IMAP Emailserver unter Linux

15. Juli 2002

Teil I: Der kompakte Weg zum Emailserver

Version 1.0.9.de

basierend auf: SuSE Linux 7.x / 8.x

Um Verbesserungen bzw. Änderungen dieser Dokumentation verfolgen zu können, bitte ich jeden Leser darum, diese an die Email-Adresse: **`roland@linux-tin.org`** zu senden. Über diese Email-Adresse kann das Dokument auch in ASCII und LaTeX angefordert werden. Die jeweils aktuelle Version des Dokuments kann man unter der im Anhang A angegebenen Webadresse beziehen.

Verwendete Software für die Dokumenterstellung:

- LyX unter SuSE Linux 8.0
- GNU Ghostscript 6.5.3
- Open Office 1.0
- Und natürlich eine Reihe anderer Applikationen

Inhaltsverzeichnis

Der folgende Artikel erklärt, wie man unter SuSE Linux einen Emailserver basierend auf Postfix, Cyrus IMAP und Fetchmail erfolgreich einrichtet und administriert

1 Grundsätzliches

1.1 Vorwort

Eigentlich hasse ich ja überlange Vorworte aber selbst kann ich mich nie zurückhalten (man möge mir verzeihen). Vorweg schicken muss ich jedoch, dass ein gut funktionierender Email-Server durchaus **kein** ganz triviales Thema ist. Als ich vor einiger Zeit begann den eigenen Mailserver zu "konstruieren", wusste ich noch nicht, dass dies mein aufwendigstes Projekt werden würde. Mir schwebte vor, die volle Email - Funktionalität eines Servers ala Microsoft Exchange bzw. Lotus Notes nachzubilden (wie gesagt nicht die komplette Groupware Funktionalität nur den Email Part). Das Ganze jedoch mit standardisierten Methoden und Protokollen. Letztendlich landet man dabei bei **IMAP** und **OpenLDAP** als Adressbuch (Erklärungen zu diesen Begriffen kommen später).

Die Informationssammlung stellte mich auf eine harte Probe. Beginnend beim MTA[1] Sendmail stellte ich schnell fest, dass die "per-Hand" Konfiguration und Administration viel zu aufwendig war. Über kurz oder lang landete ich (neben einigen Ausflügen zu QMail) bei Postfix. Keine Frage, für meinen Kenntnisstand und Ziel gab es keine bessere Lösung. Dies jedoch ist reine Geschmacksache. Alle MTAs sind standardisiert und spielen hervorragend mit den anderen Komponenten zusammen, hier in dieser Anleitung wird jedoch immer nur Bezug auf Postfix genommen.

Die restlichen Teile des Mailservers, wie Cyrus IMAP und Fetchmail gesellten sich erst nach einer Weile hinzu, die ich dir aber hier nicht vorenthalten möchte und werde.

Nachdem ich jetzt jeden, der sich noch nicht mit diesem Thema beschäftigt hat wohl mehr verwirrt habe, als Klarheit geschaffen, erkläre ich zunächst die Begriffe, die hier immer wieder auftauchen werden. (Keine Panik, alles ist gar nicht mal so kompliziert, wie es sich vielleicht anhören mag :-)

1.2 Begriffserklärungen

MTA: **M**ail **T**ransfer **A**gent. Das Bindeglied und zentraler Punkt des Mailservers. Der MTA übergibt Emails an andere MTAs oder liefert sie an lokale MDAs aus und nimmt Emails von wieder anderen MTAs entgegen bzw. erhält sie über weitere Zwischensysteme (z.B. Fetchmail). Postfix und Sendmail sind bekannte Vertreter dieser Gattung. Hört sich kompliziert an, ich weiss, ist es aber nicht. Siehe dazu die Abbildung in 1.3.1 unten. Stelle dir das Ganze vor, wie den normalen Briefverkehr, dann ist der MTA das **Postamt**, das den Briefträgern und den Verwalltungsangestellten vorgibt, was zu tun ist.

MDA: **M**ail **D**elivery **A**gent: Übernimmt die Emails vom MTA und verteilt diese an den entsprechenden User - schiebt diese in Queues oder zu Programmen, die diese Mails weiterverarbeiten. Procmail oder Deliver sind MDAs. Zurück zum Briefverkehr: Der MDA ist der **Postbote**, der anhand der Anschrift und Postleitzahl weiss, wohin der Brief zugstellt werden soll - in ein Postfach, oder an eine bestimmte Adresse.

