vielleicht, warum ich gerade 0:10 Uhr gewählt habe und nicht 0:00 Uhr. Ich vermeide Cron-Jobs zu exakten vollen Stunden, insbesondere um 0:00 Uhr, denn das Betriebssystem selbst pflegt zu dieser Zeit seine eigenen Protokolldateien. Je mehr Aktionen der Computer zur selben Zeit ausführt, um so langsamer wird er in dieser Zeit.

Ausserdem würde sich der Fragmentierungsgrad der Festplatte unnötig erhöhen. Fragmentierung entsteht immer, wenn mehrere Dateien zugleich wachsen oder kleiner werden. Fragmentierung reduziert die Geschwindigkeit der Festplatte. Machen Sie sich darüber jetzt aber bitte nicht übermässig Kopfzerbrechen. Vor etwa 10 Jahren war dieses Thema noch wirklich interessant, weil die Festplatten erheblich langsamer waren, als heute. Das Dateisystem von MS-Dos (FAT) ging auch nicht gerade sorgsam mit der Fragmentierung um und erzwang früher oder später die Benutzung einer Defragmentierungs-Software.

Inzwischen sind die Festplatten aber sehr viel schneller geworden und die Betriebssysteme wurden verbessert. Fragmentierung stellt heute nur noch äusserst selten ein Performanceproblem dar.

Der oben gezeigte Cron-Job stellt sicher, dass die Protokolldatei der SMS Server Tools nicht unendlich groß wird. Sie können jederzeit auf die Protokolle des laufenden Tages sowie auf den Vortag zugreifen. Doch was ist, wenn Ihnen ein Tag nicht ausreicht?

Schreiben Sie sich das folgende Script ab und führen Sie es als Cron-Job jede Nacht einmal auf (logfiles.sh):

```
#!/bin/sh

rename()
{
  # $1 is a filename
  # $2 is the number of days to keep
  rm $1.$2 >/dev/null 2>&1
  num=$2
  while [ $num -gt 1 ]; do
    before=`expr $num - 1`
    mv $1.$before $1.$num >/dev/null 2>&1
    num=$before
  done
  cp $1 $1.1
  cp /dev/null $1
}

rename /var/log/smsd.log 7
rename /usr/local/apache/logs/access_log 2
rename /usr/local/apache/logs/error_log 7
```

Der Eintrag in der Cron-Tabelle sieht unter Unix demnach so aus:

```
10 0 * * * /usr/local/bin/logfiles.sh
```

Windows Anwender müssen den Eintrag in die Datei so schreiben:

```
10 0 * * * c:\cygwin\bin\bash /usr/local/bin/logfiles.sh
```

Dieses Script benennt die drei am Ende genannten Protokolldateien um. Es bewahrt dabei jeweils 7 bzw. 2 alte Dateien auf. Sie können hier ganz einfach noch weitere Dateien einfügen.

Die alten Dateien der Vortage werden durchnummeriert:

Die "smsd.log" von gestern heisst "smsd.log.1". Die Datei von Vorgestern heisst "smsd.log.2" und so weiter.

Für Interessierte erkläre ich nun, wie dieses Script funktioniert:

Zu Beginn wird die Funktion rename() deklariert, damit man im Hauptprogramm diese mehrmals benutzen kann:

```
rename()
{
...
}

rename /var/log/smsd.log 7
rename /usr/local/apache/logs/access_log 2
rename /usr/local/apache/logs/error_log 7
```

Die Funktion rename soll die genannte Protokolldatei löschen und von den alten Versionen so-und-so viele Tage aufbewahren.

Der Dateiname heisst innerhalb der Funktion $1, weil er das erste Argument des Funktionsaufrufes ist. Die Anzahl der Tage ist entsprechend $2.

Die Funktion löscht zuerst die älteste Protokolldatei, da diese jetzt verfallen soll:

```
rm $1.$2 >/dev/null 2>&1
```

Mittels Ausgabeumleitung werden Fehlermeldungen unterdrückt. Diese treten auf, wenn die zu löschende Datei gar nicht existiert. Wir wollen in diesem Fall keine Fehlermeldung sehen.

```
num=$2
 while [ $num -gt 1 ]; do
   before=`expr $num - 1`
   mv $1.$before $1.$num >/dev/null 2>&1
   num=$before
 done
```

Nun sollen die bereits vorhandenen alten Protokolldateien umbenannt werden, indem die Ziffer am Ende des Dateinamens um eins höher gesetzt wird.

Die Variable "num" beginnt mit der höchsten Nummer und wird bei jedem Schleifendurchlauf um 1 verringert, bis sie nicht mehr grösser als 1 ist. Mittels expr Befehl und Ausgabeumleitung wird die Variable "before" auf die Nummer davor gesetzt, also eins weniger.

Innerhalb der Schleife ist $num also die neue Nummer und $before die alte Nummer. Der mv Befehl benennt die alte Datei in den neuen Namen um. Dabei wird der Dateiname aus $1 und den Nummern kombiniert. Hier wird eine eventuelle Fehlermeldung wieder wie beim rm Befehl unterdrückt.

Nach der Schleife wird die gerade aktuelle Protokolldatei in irgendwas.1 umkopiert und dann wird Sie durch "nichts" überschrieben,also quasi geleert:

```
cp $1 $1.1
cp /dev/null $1
```

Wenn 7 Dateien aufbewahrt werden, erledigt das Script insgesamt folgende Aufgaben:

- Irgendwas.7 wird gelöscht
- dateiname.6 wird in dateiname.7 umbenannt
- dateiname.5 wird in dateiname.6 umbenannt
- dateiname.4 wird in dateiname.5 umbenannt
- dateiname.3 wird in dateiname.4 umbenannt
- dateiname.2 wird in dateiname.3 umbenannt
- dateiname.1 wird in dateiname.2 umbenannt
- dateiname wird nach dateiname.1 umkopiert
- dateiname wird durch "nichts" überschrieben.

6.1.2. Alte SMS Dateien löschen

Die Protokolldateien bereiten Ihnen jetzt keine Sorgen mehr. Je nach Bedarf können Sie unterschiedlich viele Tage aufbewahren, aber Ihre Festplatte wird nun nicht mehr unbegrenzt voll.

Nun sollten Sie sich um die alten SMS Dateien kümmern. Je nach Konfiguration fallen hier mehr oder weniger Dateien an. Schauen Sie in Ihre /etc/smsd.conf:

```
sent=/var/spool/smsd/sent
failed=/var/spool/smsd/failed
received=/var/spool/smsd/received
```

Falls Sie eine oder mehrere dieser Zeilen in Ihrer Konfigurationsdatei haben, werden Ihre gesammelten SMS Dateien im Laufe der Zeit immer mehr. Alte Dateien müssen daher regelmässig gelöscht werden.

Richten Sie sich den Befehl

find /var/spool/sms -ctime +100 -exec rm {} \;

als Cronjob ein. Unter Unix ist der Eintrag in der Cron Tabelle:

```
13 0 * * * find /var/spool/sms -ctime +100 -exec rm {} \;
```

Unter Windows lautet er (alles in einer einzigen Zeile)::

```
13 0 * * * c:\cygwin\bin\bash -c "find /var/spool/sms -ctime +100 -exec rm {} \;"
```

Der Find Befehl findet unter /var/spool/sms alle Dateien, die älter als 100 Tage sind. Für jede Datei wird der rm Befehl aufgerufen, der sie löscht.

6.1.3. Alte SQL Daten löschen

Wenn Sie das Script mysmsd benutzen, werden alle gesendeten und empfangenen Kurzmitteilungen in einer Tabelle protokolliert.

Diese Daten sollten sie regelmässig löschen, z.B. durch den SQL Befehl:

use smsd;
delete from sms_log where sent < now()- interval 100 day;

Dieser Befehl löscht alle Tabelleneinträge, die älter als 100 Tage sind. In dem where-Ausdruck werden zwei MySQL Funktionen angewendet, die ich Ihnen in diesem Buch noch nicht vorgestellt habe. Sie finden deren Erklärung in der MySQL Dokumentation Kapitel 6.3.4.

Die Funktion now() liefert das aktuelle Datum mit Uhrzeit. Die Funktion "interval 100 day" liefert einen Zahlenwert für 100 Tage. Diesen kann man von einem Datum subtrahieren und erhält so das Datum von vor 100 Tagen.

Wenn Sie diesen Befehl in ein Shell Script verpacken, könnte das Script so aussehen (delete_old_sql.sh):

```
#!/bin/sh
SQL_HOST=localhost
SQL_USER=root
SQL_PAssWORD=""
SQL_DATABASE=smsd
SQL_TABLE=sms_log

if [ "$SQL_PAssWORD" != "" ]; then
  SQL_ARGS="-p$SQL_PAssWORD";
else
  SQL_ARGS="";
fi
SQL_ARGS="-h $SQL_HOST -u $SQL_USER $SQL_ARGS -D $SQL_DATABASE -s -e"

mysql $SQL_ARGS "delete from sms_log where sent < now()- interval 100 day;"
```

Als Cron-Job installiert, löscht es regelmässig alle alten SQL Daten:

```
14 0 * * * /usr/local/bin/delete_old_sql.sh >/dev/null 2>&1;
```

Unter Windows muss der Cron-Job so aussehen (alles in eine Zeile schreiben):

```
14 0 * * * c:\cygwin\bin\bash -c "/usr/local/bin/delete_old_sql.sh >/dev/null 2>&1"
```

6.1.4. Selbst-Test

Um absolut sicherzustellen, dass die SMS Server Tools SM richtig versenden und empfangen, könnten Sie ein Script schreiben, dass einen Selbsttest durchführt. Schicken Sie sich selbst eine SM und kontrollieren Sie, ob diese ankommt.

Die folgenden Zeilen gehen davon aus, dass Sie drei Modems mit den Namen GSM1, GSM2 und GSM3 haben. Sie haben sich bereits eine entsprechende Konfigurationsdatei angelegt.

Legen Sie sich dazu das Verzeichnis /var/spool/sms/self-test an. Darunter legen Sie sich für jeden Modem eine hereinkommende und eine herausgehende Warteschlange an:

/var/spool/sms/self-test/GSM1.in

/var/spool/sms/self-test/GSM2.in

/var/spool/sms/self-test/GSM3.in

/var/spool/sms/self-test/GSM1.out

/var/spool/sms/self-test/GSM2.out

/var/spool/sms/self-test/GSM3.out

Tragen Sie die herausgehenden Warteschlangen in die Konfigurationsdatei /etc/smsd.conf ein. Jedes Modem soll diese als letzte Warteschlange bedienen. Die neuen Zeilen sind diese:

```
[queues]
GSM1 = /var/spool/sms/GSM1.out
GSM2 = /var/spool/sms/GSM2.out
GSM3 = /var/spool/sms/GSM3.out

[GSM1]
queues = ..., GSM1

[GSM2]
queues = ..., GSM2

[GSM3]
queues = ..., GSM3
```

Anstelle der Pünktchen (...) lassen Sie bitte die regulären Warteschlangen stehen, die das Modem bereits bedient. Wichtig ist, dass sie als letzte die Test-Warteschlange hinzufügen.

Sie können nun gezielt Nachrichten über jedes einzelne Modem versenden, indem Sie im Vorspann der SMS Dateien den Modemnamen als Provider angeben, z.B.:

```
To: 491721234567
Provider: GSM1

SELF-TEST-MEssAGE
```

Die empfangenen Test-Nachrichten sollen nicht im Verzeichnis /var/spool/sms/incoming landen, sondern in die Modemspezifischen Verzeichnisse verschoben werden. Das erledigt ein Ereignisprogramm, dass Sie bitte als Eventhandler in die Konfigurationsdatei /etc/smsd.con eintragen:

```
eventhandler = /usr/local/bin/self-test-eventhandler.sh
```

Das Scipt hat diesen Inhalt (self-test-eventhandler.sh):

```
#!/bin/sh
if [ "$1" = "RECEIVED" ]; then
   if grep "SELF-TEST-MEssAGE" $2 >/dev/null; then
      modem=`cat $2 | formail -zx Subject:`
      mv $2 /var/spool/sms/self-test/$modem.in
   fi
fi
```

Es verschiebt alle empfangenen Nachrichten mit dem Text "SELF-TEST-MEssAGE" in das entsprechende Testverzeichnis mit der Endung ".in". Test-Nachrichten vom Modem GSM1 landen also im Verzeichnis GSM1.in und so weiter.

Sie haben sich nun die Möglichkeit geschaffen, gezielt mit jedem einzelnen Modem Testnachrichten zu versenden und Sie empfangen diese in gesonderte Verzeichnisse, anstatt /var/spool/sms/incoming.

Jetzt brauchen Sie noch ein Script, dass regelmässig Testnachrichten sendet und kontrolliert, ob sie zeitgerecht empfangen werden. Das folgende Script erledigt diesen Job für ein einzelnes Modem, sie müssen es also dreimal starten, wenn Sie drei Modems haben. Das Script soll self-test.sh genannt werden:

```
#!/bin/sh
out=/var/spool/sms/outgoing/$1.test
in=/var/spool/sms/self-test/$1.in
while true; do
   echo "To: $2" > $out
   echo "Provider: $1" >> $out
   echo "" >> $out
   echo "SELF-TEST-MEssAGE" >> $out
   sleep 3600
   found=`ls -A $in`
   if [ "$found" ]; then
      rm $in/*
      echo "$1 test ok"
   else
      echo "$1 test failed"
   fi
done
```

Starten Sie das obige Script durch Eingabe des Befehls:

self-test.sh GSM1 491721234567 &

Tauschen Sie die obige Telefonnummer bitte durch die Telefonnummer des Modems GSM1 aus. Das Script sendet eine Testnachricht über das genannte Modem an sich selbst. Dann wartet es eine Stunde lang (3600 Sekunden). Nach Ablauf der Stunde prüft das Script, ob inzwischen eine Testnachricht empfangen wurde.

Wenn ja, werden alle Testnachrichten gelöscht und eine Erfolgsmeldung angezeigt. Wenn nein, wird nur eine Fehlermeldung angezeigt.

Der ganze Vorgang wird mittels Endlosschleife so lange wiederholt, bis Sie das Script mit dem Befehl **pkill self-test.sh** anhalten.

Starten Sie das Script self-test.sh für jedes Modem, indem Sie den Modemnamen und dessen Rufnummer angeben. Sie haben dann eine ständige Selbstkontrolle der SMS Server Tools:

self-test.sh GSM1 491721234001 &
self-test.sh GSM1 491721234002 &
self-test.sh GSM1 491721234003 &

Das & Zeichen am Ende des Befehls bewirkt, dass das Programm im Hintergrund ausgeführt wird. So können Sie alle Modems in einem einzigen Fenster beobachten.

Anstelle des Fehlertextes "$1 test failed" in diesem Script können Sie jeden beliebigen Befehl einsetzen, der Sie über die Störung informiert. Wie wäre es mit einer eMail?:

```
echo "Subject: $1 test failed"  | sendmail myname@mydomain
```

Wenn Sie die Selbst-Tests automatisch beim Hochfahren des Rechners ausführen wollen, legen Sie sich ein Startup-Script mit dem Namen /etc/init.d/self-test an:

```
#!/bin/sh
case "$1" in
  start)
    /usr/local/bin/self-test.sh GSM1 401721234001 &
    /usr/local/bin/self-test.sh GSM1 401721234002 &
    /usr/local/bin/self-test.sh GSM1 401721234003 &
    ;;
  stop)
    pkill self-test.sh
    ;;
esac
```

Vergessen Sie nicht, dass das Script ausführbar sein muss, also

chmod a+x /etc/init.d/self-test
chmod a+x /usr/local/bin/self-test.sh

Erstellen Sie nun eine Verknüpfung dazu:

cd /etc/etc/rc3.d
ln -s /etc/init.d/self-test S83self-test

Wie bereits bei der Installation der SMS Server Tools erklärt, müssen Sie die 3 in rc3.d eventuell durch eine andere Ziffer austauschen. Es handelt sich um das Verzeichnis, in dem sich Verknüpfungen für alle Programme befinden, die beim Hochfahren des Rechner gestartet werden sollen.

Programme, die beim Hochfahren gestartet werden, dürfen keine Bildschirmausgaben machen. Entfernen Sie daher alle Bildschirmausgaben und ersetzen Sie diese durch eMails oder irgendwelche anderen Befehle, die Sie über die Störung informieren (self-test.sh):

```
#!/bin/sh
out=/var/spool/sms/outgoing/$1.test
in=/var/spool/sms/self-test/$1.in
while true; do
   echo "To: $2" > $out
   echo "Provider: $1" >> $out
   echo "" >> $out
   echo "SELF-TEST-MEssAGE" >> $out
   sleep 3600
   found=`ls -A $in`
   if [ "$found" ]; then
      rm $in/*
   else
      echo "Subject: $1 test failed" | sendmail myname@mydomain
   fi
done
```

Geben Sie anstelle von "myname@mydomain" die eMailadresse des Systemadministrators ein - das sind Sie vermutlich selbst.

Sie haben die SMS Server Tools nun um ein Testprogramm erweitert, dass jeden Ausfall nach spätestens einer Stunde erkennt und per eMail anzeigt.

Als Nebenwirkung zeigt das Programm auch an, wenn ihr System zu wenig Modems hat und den Versand von SM nicht mehr zeitgerecht bewältigen kann.

Denn wenn der Versand nicht schnell genug vonstatten geht, sammeln sich immer mehr Nachrichten in den Netzbetreiber-Warteschlangen. Die Warteschlangen für die Testnachrichten haben Sie in der Konfigurationsdatei als letzte angegeben. Das bedeutet, dass Testnachrichten nur gesendet werden, wenn alle anderen Warteschlangen leer sind.

Bei einem voll ausgelastetem System tritt dieser Fall niemals ein, die Testnachrichten werden dann nicht gesendet. Demzufolge alarmiert das Programm self-test.sh, dass Tests fehlschlagen.

Eine Fehlermeldung vom Testprogramm kann also auf mehrere Probleme hindeuten:

- Ein Modem sendet nichts mehr
- Ein Modem empfängt nichts mehr
- Ein Modem ist voll ausgelastet = Kapazitätsengpass

Sollte der dritte Fall häufig eintreten, bleiben Ihnen drei Lösungsmöglichkeiten:

- Sie senden weniger Kurzmitteilungen pro Stunde
- Sie fügen weitere Modems hinzu
- Sie tolerieren längere Verzögerungen und erhöhen die Wartezeit von 3600 Sekunden auf eine längere Zeit

Natürlich können SIe die Wartezeit auch verringern, wenn Sie bereits bei kürzeren Verzögerungen alarmiert werden möchten.