[1] MTA: Mail Transfer Agent - z.B Sendmail, QMail, Postfix - Informationen dazu bei Punkt 1.2

MUA: **M**ail **U**ser **A**gent. In der elektronischen Welt ist dies der Email - Client, bzw. der serverseitige MUA namens imapd oder popd. In der realen Welt kann man dies mit dem Briefeschreiber/-empfänger oder dem **Postkasten** vergleichen.

POP3: **P**ost **O**ffice **P**rotocol. Die dritte Version dieses Protokolls erlaubt einem Client Computer Email abzuholen und auf dem Rechner zu speichern und zu verwalten. Eine Verwaltung auf dem Server ist äußerst eingeschränkt und eigentlich nicht vorgesehen. Ein Vergleich mit der "realen" Welt kommt das Protokoll dem einfachen **Postschliessfach** nahe. Jede neue Post wird einfach eins nach dem anderen hineingeworfen und kann je nach Wunsch abgeholt werden.

IMAP4: **I**nternet **M**essage **A**ccess **P**rotocol. "Unser" Protokoll, da wir damit die Verwaltung der Mails serverseitig erledigen können. Ebenso ein Backup wird erleichtert, wie auch das problemlose Wechseln des Clients. Es gibt drei Möglichkeiten der Mailverwaltung: Online, offline oder verbindungsorientiert. Dieses Protokoll mit der Realität zu vergleichen ist schon ein bisschen schwerer. Ich vergleiche es dennoch gerne mit einer(m) **Sekretär(in)**, welche(r) die Post vor- und in Ordner soriert und dann vorlegt und später dann weiterverwaltet.

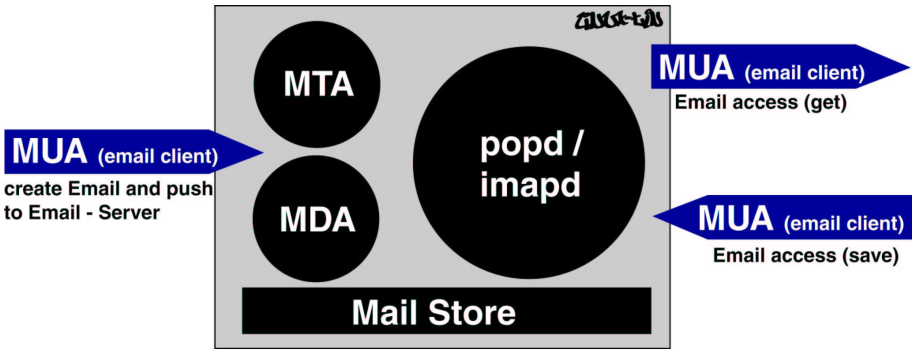
SMTP: **S**imple **M**ail **T**ransport **P**rotocol: Ein Server zu Server oder Client zu Server Protokoll - so tauschen MTAs Nachrichten aus bzw. so senden Clients ihre Mails direkt an einen zentralen MTA.

1.3 Grafische Darstellung

Hier möchte ich die in 1.1 und 1.2 vorgestellte Thematik noch einmal in eine grafische Form bringen. Die Bilder sollen nur eine Übersicht bieten:

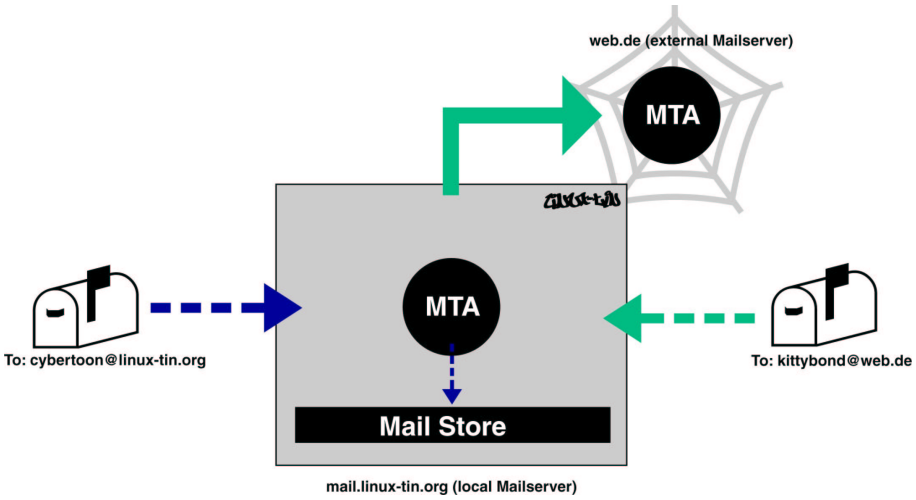
1.3.1 Emailserver - Typische Teile eines Emailservers im Ganzen