6.2. Statusmonitor verbessern

Den Statusmonitor der SMS Server Tools haben Sie sicher bereits ausprobiert. Sie haben bereits gelernt, dass Sie zu diesem Zweck smsd mit der Option -s starten müssen, z.B.

smsd -s

Die Bildschirmausgabe ist informativ, aber sehr schlicht gehalten und nicht besonders übersichtlich:

```
iiiriisssisss---
iiiriisssisss---
rriiiiissisis---
rriiiiiiiiii---
iiiiiiiiiiir---
```

Ausserdem hatte ich in Kapitel 5.8 geschrieben, dass Sie diesen Statusmonitor nicht gebrauchen können, wenn Sie die SMS Server Tools durch ein Start-Script beim Hochfahren automatisch starten lassen.

Die Verbesserung des Statusmonitors besteht also in diesen Arbeitsschritten:

- Umleitung der Ausgabe in ein Hilfsprogramm
- Umformatierung in eine schönere Darstellung

Die Umsetzung der ersten Aufgabe sollte Ihnen klar sein. Ändern Sie das Start-Script der SMS Server Tools (also /etc/init.d/sms) so ab:

```
...
    /usr/local/bin/smsd -s | /usr/local/bin/statusmonitor.sh &
...
```

Jetzt schreiben Sie das folgende Script ab und speichern es unter dem Namen /usr/local/bin/statusmonitor.sh:

```
#!/bin/sh
show()
{
  case "$2" in
    r) echo "Modem $1 is receiving" ;;
    s) echo "Modem $1 is sending" ;;
    i) echo "Modem $1 is idle" ;;
    b) echo "Modem $1 is blocked" ;;
  esac
}

while read status; do
  for num in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16; do
    show $num `echo $status | cut -c$num`
  done > /var/log/smsd.status
done
```

Der read Befehl liest von seinem Eingabekanal eine einzige Zeile und speichert sie in die Variable $status. Durch die while Schleife wird dies solange wiederholt, bis read einen Fehlercode zurückliefert. Das passiert genau dann, wenn smsd beendet wurde.

In einer for schleife wird jedes einzelne Zeichen der Statuszeile mit der Funktion show angezeigt. Hier wird der cut Befehl benutzt, um nacheinander die Zeichen 1 bis 16 zu extrahieren.

Die show Funktion wandelt den Buchstaben in einen lesbarenText um, aber nur bei den Buchstaben r, w, i und b. Bei anderen Buchstaben unterbleibt jede Ausgabe.

Die Ausgabe der for Schleife wird in die Datei /var/log/smsd.status umgeleitet. Der aktuelle Modemstatus kann also jederzeit in dieser Datei nachgesehen werden.

Wenn Sie drei Modems haben, sieht Dateiinhalt so aus:

```
Modem 1 is sending
Modem 2 is idle
Modem 3 is receiving
```

Nun ist es nicht sonderlich komfortabel,immer wieder neu in die gleiche Datei hereinzuschauen, um fortlaufend den Status sehen zu können. Sie können dies durch einen simplen Einzeiler automatisieren:

while true; do cat /var/log/smsd.status; sleep 1; done

Diese while Schleife läuft solange, bis Sie Strg-C drücken. Sie zeigt immer wieder den Status an und wartet dann eine Sekunde.

6.2.1. Statusmonitor als Webseite

Wenn Sie eine bunte Statusanzeige bevorzugen, dann schreiben Sie sich das statusmontior.sh Script aus dem vorherigen Kapitel so um, dass es eine HTML Datei erzeugt, anstelle einer Textdatei.

Ausserdem sollte diese HTML Datei ein kleines Java Script enhalten, dass für eine regelmässige Aktualisiserung sorgt. Die HTML-Version des Statusmonitors könnte so realisiert werden (statusmonitor2.sh):

```
#!/bin/sh
show()
{
  case "$2" in
    r) echo "<td bgcolor=green>Modem $1<br>receiving</td>" ;;
    s) echo "<td bgcolor=red>Modem $1<br>sending</td>" ;;
    i) echo "<td bgcolor=white>Modem $1<br>idle</td>" ;;
    b) echo "<td bgcolor=lightgray>Modem $1<br>blocked</td>" ;;
  esac
}

while read status; do
  echo "<html><body " \
       "onLoad=\"window.setTimeout('location.reload()',3000)\">" \
       "<table border=1><tr>" \
       > /var/log/smsd.status.html
  for num in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16; do
    show $num `echo $status | cut -c$num`
  done >> /var/log/smsd.status.html
  echo "</tr></table></body></html>" >> /var/log/smsd.status.html
done
```

Wenn Sie dieses Script benutzen, um den Statusmonitor zu verbessern, erhalten Sie eine bunte HTML Tabelle mit jeweils einem Kästchen pro Modem. Die Webseite aktualisisert sich alle 3 Sekunden selbst. Für andere Aktualisiserungszeiten ändern Sie die 3000 entsprechend ab. 5000 bedeutet 5 Sekunden.

Wie gehabt müssen Sie die Datei /etc/init.d/smsd entsprechend ändern:

```
...
    /usr/local/bin/smsd -s | /usr/local/bin/statusmonitor2.sh &
...
```

6.2.2. Alarme als bunte Webseite

Sie können jetzt den aktuellen Status aller Modems bequem im Webbrowser beobachten. Nützlich wäre es jetzt, wenn Sie zusätzlich die letzten Alarme sehen könnten.

Legen Sie sich dazu einen Alarmhandler an, der alle Alarme in eine Alarm-Protokolldatei im HTML Format schreibt (alarmhandler.sh):

```
#!/bin/sh
file=/var/log/smsd.alarm.html
touch $file

mv $file $file.old
echo "<html><body bgcolor=lightgray" \
     "onLoad=\"window.setTimeout('location.reload()',60000)\">" \
     > $file

case "$4" in
  1|2) echo "<font color=red>$2 $3 ($5) $6</font><br>" ;;
  3) echo "<font color=orange>$2 $3 ($5) $6</font><br>" ;;
  4) echo "<font color=yellow>$2 $3 ($5) $6</font><br>" ;;
  *) echo "<font color=black>$2 $3 ($5) $6</font><br>" ;;
esac >> $file

sed -e'2,100p' --silent $file.old >> $file
```

Das obige Script legt zuerst mit dem touch Befehl eine Logdatei an, falls keine existiert. Dann wird die Datei im .old umbenannt.

Danach legt es eine neue Datei an und schreibt mit dem echo Befehl einen Header hinein. Der Header enthält ein Javascript, dass für eine minütliche Aktualisierung der Anzeige im Webbrowser sorgt.

Dann schreibt das Script mit der case Anweisung einen Farb-Befehl in die Datei, gefolgt von dem Alarmtext.

Danach kopiert das Script die ersten 99 Zeilen der alten Datei in die neue. Die allererste Zeile der alten Datei wird dabei ausgelassen, denn sie enthält den alten HTML Header, den wir nicht doppelt und dreifach in der neuen HTML haben wollen.

Die neue Alarm-Datei enthält also immer nur die aktuelle Alarm-Meldung, sowie die letzten 99 alten Meldungen. Die Meldungen werden in umgekehrt chronologischer Reihenfolge angezeigt, sie sehen also die neueste Meldung immer ganz oben.

Wenn Sie sich die resultierende HTML Datei genau ansehen, werden Sie bemerken, dass am Ende "</body></html>" fehlt. Das ist Absicht, denn sonst müsste man von der alten Datei auch noch die letzte Zeile abschneiden, was ziemlich kompliziert wäre. Alle mir bekannten Webbrowser zeigen HTML Dateien auch trotz dieses Fehlers korrekt an.

6.3. Mehr als 32 Modems

Wie sie an der Ausgabe des Statusmonitors vielleicht bemerkt haben, unterstützen die SMS Server Tools maximal 32 Modems gleichzeitig.

Sollten Sie Bedarf für mehr Modems haben, müssen Sie die Quelltexte der SMS Server Tools modifizieren und das Programm erneut compilieren und installieren.

Praktischerweise brauchen Sie dazu nur eine einzige Datei zu ändern, und zwar die Datei src/smsd_cfg.h:

```
#define PROVIDER 16
#define DEVICES 32
#define NUMS 16
```

Hier finden Sie die Begrenzungen für die Anzahl der Netzanbieter (und Warteschlangen), die Anzahl der Modems und die Anzahl der Ortsvorwahlen pro Netzanbieter.

Ändern Sie die z.B. Anzahl der Devices auf 50, dann können Sie bis zu 50 Modems anschliessen.

Der PROVIDER Wert bestimmt die Maximale Anzahl von Netzanbietern und Warteschlangen!

6.4. Eine SMS an mehrere Empfänger

Viele Anwender fragen nach, ob man eine SM an mehrere Empfänger zugleich senden kann. Dies ist nicht möglich, weil das Übertragungsprotokoll von Kurzmitteilungen nur einen Empfänger pro Nachricht zulässt.

Realisierbar ist allerdings, dass Sie nur eine SMS Datei anlegen, die von den SMS Server Tools an mehrere Empfänger gesendet wird. Sie brauchen dazu einen Checkhandler, der die betroffenen SMS Datei in mehrere einzelne Dateien zerlegt.

Eine SMS Datei mit mehreren Empfängern sieht so aus:

```
From: Stefan
To: 491720000001
To: 491720000002
To: 491720000003

Hallo, Test!
```

Wenn Sie diese Datei ganz normal senden würden, würde smsd nur die letzte aufgeführte Rufnummer verwenden und die Nachricht dorthin senden.

Das folgende Script installieren Sie als Checkhandler. Es erzeugt für jede Zielnummer eine eigene SMS Datei. Das Senden der Originaldatei mit mehreren Zielen verbietet es, indem es immer den Exitcode 1 zurückliefert. Nennen Sie das Script split_multi_sms.sh:

```
#!/bin/sh
if [ `cat $1 | formail -zx To: | wc -l` -gt 1 ]; then
    cat $1 | formail -zx To: | \
    while read num; do
      file=`mktemp /var/spool/sms/outgoing/send_XXXXXX`
      cat $1 | formail -f -I "To: $num" > $file
    done
    exit 1
else
  exit 0
fi
```

In diesem Script wird der formail Befehl dreimal benutzt. Die ersten beiden Aufrufe liefern eine Liste mit den Empfängern. Der wc Befehl zählt die Anzahl der Zeilen (also die Anzahl der Empfänger). Nur wenn mehr als eins gezählt wurden, wird die Datei gesplittet.

Beim zweiten mal wird diese Liste der Empfänger mittels Ausgabeumleitung (durch das Zeichen "|") an eine while Schleife übergeben. Die Schleife für jeden Empfänger wiederholt und liest dessen Rufnummer mit dem read Befehl in die Variable $num ein. Dann erzeugt sie eine neue SMS Datei mit mktemp. Es folgt ein Formail Aufruf, der die Originaldatei in die neue Zieldatei schreibt, und dabei gleichzeitig dem Empfänger durch $num ersetzt. Die neue SMS Datei enthält danach also nur noch einen einzigen Empfänger.

Bitte beachten Sie, dass diese Methode nicht mit binären Dateien funktioniert. Formail wurde für Texte entwickelt und kann binäre Daten nicht korrekt wiedergeben. Sollten Sie die gleiche Funktionalität auch für Binäre SMS Dateien brauchen, müssen Sie sich ein "echtes" C Programm schreiben lassen, welches diese Aufgabe erledigt. Im Rahmen dieses Buches werde ich nicht weiter darauf eingehen.

7. Praktische Anwendungsbeispiele

In diesem Kapitel stelle ich Ihnen einige praktische Anwendungsbeispiele vor. Betrachten Sie das Kapitel als Anregung für eigene Entwicklungen.

7.1. Zentrales Alarmsystem

Das Zentrale Alarmsystem sammelt Alarme von mehreren Quellen an einem zentralen Punkt.

Alle Einbruch-Meldeanlagen können auf diese Weise drahtlos und über beliebige Entfernungen mit dem Wachdienst verbunden werden. Maschinen können Störungen an den betreibenden Techniker senden. Die Meldungen mehrere Alarmquellen werden an einem Computer gesammelt und auf dem Bildschirm angezeigt.

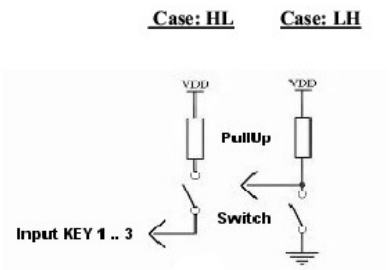
Auf der Empfängerseite benötigen Sie einen Computer mit den SMS Server Tools sowie einem beliebigen GSM Modem. Dieser empfängt die Alarme aller Sender und zeigt sie an.

Auf der Senderseite benötigen Sie ein oder mehrere Falcom A3D Modems. Diese Modems haben eine besondere Firmware, die speziell für solche Alarm Anwendungen entwickelt wurde.

7.1.1. Senderseite

Sie können pro Modem bis zu drei Alarmeingänge belegen. Diese befinden sich an der 15 poligen Sub-D Buchse. Sobald Sie an einen Eingang eine Spannung von 4,5V-12V über einen 47k Ohm Widerstand legen, sendet das Modem eine Alarm-SM. Bitte beachten Sie die Anschlusshinweise im Handbuch des Modems.

Pin 1: Eingang 1
Pin 2: Eingang 2
Pin 3: Eingang 3
Pin 9: Ausgang 8-36V vom Netzteil
Pin 15: Gnd



Die Alarmeingänge funktionieren nur, wenn das Modem extern mit Strom versorgt wird. Der interne Akku versorgt diesen Teil des Gerätes nicht mit Notstrom.

Konfigurieren Sie das Modem zunächst so, dass es die PIN nach dem Einschalten automatisch eingibt, denn es soll später selbstständig ohne Computer als Alarm-Sender eingesetzt werden:

AT+CPIN=1234
&CNF PIN=1

Stellen Sie nun die SMSC Nummer ein:

AT+CSCA="+49172270000"
AT+CSAS

Mit dem nächsten Befehl legen Sie die Zielrufnummer für die Alarm-Nachrichten fest, sowie die gewünschten Textinhalte:

&CNF KEY1,s01721234567,"Alarm Signal 1",H
&CNF KEY2,s01721234567,"Alarm Signal 2",H
&CNF KEY3,s01721234567,"Alarm Signal 3",H

H bedeutet, dass ein Alarm gesendet wird, wenn der Eingang auf High wechselt. L würde einen Alarm beim Wechsel auf Low senden und E sendet einen Alarm in beiden Fällen.

Von nun an können Sie das Modem vom Computer trennen und als eigenständigen Alarm-Sender benutzen. Bei Aktivierung einer der drei Eingänge wird der gerade eingestellte Text "Alarm Signal x" gesendet.

Bei Bedarf können Sie das Falcom Modem zusätzlich anweisen, regelmässig eine "ich lebe noch" Kurzmitteilung zu senden. So können Sie an der zentralen Empfangsstelle sehen, dass das Modem nicht funktioniert und dass die Stromzufuhr nicht unterbrochen ist. Geben Sie dazu ein:

&CNF MSG,s01721234567,"Test",60

Die 60 bedeutet, dass dieser Text alle 60 Minuten gesendet wird. Um die Testnachricht abzuschalten stellen Sie 0 Minuten ein.

7.1.2. Empfängerseite

Auf der Empfängerseite installieren Sie die SMS Server Tools, wie weiter vorne im Buch beschrieben. Nun brauchen Sie nur noch ein Ereignisprogramm, dass alle Alarme anzeigt.

Das folgende Ereignisprogramm erzeugt eine HTML Datei, in der man immer die letzten 1000 Alarme sehen kann. Sie werden in umgekehrt chronologischer Reihenfolge angezeigt. Die aktuellsten Meldungen erscheinen immer oben. Die Liste aktualisiert sich jede Minute von selbst.

Speichern Sie das folgende Script als /usr/local/bin/alarmevent.sh ab:

```
#!/bin/sh
if [ "$1" != "RECEIVED" ]; then
  exit
fi

file=/var/log/smsd.alarm.html
touch $file
mv $file $file.old
echo "<html><body bgcolor=lightgray" \
     "onLoad=\"window.setTimeout('location.reload()',60000)\">" \
     > $file

SENT=`formail -zx Sent: < $2`
TEXT=`formail -I "" < $2`
echo "<b>$SENT</b> $TEXT<br>" >> $file

sed -e'2,100p' --silent $file.old >> $file
```

Dann richten Sie es als Ereignisprogramm in der Konfigurationsdatei smsd.conf ein. Die dazu nötige Zeile lautet:

```
eventhandler=/usr/local/bin/alarmevent.sh
```

Vergessen Sie nicht, das Script als ausführbar zu markieren, indem Sie

chmod a+x /usr/local/bin/alarmevent.sh

eingeben.

Das war schon alles. Sie können jetzt an zentraler Stelle Alarme empfangen und im Webbrowser anzeigen. Wenn Sie auf dem Server zusätzlich den Apache Webserver installieren oder das Verzeichnis mit der HTML Datei im lokalen Netzwerk freigeben, können auch andere PC diese Alarm-Liste anzeigen.

7.2. Fahrzeugortung

Sie können Fahrzeuge mit Hilfe der SMS Server Tools überwachen. Ein Falcom GSM Modemmit GPS Zubehör sendet regelmässig die geographische Position des Fahrzeuges an Ihren Computer. Mit Hilfe einer Landkarte können Sie dann jederzeit den Standort des Fahrzeuges ausmachen.

Eine MySQL Datenbank wird verwendet, um die gemeldeten Positionen des Fahrzeuges aufzuzeichnen. Sie können die Datenbank mit dem SQL Client oder einer PHP Webseite auslesen.

7.2.1. Senderseite

Verwenden Sie ein Falcom A3D Modem mit GPS und GSM Antenne. Verbinden Sie die Ignition Leitung des Modems mit der gleichnamigen Leitung des Zündschlosses, damit sich das Modem entsprechend aus und ein schaltet.

Konfigurieren Sie das Modem zunächst so, dass es die PIN nach dem Einschalten automatisch eingibt, denn es soll später selbstständig ohne Computer als Alarm-Sender eingesetzt werden:

AT+CPIN=1234
&CNF PIN=1

Stellen Sie nun die SMSC Nummer ein:

AT+CSCA="+49172270000"
AT+CSAS

Die nächsten Befehle legen fest, dass das Modem beim Einschalten und danach alle 5 Minuten eine Kurzmitteilung senden soll:

&CNF BOOT,s01721234567,"Auto Ein",HP
&CNF MSG,s01721234567,"Auto Timer",5

Wenn Sie mehrere Fahrzeuge überwachen möchten, vergeben Sie für jedes Auto einen eindeutigen Text oder verwenden Sie die Rufnummer zur Unterscheidung.