In der folgenden Grafik sieht man sehr gut, aus welchen Teilen ein Mailserver besteht. Für alle Teile gibt es eine Unmenge an verschiedenen (OpenSource) Produkten, dennoch harmonieren alle Teile auf Grund der offenen Standards hervorragend miteinander.



1.3.2 Was macht der MTA

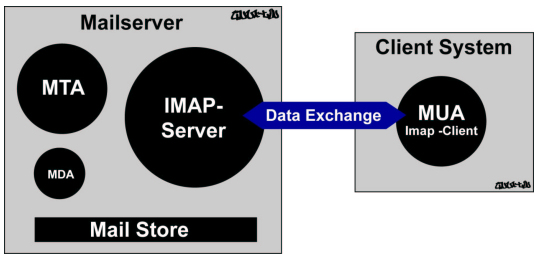
Das untenstehende Bild zeigt die Funktionsweise eines MTA. Ich stelle hier sowohl eine externe (`kittybond@web.de`), wie auch eine interne (`cybertoon@space.mil`) Email vor, bzw. deren Wege. Die interne Adresse wird durch den MTA als solche erkannt und in den MailStore (lokale Emailablage) weitergegeben (z.B. `/var/spool/mail`). Die Emails an die externe Adresse kann nicht in den lokalen Speicher geschrieben werden, deswegen wird diese (evtl. über einen Zwischenhost) an den externen MTA übergeben. In den Begriffen eines Postsystems, wird einmal der Brief an das Postamt ausgeliefert, das dem Empfänger am nächsten ist und dann von dort zugestellt. Das andere Mal ist der Empfänger im selben Postleitzahlengebiet wie das Postamt - der Brief kann direkt zugestellt werden ohne ein weiteres Postamt zu bemühen.



1.3.3 Was macht ein MUA

Im nächsten Bild sieht man die zwei möglichen User Agents. Ein IMAP Client tauscht (im Online-Modus) während der Bearbeitung ständig Daten aus. Dabei wird im Normalfall oder Idealfall pro Client ein Serverprozeß gestartet. Dennoch ist der IMAP - Daemon nicht das belastende Moment des Servers, sondern der MDA - dazu später jedoch mehr.

Im realen Leben kann man sich den Datenaustausch als ein Gespräch zwischen Verwaltungsangestellten vorstellen, die aushandeln, wohin z.B. die einzelnen Briefe sortiert werden sollen.



1.4 Lange Einleitung und kein Ende ...

Das war mal wieder eine lange Einleitung, aber um “unseren” Mailserver erfolgreich implementieren zu können sind obige Grundbegriffe essentiell. Im nächsten Abschnitt geht es um die Vorbereitungen. Das hier vorgstellte System ist sehr genügsam, dennoch darf man nicht unterschätzen, was ein IMAP-Mailserver alles leisten muss.

2 Vorbereitungen

2.1 Analyse Phase 1: Server Hardware

Gehört dieser Punkt eigentlich in diese Anleitung? Fast schon wollte ich diesen Punkt ab Version 1.0.0 streichen, aber da kaum jemand glaubt, wie genügsam ein gut konfigurierter IMAP-Mailserver sein kann im Vergleich zu vielen kommerziellen System, habe ich diesen Punkt nochmal überarbeitet.

Leider ist jedoch eine π mal Daumen - Rechnung oder eine allgemeine Aussage nicht möglich. Alles hängt immer vom Mailverhalten der User und dem zur Verfügung stehenden Budget ab. Es gibt Firmenkulturen, die das interne Mailsystem auch als Filesharing-Tool benutzen. große Datenmengen müssen übertragen und gespeichert werden - der Server hat also jede Menge zu tun.