Sie können das Modem so konfigurieren, dass es nach Abschalten der Zündung nicht sofort aus geht. Der folgende Befehl stellt das Modem so ein, dass es sich erst 20 Minuten verzögert abschaltet:

&CNF IGN,20

Sie können Werte im Bereich 0-60 einsetzen. 0 bedeutet, dass das Modem sofort aus geht.

Das Modem sendet nun beim Einschalten die Position des Fahrzeuges, sowie danach alle 5 Minuten.

7.2.2. Empfängerseite

Auf der Empfängerseite installieren Sie die SMS Server Tools, wie weiter vorne im Buch beschrieben. Nun brauchen Sie nur noch ein Ereignisprogramm, dass die empfangenen Nachrichten in eine SQL Datenbank einträgt.

Speichern Sie das folgende Script als /usr/local/bin/gpsevent.sh ab:

```
#!/bin/sh
if [ "$1" != "RECEIVED" ]; then
  exit 1
fi
SENT=`formail -zx Sent: < $2`
TEXT=`formail -I "" < $2`

SQL_ARGS="-h localhost -u root -ppassword -D smsd -s -e"
mysql $SQL_ARGS "insert into gps (sent,text) values (\"$SENT\",\"$TEXT\");"
```

Passen Sie die Parameter in der Zeile SQL_ARGS=... entsprechend Ihrer Datenbank an.

Dann richten Sie es als Ereignisprogramm in der Konfigurationsdatei smsd.conf ein. Die dazu nötige Zeile lautet:

```
eventhandler=/usr/local/bin/gpsevent.sh
```

Vergessen Sie nicht, das Script als ausführbar zu markieren, indem Sie

chmod a+x /usr/local/bin/alarmevent.sh

eingeben.

Legen Sie sich eine Datenbank für die GPS Positionsdaten an:

```
stefan@server> mysql -u root -p
mysql> create database smsd;
Query Ok...
mysql> use smsd;
Database changed.
mysql> create table gps (
    -> id int auto_increment not null,
    -> primary key(id),
    -> sent datetime,
    -> text char(160)
    -> );
Query Ok...
```

Das war schon alles. Sie können jetzt an zentraler Stelle die Positionen aller Fahrzeuge empfangen. Zum Anzeigen der Daten geben Beispielsweise den folgenden SQL Befehl im MySQL Client ein:

```
select * from gps where text like "%Auto%";
```

Es erscheint eine Liste aller gemeldeter Positionen des Fahrzeuge mit dem Namen "Auto". Um die Ausgabe auf einen bestimmten Zeitraum zu beschränken, geben Sie ein:

```
select * from gps where text like "%Auto%"
    and sent>="2003-02-22" and sent<="2003-02-24";
```

Dies zeigt alle Positionen vom 22.2.2003 ab 0:00 bis zum 24.2.2003 0:00 an.

Das Modem unterstützt zahlreiche unterschiedliche Nachrichtenformate. Wenn Sie die Standardvorgabe behalten, empfangen Sie Kurzmitteilungen im GGA Format. Beispiel:

```
$GPGGA,161229.487,3723.2475,N,12158.3416,W,1,07,1.0,9.0,M,,,,0000*18
```

Die folgenden Zahlen sind dabei am interessantesten:

161229487	16:12:29 Uhr UTC
3723.2475 N	Breitengrad 37 Grad 23 Minuten 24 Sekunden Nord
12158.3416 W	Längengrad 121 Grad 58 Minuten 34 Sekunden West
7	7 Satelliten wurden empfangen
9.0 M	Höhe 9 Meter über Normal Null

7.3. eMail zu SMS Gateway

Sie können mit einem eMail Client SMS senden, wenn Sie Ihren Mailserver mit den SMS Server Tools verbinden. Zwei unterschiedliche Lösungsmöglichkeiten möchte ich Ihnen in diesem Kapitel vorstellen:

- eMail zu SMS Gateway mit lokalem Mailserver
- eMail zu SMS Gateway mit entferntem Mailserver

Die erste Variante geht nur unter Unix, denn nur Unix Mailserver unterstützen die in diesem Buch vorgestellte Methode.

Die zweite Variante können Sie auch unter Windows anwenden.

Nach Installation eines solchen Gateways können Sie SMS mit jedem eMail Programm senden. Als Empfänger geben Sie eine bestimmte feste Adresse an. In die Betreff-Zeile schreiben Sie die Handynummer des Empfängers.

Sie müssen das eMail Programm so einstellen, dass es den Nachrichtentext im Ascii Format sendet und nicht im HTML Format!

Eine geeignete eMail sieht z.B. so aus:

```
From: "Frank Mustermann" <mustermann@irgendwo.de>
To: "+491721234567" <sms@localhost>
Subject: not used

Hallo, das ist eine Test-SMS!
```

7.3.1. eMail zu SMS Gateway mit lokalem Mailserver

Sie benötigen dazu einen Server mit den SMS Server Tools und der Mail-Server Software Sendmail. Sie können die beiden Programme auch auf unterschiedlichen Rechnern laufen lassen, aber dann benötigen Sie ein Netzlaufwerk, auf das beide Rechner gemeinsam zugreifen.

Übergabepunkt zwischen Sendmail und den SMS Server Tools ist das Verzeichnis für die herausgehende Warteschlange, also normalwerweise /var/spool/sms/outgoing.

Legen Sie unter Unix einen User mit dem Namen sms an. Stellen Sie sicher, dass Sendmail Nachrichten an diesen User korrekt ausliefert.

Wenn Sendmail nicht korrekt funktioniert, schauen Sie bitte in das Kapitel 4.5.

Prüfen Sie Sendmail, indem Sie diesen Befehl eingeben:

/usr/lib/sendmail sms
Hallo!
<Strg-D>

Loggen Sie sich jetzt als User sms ein und prüfen Sie den Posteingang, oder schauen Sie in die Datei /var/spool/mail/sms. Sie sollten die eMail mit dem Text "Hallo!" sehen können.

Sendmail stellt lokale eMails mit dem Programm procmail zu. Jeder User kann Procmail dazu anweisen, empfangene eMails woanders zu speichern, als in der normalen eMail Datei. Genau diese Funktion nutzen wir jetzt aus.

Loggen Sie sich als User sms ein. Legen Sie dann die Datei .procmailrc im Home-Verzeichnis an. Sie soll diesen Inhalt haben:

```
VERBOSE=off
MAILDIR=/var/spool/mail
DEFAULT=/var/spool/mail/sms
LOGFILE=/var/log/procmail

:0
* ^TOsms
| /usr/local/bin/email2sms
```

Wenn Sie jetzt eine eMail an den SMS User senden, wird sie nicht in dessen Posteingang gespeichert, sondern an das Script email2sms gesendet. Dieses Script wandelt die eMail in eine SMS Datei um, die dann umgehend von den SMS Server Tools gesendet wird.

Geben Sie in das To: Feld der eMail dessen Handynummer ein, sowie die eMail Adresse des Users sms. Beispiel:

To: „Herbert +491721234567“ <sms@localhost“

Das Script email2sms ist Bestandteil der SMS Server Tools. Sie können es sich gerne ansehen um davon zu lernen. Das Script ist klein und einfach gehalten.

7.3.2. eMail zu SMS Gateway mit entferntem Mailserver

Sie benötigen dazu einen Server mit den SMS Server Tools, Fetchmail und Formail. Weiterhin brauchen Sie einen Mailserver und darauf ein POP3 Postfach.

Diese Variante können Sie auch unter Windows benutzen, denn sie funktioniert ohne Sendmail.

Bevor Sie jetzt vergebens nach den Installationsdateien von Formail suchen, nehmen Sie bitte zur Kenntnis, dass Formail ein Hilfsprogramm von Procmail ist und daher in dem Programmpaket Procmail versteckt ist.

In Kapitel 4.5.4 habe ich Ihnen gezeigt, wie Sie eMail von einem POP3 Postfach abrufen.

Legen Sie sich eine /etc/fetchmailrc Datei an, die so aussieht (alle in einer einzigen Zeile):

```
poll mail.isis.de protocol POP3 user sms pass passwort mda "/usr/local/bin/email2sms"
```

Setzen Sie bitte den Namen Ihres Mailserver ein, sowie den richtigen Usernamen und das richtige Passwort. Setzen Sie die Zugriffsrechte auf die Datei so:

chmod 0710 /ect/fetchmailrc

Starten Sie den Fetchmail Dämon durch Eingabe von:

fetchmail -f /etc/fetchmailrc -d 300

Fetchmail ruft nun alle 300 Sekunden alle eMail für den konfigurierten aus dem POP3 Postfach ab und sendet sie an das Script email2sms. Dieses Script wandelt die eMails in SMS Dateien um und speichert sie in das Verzeichnis für die herausgehende Warteschlange. Die SMS Server Tools senden die Nachrichten dann umgehend.

Wie Sie sehen, ist diese Lösung ebenso einfach, wie die aus dem vorherigen Kapitel. Der Nachteil ist, dass Fetchmail die Mails in regelmässigen Zeitabständen abrufen muss, was zu Verzögerungen im Versand führt.

Der Vorteil ist, dass Sie jeden beliebigen bereits existierenden Mailserver benutzen können, ohne dort irgendetwas zu konfigurieren.

Das Script email2sms ist Bestandteil der SMS Server Tools. Sie können es sich gerne ansehen um davon zu lernen. Das Script ist klein und einfach gehalten.

7.4. Absender mit SQL prüfen

Wenn Sie eine eMail zu SMS Gateway oder ein Online-Formular für SMS öffentlich anbieten, z.B. im Internet, wollen sicher nur bestimmten Personen das Senden von SMS erlauben.

Sie legen sich dazu eine SQL Tabelle an, in der alle berechtigten Absender mit Name und Passwort aufgelistet sind. Dann setzen Sie einen Checkhandler ein, der bei jeder SMS Datei prüft, ob der Absender zum Senden von SM berechtigt ist.

Legen Sie die Tabelle durch Eingabe der folgenden Befehle im MySql Client an:

```
stefan@server> mysql -u root -p
mysql> create database smsd;
Query Ok...
mysql> use smsd;
Database changed.
mysql> create table smsuser (
    -> id int auto_increment not null,
    -> primary key(id),
    -> name char(40),
    -> password char(16)
   -> );
Query Ok...
```

Füllen Sie die Tabelle mit einigen Usernamen und Passwörtern:

mysql> **insert into smsuser set name="Michaela Meier", password="abxdf";**
mysql> **insert into smsuser set name="Richard Wagner", password="blabla";**

Und so weiter.

Der Checkhandler soll nun so geschrieben werden, dass er auch mit eMail Clients benutzt werden kann (siehe letztes Kapitel). Den Usernamen kann man aus der Absenderadresse der eMail entnehmen, für das Passwort benutzen wir einfach die Betrifft-Zeile.

Absender:	"Michaela Meier" <michaela.meier@mail.isis.de>
Empfänger:	"+491721234567" <sms@mailserver.de>
Betrifft:	passwort
Text:	Hallo Welt!

Das Script email2sms, dass sie im vorherigen Kapitel kennen gelernt haben, macht daraus eine SMS datei, die so aussieht:

```
From: Michaela Meier
To: +491721234567
Subject: passwort

Hallo Welt!
```

Schreiben Sie sich nun das Script /usr/local/bin/checkhandler.sh mit diesem Inhalt:

```
#!/bin/sh
name=`formail -zx From: < $1`
to=`formail -zx To: < $1`
password=`formail -zx Subject: < $1`

SQL_ARGS="-h localhost -u root -ppasswort -D smsd -s -N -B -e"
result=`mysql $SQL_ARGS select name from smsuser where name=\"$name\" and
password=\"$password\";`

if [ -z "$result" ]; then
  exit 1
fi
```

Das Script überprüft den Absendernamen und das Passwort. Wenn ein gleichlautender Eintrag in der Datenbank steht, beendet sich das Script ohne Fehlercode. Wurde aber kein Eintrag gefunden, beendet sich das Script mit Fehlercode 1.

Der Name und das Passwort werden aus den From: und Subject: Zeilen mit Formail extrahiert.

Die SMS Server Tools senden die Kurzmitteilung nur dann, wenn der Checkhandler ohne Fehlercode endet.

Richten Sie sich dieses Script als Checkhandler in der Konfigurationsdatei /etc/smsd.conf ein:

```
checkhandler=/usr/local/bin/checkhandler.sh
```

7.5. Eisprung Erinnerung

Dieses Beispiel ist nicht wirklich meine eigene Idee. Bei der Suche nach interessanten Webseiten ist meine Frau auf diese Erfindung gestossen. Da ich sie interessant finde, möchte ich Ihnen zeigen, wie Sie einen solchen Dienst mit den SMS Server Tools realisieren können.

Aufgabe des Dienstes "Eisprung Erinnerung" ist es, interessierte Frauen kurz vor dem Eisprung und kurz vor der Blutung per SMS an den Termin zu erinnern.

In einem Anmeldeformular tragen die Kundinnen ein, wann die letzte Blutung begann und wie lange eine Periode bei ihr dauert.

Aus diesen Angaben berechnet Ihr Server das Datum des nächsten Eisprunges und der nächsten Blutung. Daraus ergeben sie die Tage, an denen die Kundinnen Erinnerungs-SM erhalten werden.

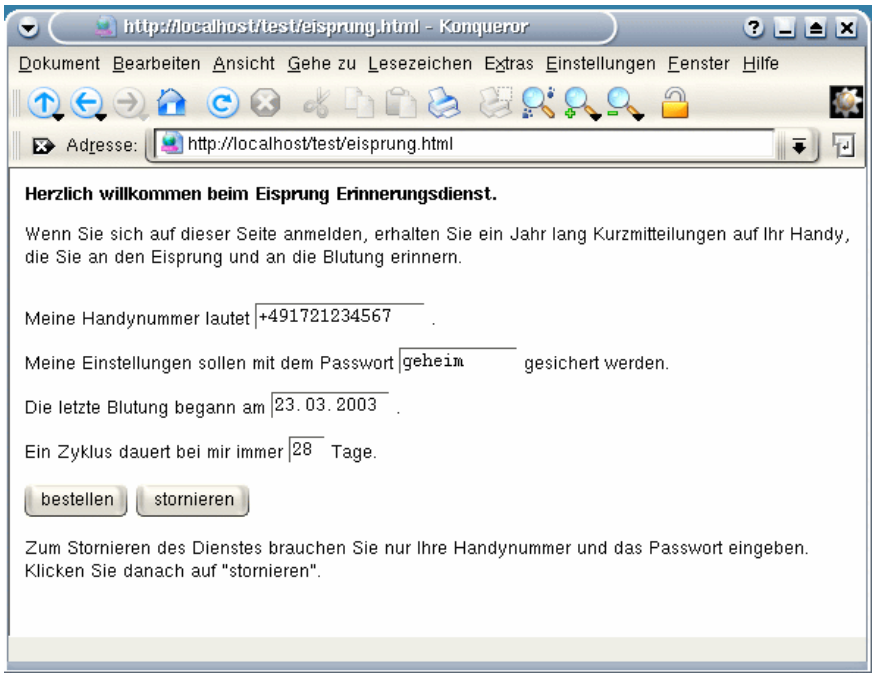
Ein Zyklus dauert immer ungefähr 28 Tage, die genaue Dauer ist von Frau zu Frau unterschiedlich. Der Eisprung findet etwa 13 Tage nach Beginn der Blutung statt.

7.5.1. Das Anmeldeformular

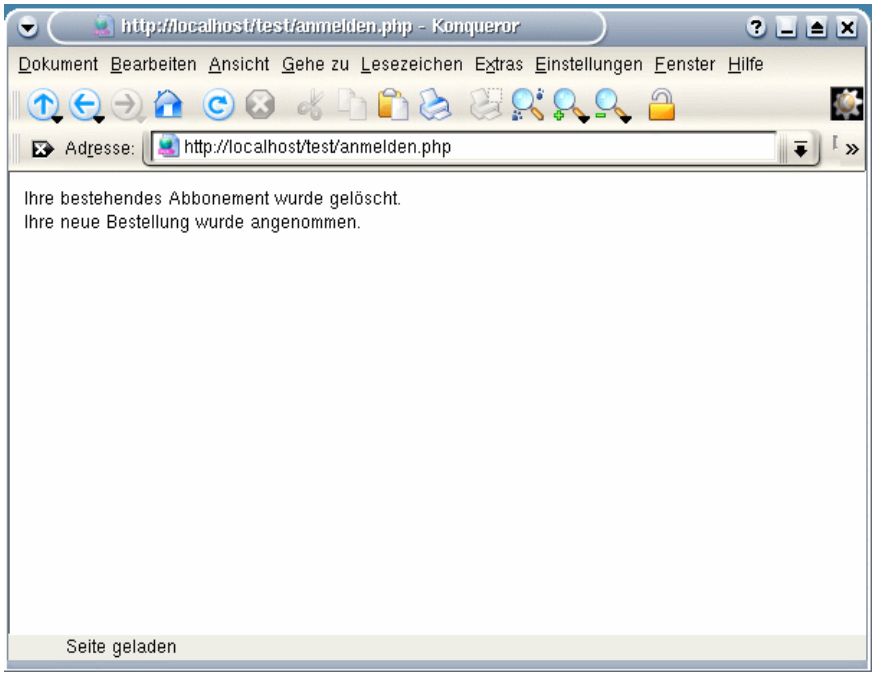
Die Anmeldeseite lautet eisprung.html und sieht so aus:

```
<html><body>
<b>Herzlich willkommen beim Eisprung Erinnerungsdienst.</b>
<p>
Wenn Sie sich auf dieser Seite anmelden, erhalten Sie ein Jahr lang Kurzmitteilungen auf
Ihr Handy, die Sie an den Eisprung und an die Blutung erinnern.
<p>
<form action="anmelden.php" method="post">
  Meine Handynummer lautet
  <input type="text" name="handynummer"
    size=15 maxlength=15 value="+491721234567">  .<p>
  Meine Einstellungen sollen mit dem Passwort
  <input type="text" name="password"
    size=10 maxlength=15 value="geheim">
  gesichert werden.<p>
  Die letzte Blutung begann am
  <input type="text" name="blutung"
    size=10 maxlength=10 value="23.3.2003">.<p>
  Ein Zyklus dauert bei mir immer
  <input type="text" name="zyklus"
    size=2 maxlength=2 value="28"> Tage.<p>
  <input type="submit" name="aktion" value="bestellen">
  <input type="submit" name="aktion" value="stornieren">
</form>
<p>
Zum Stornieren des Dienstes brauchen Sie nur Ihre Handynummer und das Passwort eingeben.
Klicken Sie danach auf "stornieren".
</body></html>
```