Da das hier vorgestellte Mailsystem keine direkte Verbindung zum Internet hat (sprich: keine feste IP-Adresse), sondern der Partner eines weiteren Systems ist (Stichwort: Workgroup Mailserver), beschränken wir uns auf eine Useranzahl von maximal 50. Mehr macht bei dieser Art des Email-Austausches keinen Sinn mehr. Der Hauptserver kann übrigens alles sein, z.B. ein “grosser” Firmenserver mit direkter SMTP - Verbindung oder auch der POP3 Server eines Providers.

Ich habe mir angewöhnt pro User 200MB Festplattenplatz zu veranschlagen. Die IMAP - Mailbox zusammen mit den LOG-Files usw. liegen zwar oftmals darunter, aber etwas Luft zum Atmen sollte unser System noch haben. Zudem setzte ich 4 MB Hauptspeicher pro User an. Daten werden in den Puffer geschrieben und der IMAP Server arbeitet effizienter.

Nehmen wir also an, unser Server wird maximal mit 25 Usern “belastet”. Für diese sollte dann ein Plattenplatz von $25 \times 200MB = 5000MB$ bereitgestellt werden. Aus Erfahrungsgründen ist auf Grund von Wartbarkeit und bei einem Datengau eine Verdopplung des Plattenplatzes durchaus zu empfehlen und $2 \times 25 \times 200MB = 10000MB \approx 10GB$ bereitzustellen. Diese stellen den optimalen Ausbau dar; bei heutigen Plattenpreisen klingt das immer noch lächerlich. Zurück zum Hauptspeicher - $25 \times 4MB = 100MB$ für den Mailserver sollte es schon sein - mehr ist dann nur noch Luxus :-).

Da unser Linux-Betriebssystem auch noch gerne ein wenig Hauptspeicher und Plattenplatzen für sich hätte, geben wir diesem “luxuriöse” 128 MB und 2 GB Fesplattenplatz.

Fassen wir zusammen: ca. 12GB Festplattenplatz und 256MB RAM für 25 User macht schon Freude. Die Frage der CPU erübrigt sich in diesen modernen Zeiten. Die “billigste” aktuelle Prozessorgeneration ist schon mehr als genug. Selbst ein Uraltsystem der Pentium I Reihe (iA32) und 100MHz wäre noch ausreichend, obwohl ich bei hochfrequentierten Systemen bei einigen hundert offenen IMAP Threads schon Wartezeiten auf einem Dual Xeon System erlebt habe. Dies ist jedoch zu 99% ein Problem der Clientsoftware im Zusammenspiel mit schlechter Administration - sprich: keine

gesetzten Quotas bei denen dutzende von Benutzern meherere Gigabyte große Mailarchive händelten.

Nicht zu verschweigen sei, dass auch andere Architekturen gut geeignet sind (von uns getestet auf PPC604/200MHz unter SuSE 7.1 *PPC*)

Zurück zum Thema: obige Rechnung ist also nur ein Anhaltspunkt und die Minimalausstattung - bei entsprechendem Budget ist selbstredend ein redundantes System zu bevorzugen.

Wie kommt man zu den Zahlen? Eine Umfrage unter großen Betreibern von Cyrus-IMAP Servern zeigt, dass ein IMAP System mit etwa 1 bis 2 MB physischem Speicher pro aktiven IMAP-Prozess ausgestattet sein muss. Pro User werden im Schnitt 2 Prozesse gestartet. Dadurch kommt man auf die Zahl 4 MB. Es ist also mkehr oder minder eine Methode des Testens angesagt.

2.2 Analyse Phase 2: Server Software

Das Betriebssystem GNU/Linux stellt eine sehr gute Basis für einen erfolgreichen Mailserver dar. Bei Mehrprozessorsystemen sollte man mindestens Kernel 2.4.x heranziehen, welcher spätestens seit 2.4.18 sehr zu empfehlen ist. Ich benutze für diesen Artikel die SuSE Distribution 8.0 für iA32.

Weiterhin benutze ich u.a. die SuSE Pakete:

- postfix
- cyrus-imapd
- openldap2
- openldap2-client
- openssh
- pam-ldap
- fetchmail
- inetd
- iputils

3 Postfix - der MTA

3.1 Postfix im Überblick

Bei Postfix handelt es sich um einen flexiblen, einfach konfigurierbaren, modularen und sicheren MTA.

3.1.1 Das Konzept hinter Postfix

Durch seinen, im Gegensatz zu Sendmail, konsequent modularen Aufbau ist es der Mail Transfer Agent meiner Wahl und kann ihn dir deshalb nur ans Herz legen. Natürlich würde unser IMAP Mailserver auch mit dem Standard Sendmail zusammenarbeiten, aber dieser ist wesentlich schwerer zu konfigurieren und zu verwalten.

Modularer Aufbau bedeutet, dass es sich nicht um ein einzelnes monolithisches Programm handelt sondern aus vielen Teilen besteht. Diese Teile wiederum können unterschiedliche Rechte besitzen und müssen nicht als Superuser root laufen. Welche Teile dies sind, möchte ich dir in folgender Tabelle zeigen:

Modul	Erklärung
`master`	Wie der Kernel von Linux handelt es sich bei master um den Koordinator des Zusammenspiels der übrigen Module
`qmgr`	Queue-Manager, also die Verwaltungseinheit von Postfix (alte Version)
`nqmgr`	New Queue-Manager, demnach also die neue und verbsserte Version von qmgr. Kümmert sich um die Warteschlange und den Email-Versand
`pickup`	Zuständig für die Aufnahme neuer Emails aus der Queue und Weitergabe an cleanup
`postdrop`	Nimmt lokal erzeugte Emails entgegen und speichert diese in der Queue
`smtp`	Zuständig für den Versand an andere Mailserver via smtp
`smtpd`	Der Gegensatz von smtp. Nimmt also Emails entgegen, die von anderen Hosts stammen
`bounce`	Zuständig für unzustellbare Emails
`cleanup`	Sollte eine Mail im Header unvollständig sein, ergänzt dieses Modul fehlende Einträge
`local`	leitet Emails evtl. an lokale Mailboxen weiter
`pipe`	Kann weitere externe Module zwischenschalten
`showq`	Anzeige der Queue und Status der enthaltenen Emails
`trivial-rewrite`	Verändert bzw. fügt evtl. fehlende Einträge in Headern von Nachrichten hinzu.

3.1.2 Möglichkeiten die Postfix bietet

Wie soll ich da jetzt anfangen? Soll ich unter diesem Punkt einen Roman schreiben? Nicht nötig! Postfix bietet genau die Möglichkeiten, die man von einem MTA erwartet. Sogar noch etwas mehr! Eingehende Emails werden entgegengenommen, sofern es sich nicht um unerwünschte Werbung handelt. Ausserdem werden Emails an die entsprechenden Ziele verteilt oder in Relay-Servern zwischengelagert. Zugegeben kann

Postfix schon noch ein bißchen mehr, aber Beschränke dich zunächst auf diese Möglichkeiten.

3.2 Konfiguration des Postfix Servers

Postfix hat an die 100 Konfigurationsparameter die in der Datei `/etc/postfix/main.cf` zusammengefasst sind. Dennoch sind die meisten Parameter mit idealen Defaultwerten vorbelegt. Es reicht also in den meisten Fällen aus, einige wenige Änderungen vorzunehmen.

Besonderes Augenmerk legst du auf das Verzeichnis `/etc/postfix/`. Hier liegen die Konfigurationsdateien mit deren Hilfe du 99% der Einstellungen rund um den MTA vornehmen kannst.

3.2.1 Konfigurationsdatei /etc/postfix/master.cf

Bei der master.cf handelt es sich um jene Konfiguartionsdatei, die den Koordinator „`postmaster`" beschreibt. Hier wird definiert, wie die Module zu agieren haben.

Wenn du dir diese Datei mal genauer ansiehst, siehst du, wie die einzelnen Module aufgerufen werden sollen. Ich zeige dir mal einen Ausschnitt (aus der Standarddatei):

```
# ==========================================================================
# service type private unpriv chroot wakeup maxproc command + args
# ==========================================================================
  smtp    inet n       -      n      -      -       smtpd
  pickup  fifo n       n      n      60     1       pickup
  cleanup unix -       -      n      -      0       cleanup
  qmgr    fifo n       -      n      300    1       qmgr
  cyrus   unix -       n      n      -      -       pipe
    flags=R user=cyrus argv=/cyrus/bin/deliver -e -m ${extension} ${user}
```

Unter normalen Umständen musst du nichts an dieser Datei ändern, aber was sind schon normale Umstände? Unter SuSE 8.0 hat sich hier nämlich ein Fehler eingeschlichen. Der Pfad des flags (letzte Zeile) muss nämlich lauten:

```
flags=R user=cyrus argv=/usr/lib/cyrus/bin/deliver -e -m ${extension} ${user}
```

3.2.2 Konfigurationsverzeichnisse / -dateien des Servers

In fast allen Distributionen finden sich die Konfigurationsdateien von Postfix entweder in `/etc/postfix` oder `/etc`. Im Folgenden werde ich dir die einzelnen Dateien und Pfade vorstellen um dir einen kleinen Überblick zu verschaffen.