So sieht diese Webseite im Webbrowser aus:



Nach einer Bestellung erscheint eine einfache Erfolgsmeldung:



7.5.2. Das Anmelde-Script

Wenn die Kundin nach Eingabe ihrer Daten auf [bestellen] oder [stornieren] klickt, wird das Script anmelden.php aufgerufen:

```
<html><body>
<?php

function eingabefehler($meldung)
{
  print("$meldung <br>");
  print("Bitte korrigieren Sie Ihre Eingabe.");
  exit;
}

function sqlfehler()
{
  print("Es ist ein Fehler in der Datenbank aufgetreten: <br>");
  print(mysql_error());
  exit;
}

$handynummer=$_POST["handynummer"];
$passwort=$_POST["passwort"];
$blutung=$_POST["blutung"];
$zyklus=$_POST["zyklus"];
$aktion=$_POST["aktion"];

if (($handynummer=="") || ($passwort=="") || ($blutung=="") || ($zyklus==""))
  eingabefehler("Sie müssen alle Felder ausfüllen.");

if (($handynummer=="+491721234567") || ($passwort=="geheim"))
  eingabefehler("Sie müssen Ihre eigene Handynummer und Passwort eingeben.");

if (($zyklus<20) || ($zyklus>36))
  eingabefehler("Die Dauer des Zyklus' kann nicht stimmen.");

$datum=explode(".",$blutung);
if (checkdate($datum[1],$datum[0],$datum[2])==false)
  eingabefehler("Das Datum ist ungültig");

$jetzt=time();
$zeitstempel=mktime(0,0,0,$datum[1],$datum[0],$datum[2]);
if ($zeitstempel>$jetzt)
  eingabefehler("Das Datum muss in der Vergangenheit liegen.");

mysql_connect("localhost","root","passwort");

if ($result=mysql_db_query("smsd","select passwort from eisprung ".
   "where handynummer=\"$handynummer\" limit 1;"))
{
  $reihe=mysql_fetch_row($result);
  $altes_passwort=$reihe[0];
}
else
  sqlfehler();

if ($altes_passwort!="")
{
  if ($altes_passwort!=$passwort)
    eingabefehler("Sie haben ein falsches Passwort eingegeben!");
  else
  {
    if (mysql_db_query("smsd","delete from eisprung ".
       "where handynummer=\"$handynummer\";"))
    {
      print("Ihre bestehendes Abbonement wurde gelöscht.<br>");
    }
    else
      sqlfehler();
  }
}

if ($aktion=="bestellen")
{
  $anzahl=0;
  while ($anzahl<24)
  {
    $zeitstempel=$zeitstempel+13*60*60*24;
    if ($zeitstempel>$jetzt)
    {
      $neudatum=strftime("%Y-%m-%d",$zeitstempel);
      if (mysql_db_query("smsd","insert into eisprung (handynummer,passwort,datum,typ) ".
         "values (\"$handynummer\",\"$passwort\",\"$neudatum\",\"E\");")==false)
        sqlfehler();
      $anzahl++;
    }
    $zeitstempel=$zeitstempel+($zyklus-13)*60*60*24;
    if ($zeitstempel>$jetzt)
    {
```

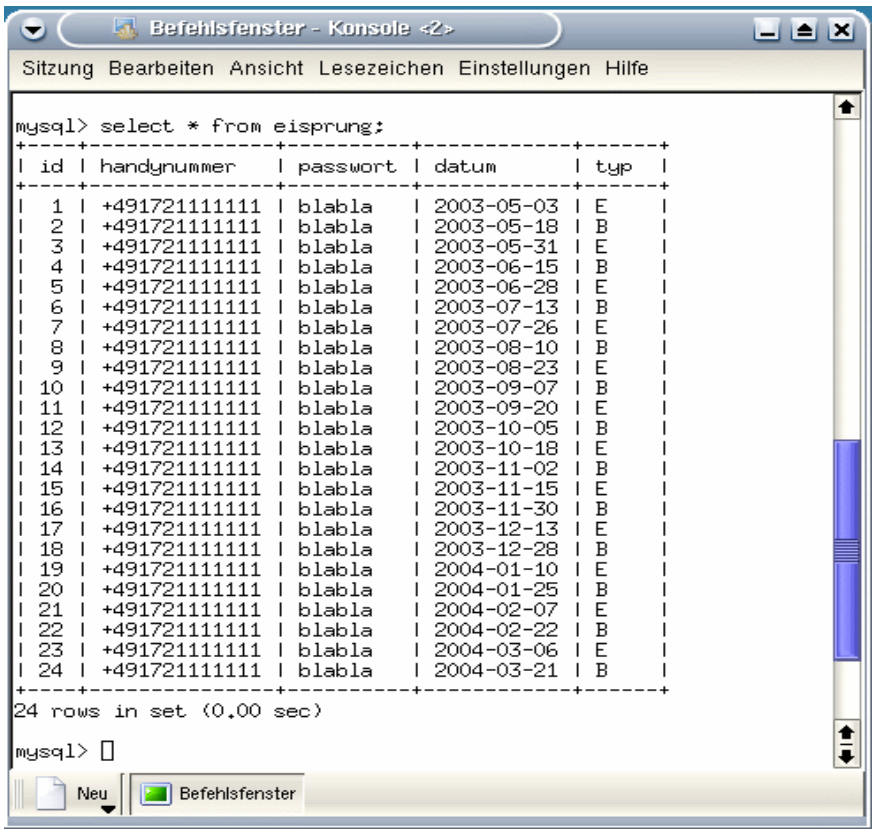
```
        $neudatum=strftime("%Y-%m-%d",$zeitstempel);
        if (mysql_db_query("smsd","insert into eisprung (handynummer,passwort,datum,typ) ".
           "values (\"$handynummer\",\"$passwort\",\"$neudatum\",\"B\");")==false)
          sqlfehler();
        $anzahl++;
      }
    }
    print("Ihre neue Bestellung wurde angenommen.");
}

?>
</body></html>
```

Das Script beginnt mit einigen Plausibilitätskontrollen der eingegebenen Daten. Dann löscht es die alten Bestellungen und trägt ggf. neue Bestellungen in die SQL Tabelle ein.

Die Argumente zum mysql_connect Befehl müssen Sie gemäss Ihrem Datenbankserver anpassen.

7.5.3. Die Tabelle mit Terminen

Das Script erzeugt einen Zeitplan in Form einer SQL Tabelle, in der alle Tage stehen, an denen die Kundin eine SM bekommen soll. Die Tabelle sieht so aus:



Diese Tabelle müssen Sie natürlicht erstmal anlegen, sonst kann das Script nicht funktionieren:

```
stefan@server> mysql -u root -p
mysql> create database smsd;
Query Ok...
mysql> use smsd;
Database changed.
mysql> create table eisprung (
    -> id int auto_increment not null,
    -> primary key(id),
    -> handynummer char(15),
    -> passwort char(15),
   -> datum date,
    -> typ char(1)
    -> );
Query Ok...
```

7.5.4. Der Cronjob

Ihre Kundinnen können den SMS Dienst jetzt abbonieren und stornieren. Was jetzt noch fehlt, ist ein Cronjob, der täglich die angemeldeten Termine abarbeitet und Kurzmitteilungen sendet.

Schreiben Sie sich dazu das Script /usr/local/bin/eisprung.sh:

```
#!/bin/sh
liste=/tmp/eisprung.liste
heute=`date +%Y-%m-%d`

SQL_ARGS="-h localhost -u root -ppassword -D smsd -N -B -e"
mysql $SQL_ARGS "select typ,handynummer from eisprung where datum=\"$heute\";" > $liste

while read zeile; do
  typ=`echo $zeile | cut -f1 -d' '`
  nummer=`echo $zeile | cut -f2 -d' '`
  case $typ in
    E) sendsms $nummer "Sie haben einen Eisprung!" ;;
    B) sendsms $nummer "Ihre Blutung wird heute beginnen!" ;;
  esac
  echo $typ $nummer
done < $liste
mysql $SQL_ARGS "delete from eisprung where datum<=\"$heute\";"
```

Die SQL Parameter in SQL_ARGS müssen Sie an Ihren SQL Server anpassen.

Als Cronjob eingerichtet, sendet es täglich alle SMS gemäss der Eisprung-Tabelle und löscht alte Einträge in dieser Tabelle.

Der Cronjob könnte so aussehen:

```
30 6 * * * /usr/local/bin/eisprung.sh
```

Bzw. unter Windows:

```
30 6 * * * c:\cygwin\bin\bash -c "/usr/local/bin/eisprung.sh"
```

7.6. Klingeltöne, Logos und Sprüche

Einer der beliebtesten SMS Dienste sind zweifellos die Klingeltöne und Logos. Daher möchte ich Ihnen zeigen, wie sie einen solchen Dienst realisieren können.

Zunächst brauchen Sie Klingeltöne, Logos, Sprüche oder welche Daten Sie auch immer anbieten möchten. Im Internet werden zahlreiche Programme angeboten, mit denen Sie Klingeltöne und Logos erstellen können.

Die meisten dieser Programm laufen nur unter Windows. Sie können die Dateien in der Regel nicht im Binärformat speichern. Um an die "Rohdaten" heranzukommen, benötigen Sie einen Computer mit den SMS Server Tools und einen Windows PC, auf dem Sie die Dateien erstellen.