/etc/postfix/main.cf: 90% von Postfix werden in dieser Datei konfiguriert. Hier legst du all jene Einstellungen fest, die zur grundsätzlichen Funktion notwendig sind. Läuft dein Mailserver einmal korrekt, wirst du nur noch selten über diese Datei stolpern, da alle Variablen Einträge in anderen Dateien vorgenommen werden.

/etc/postfix/master.cf: Diese Datei hatten wir schon mal :-). Eigentlich sollte diese Datei nur in Ausnahmefällen und Spezialanwendungen verändert werden.

/etc/postfix/access: Konfiguration rund um eingehende Emails. Sie werden hier anhand des Mailheaders auf ihre „Rechtmäßigkeit" eingestellt.

/etc/aliases: Anhand des aliases werden lokale Emails anderen Adressen zugeordnet. Weshalb dies notwendig ist, erkläre ich weiter unten.

/etc/postfix/canonical: Neben Aliases kann es notwendig sein, auch externe Emails „umzubiegen", d.h. neue Adressen zu geben, um eine reibungslose Funktion zu gewährleisten. Auch dazu unten mehr.

/etc/postfix/relocated: Muss eine Adresse zurückgewiesen werden, weil der tatsächliche Empfänger nicht mehr unter dem alten Namen existiert, aber die neue Adresse bekanng ist, wird hier ein Eintrag vorgenommen.

/etc/postfix/transport: Normalerweise ist das Protokoll SMTP der Standard zum versenden von Emails. Muss aber ein anderes Protokoll verwendet werden, z.B. UUCP, kann dies hier konfiguriert werden.

/etc/postfix/virtual: Diese Datei ist vergleichbar mit /etc/aliases. Bietet zusätztlich erweiterte Funktionalität (z.B. externe Weiterleitung).

3.2.3 Konfigurationsdatei /etc/postfix/main.cf

Diese Konfigurationsdatei kannst du auch via Yast automatisiert anlegen lassen - was du jedoch tunlichst unterlassen solltest, da du dann auf etwaige Seiteneffekte keinen Einfluss mehr hast.

Wesentlich mehr Möglichkeiten Postfix zu steuern hast du, wenn du die Datei mit einem Texteditor bearbeitest - ausserdem kannst du nur so die Funktionsweise eines MTAs erlernen.

Unter SuSE bis 7.3 solltest du die automatische Generierung durch YAST wie folgt ausschalten: Bearbeite die Datei `/etc/rc.config.d/postfix.rc.config` und setzte den Eintrag `POSTFIX_CREATECF` auf „`no`".

Ab SuSE 8.0 hat sich das ein wenig geändert: Bearbeite die Datei `/etc/sysconfig/mail` und setze den Eintrag `MAIL_CREATE_CONFIG` auf „`no`".

Im Folgenden erkläre ich dir die wichtigsten Postfix-Variablen in der **Konfigurationsdatei** main.cf. Du erkennst die Einträge anhand der Schriftart (`Typewriter` bzw. `Courier`). Die Erklärungen sind in der Schriftart Times gehalten. Übrigens, anders als in den meisten Erklärungen beginne ich mit einer etwas anderen Reihenfolge...