Erstellen Sie nun z.B. ein Logo und senden Sie es als SM mit dem Windows PC an den anderen Computer mit den SMS Server Tools. Dieser speichert die empfangene Nachrichten im Verzeichnis /var/spool/sms/incoming ab. Nun haben Sie eine SMS Datei im richtigen Format. Diese können Sie an jede beliebige andere Rufnummer senden.

Bei Betreiberlogos sollten Sie wissen, dass der Name (genauer dessen Kennzahl) im Logo enthalten ist. Wenn Sie ein Logo für D2 erstellen, und an einen D1 Kunden senden, wird dieser das Logo niemals sehen. Das Handy würde dieses Logo nur anzeigen, wenn der Kunde zu D2 wechseln würde.

Um die empfangenen Binärdatei an ein anderes Ziel senden zu können, müssen Sie eine "To:" Zeile einfügen. Dazu können Sie das Programm Formail leider nicht gebrauchen, denn es verändert die Binärdaten. Formail funktioniert nur mit Textnachrichten korrekt.

Fügen Sie die "To:" Zeile daher mit dem echo und cat Befehl ein, z.B. so:

echo "To: 491721234567" > neu.sms
cat original.sms >>neu.sms

Die Originaldatei darf freilich keine "To:" Zeile haben, sonst enthält Ihre Zieldatei zwei solche Zeilen. Smsd erkennt immer nur die letzte Empfängernummer, was dann die falsche wäre.

7.6.1. Content bereitstellen

Sie wissen nun, wie Sie Betreiberlogos korrekt speichern, so dass Sie sie an andere Empfänger senden können. Sie wissen, wie Sie die Empfängernummer in die SMS Datei einfügen können.

Nun legen Sie sich ein Verzeichnis an, in dem Sie alle Dateien, die sie anbieten möchten, zusammenstellen. Verwenden Sie der Einfachheit halber Nummern als Dateiname.

Achten Sie bei Betreiberlogos darauf, dass der Netzbetreibername Teil des Dateinamens ist, denn jeder Betreiber benötigt eine eigene SMS Datei.

Sie könnten die Daten z.B. in /var/sms/content speichern. Beispiel:

/var/sms/content/0001	Klingelton 1
/var/sms/content/0002	Klingelton 2
/var/sms/content/0003	Klingelton 3
/var/sms/content/1001D1	Betreiber Logo 1 für D1
/var/sms/content/1001D2	Betreiber Logo 1 für D2
/var/sms/content/1001EPLUS	Betreiber Logo 1 für EPLUS
/var/sms/content/1002D1	Betreiber Logo 2 für D1
/var/sms/content/1002D2	Betreiber Logo 2 für D2
/var/sms/content/1002EPLUS	Betreiber Logo 2 für EPLUS
/var/sms/content/1003D1	Betreiber Logo 3 für D1
/var/sms/content/1003D2	Betreiber Logo 3 für D2
/var/sms/content/1003EPLUS	Betreiber Logo 3 für EPLUS
/var/sms/content/2001	Schlauer Spruch 1
/var/sms/content/2002	Schlauer Spruch 2
/var/sms/content/2003	Schlauer Spruch 3

Alle diese Dateien sind SMS Dateien mit einem Vorspann ohne "To:" Zeile!

Wenn Ihre Kunden nun eine Datei bestellen, sollen Sie Ihnen einfach eine SM mit dem dem Dateinamen schicken. In einem Katalog oder auf einer Webseite bieten Sie die Dateien entsprechend an.

Um mit dem Dienst Geld zu verdienen, lassen Sie sich von allen Netzbetreibern jeweils eine fünfstellige SMS Nummer geben. Ihre Kunden bezahlen für das Senden von SM an diese Nummern erhöhte Gebühren, von denen die Netzbetreiber einen lukrativen Anteil an Sie abtreten.

7.6.2. Das Script

Sie benötigen auf Ihrem Server mit den SMS Server Tools nicht nur die gerade genannten SMS Dateien, sondern natürlich auch ein Script, dass die Bestellanforderungen bearbeitet.

Richten Sie sich ein SQL Protokoll gemäss Kapitel 5.13 ein, um Abrechnungen und Reklamationen prüfen zu können. Aktivieren Sie Statusreports (report=yes).

Das Script installieren Sie als Ereignisprogramm (eventhandler) ein. Nennen Sie es order.sh:

```
#!/bin/sh
if [ $1 != RECEIVED ]; then
 exit
fi
from=`formail -zx From: < $2`
text=`formail -I "" < $2`
text=`echo $text`
if [ -f "/var/sms/content/$text" ]; then
 file=`mktemp /var/spool/sms/outgoing/send_XXXXXX`
 echo "To: $from" > $file
 cat "/var/sms/content/$text" >>$file
else
 file=`mktemp /var/spool/sms/outgoing/send_XXXXXX`
 echo "To: $from" > $file
 echo "" >> $file
 echo "Sorry, falsche Bestellnummer ($text) !" >> $file
fi
```

Die Zeile

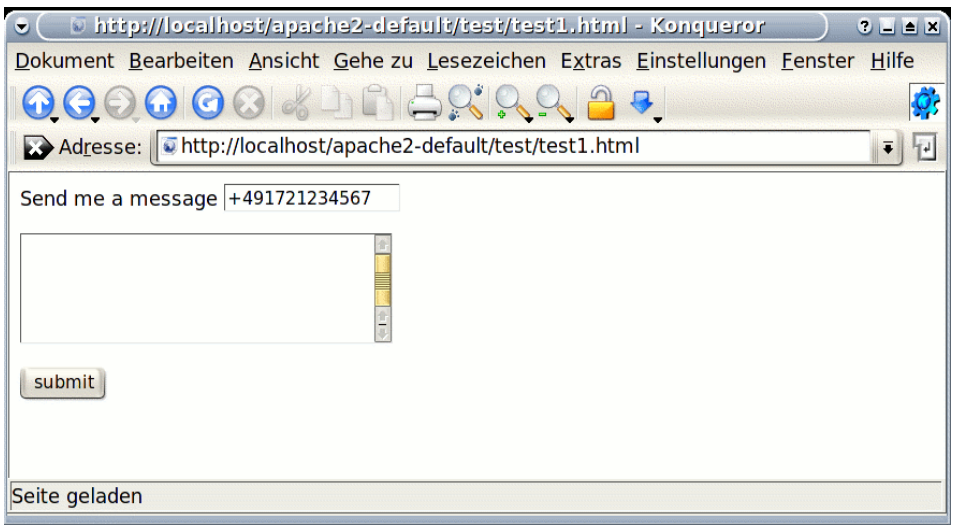
text=`echo $text`

kommt Ihnen vielleicht sinnlos vor. Sie hat aber sehr wohl einen Sinn. Auf diese Weise entfernt das Script Leerzeichen um die Bestellnummer herum. Anschliessen kann in der if Abfrage geprüft werden, ob eine gleichnamige Datei existiert.

Wenn ein Kunde Beispielsweise die Datei 1001D2 bestellen möchte, muss er Ihrem SMS Server eine SMS mit dem Text "1001D2" schicken. Bei falschen Bestellnummern bekommt er eine entsprechende Fehlermeldung als Antwort.

7.7. Web-Formular sendet Nachrichten

Das Senden einer Nachricht aus einem Web-Formular ist ganz einfach. Prinzipiell brauchen Sie nur die Nachricht als Datei im outgoing Verzeichnis speichern.



7.7.1. Das Web-Formular

Das ist der Quelltext der Webseite test1.html:

```
<html><body>
<form action="send.php" method="post">
  Sende mir eine Nachricht
  <input type="text" name="destination"
    size=15 maxlength=15 value="+491721234567">
  <p>
  <textarea name="message" cols="30" rows="4"></textarea>
  <p>
  <input type="submit" name="action" value="submit">
</form>
</body></html>
```

7.7.2. Das sende-Script

Das ist der Quelltext des Sende-Scriptes send.php:

```
<?php
$filename = tempnam("","");
$file=fopen($filename,"w");

fwrite($file, "To: ". $_POST["destination"] ."\n");
fwrite($file, "\n");
fwrite($file, $_POST["message"] ."\n");

fclose($file);
copy($filename,"/var/spool/sms/outgoing/". basename($filename));
unlink($filename);
?>

<html><body>
  Ihre Nachricht wird in wenigen Sekunden gesendet.
</body></html>
```

Zuerst erzeugt das Script eine temporäre Datei. Die Funktion tempnam ist Betriebssystem spezifisch und erzeugt eine Datei (nicht nur einen Dateinamen) mit zufälligem Namen in dem Verzeichnis, wo normalerweise temporäre Dateien hin gehören.

Dann schreibt das Script die Telefonnummer, eine Leerzeile und den Nachrichtentext in die temporäre Datei.

Dann kopiert es die temporäre Datei in das outgoing Verzeichnis des SMS Servers und löscht die temporäre Datei.

Sie könnten die Datei auch direkt im outgoing Verzeichnis anlegen, aber dann müssen Sie selbst sicherstellen, daß der Dateiname einmalig ist. Ich bevorzuge dafür die Funktionen des Betriebssystems.

Wenn Der Webserver keinen direkten Zugriff auf das ougoing Verzeichnis hat, kann er die Datei auch per ftp hochladen:

```
copy($filename,"ftp://servername/directory/". basename($filename));
```

Bitte schauen Sie in das Handbuch von PHP, um herauszufinden, ob Ihre Version diese Funktion unterstützt und wie man sie aktiviert.

- ENDE -