Los gehts... zunächst solltest du den ISP Mailserver oder Masterserver, der die zu versendenden Mails entgegennimmt, angeben. Statt dem Domänennamen könnte hier z.B. auch die IP-Adresse stehen:

```
relayhost="smtp.linux-tin.org"
```

Willst Du Deinen Mails nur zu bestimmten Zeiten an den Hauptserver weiterleiten, kannst Du die sofortige Weitergabe unterbinden mit (wenn du die zeile wegläßt werden die Emails sofort versandt):

```
defer_transports = smtp
```

Der nächste Abschnitt behandelt die Grundeinstellungen von Postfix ab SuSE 8.0 - die Einstellungen der früheren Versionen sind als Kommentar eingefügt. Besonderes Augenmerk gilt der blau/fett hervorgehobenen Zeile, die speziell für Cyrus IMAP Server eingetragen werden muss. Dabei handelt es sich um den MDA, der die Mails lokal an den IMAP Server verteilt.

```
program_directory = /usr/lib/postfix
command_directory = /usr/sbin
daemon_directory = /usr/lib/postfix
queue_directory = /var/spool/postfix
default_privs = nobody
mail_spool_directory = /var/mail
mailbox_command = /usr/lib/cyrus/bin/deliver
mailbox_transport = cyrus
# SuSE 7.x: mailbox_command = /usr/cyrus/bin/deliver
local_destination_concurrency_limit = 2
default_destination_concurrency_limit = 10
debug_peer_level = 2
```

Es folgt die IP-Range des eigenen Netzwerkes in dem sich die Clients befinden. Die hier eingetragenen IP-Adressen gelten als „trusted" also vertrauenswürdig. Nur den hier angegebenen IP-Adressen ist es erlaubt auch eMails nach draussen zu senden:

```
mynetworks = 192.168.0.0/24, 127.0.0.0/8
```

Darstellung des eigenen Hostsnames des Mailservers (+ Domäne) als nächstes nach dem Schema: mailservername.localdomain.cc. Dieser Eintrag hat zunächst keine direkte Bedeutung. Um eine Absenderadresse zu bilden, wird zieht Postfix den Eintrag aus `$mydomain` heran.

```
myhostname = mail.linux-tin.org
mydomain = linux-tin.org
myorigin = $mydomain
```

Der nächste Eintrag weist dem Benutzer postfix die Rechte der Prozesse rund um Postfix zu. Dies ist wichtig bzw. sicherheitsrelevant, damit ein Angreifer keine Möglichkeiten hat ausserhalb des Postfixprozesses schaden anzurichten. Über das modulare Konzept von Postfix werde ich später noch ein paar Worte verlieren. Halten wir fest, der nächste Eintrag weisst den Prozessen den Benutzer postfix zu:

```
mail_owner = postfix
```

Weiterhin sagst Du Postfix, dass es die Email an den Provider via SMTP schicken soll:

```
default_transport = smtp
```

Jetzt könnte hier eine lange Geschichte über den richtigen Absender bzw. Adressaten stehen. Nur kurz: weder gehört es zur Netiquette, noch ist es besonders sinnvoll NICHT-Offizielle Domänennamen als “From:” - Adressaten zu verwenden. (Besonders dann, wenn du eine Antwort auf deine Emails erwartest). Deswegen musst du dich mit dem Namen deines Providers bzw. des Hauptservers maskieren. (Auf die Canonical-Datei komme ich später noch zu sprechen). Hier erst mal nur der Eintrag, dass du dich maskieren möchtest.

```
canonical_maps = hash:/etc/postfix/canonical
```

Das SMTP-Banner legt den Begrüßungstext fest, den der smtp - Daemon an die anderen Mailserver überträgt. Grundsätzlich solltest du nicht zuviel von dir verraten, also man sollte nicht Betriebssystem etc. preisgeben. Hier die Zeile für den Hostnamen und den Protokolltyp:

```
smtpd_banner = $myhostname ESMTP
```

Als nächstes die Angabe, wo sich Alias-Einträge befinden. Auch darauf komme ich später nochmal zurück:

```
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
```

Zu guter Letzt setze noch die Gruppenidentifikation und den User-Relay:

```
setgid_group = maildrop
luser_relay = $root@mail.space.mil
```

3.2.4 Adressen richtig umsetzen /etc/postfix/canonical:

Das Umsetzen der Sender-Adresse wird benötigt, da manche Mailclients Probleme mit der lokalen Adresse haben, bzw. diese nicht selbständig umsetzen können. Diese Datei ermöglicht es dir neben Empfängeradressen auch Absenderadressen auszutauschen. Da du damit nicht nur den Austausch im SMTP-Envelope[2] durchführst sondern auch im Mailheader, werden die Adressen völlig transparent ausgetauscht. Für den Empfänger sieht es also aus, als ob die Email, die Du von intern raussendest eigentlich und tatsächlich vom externen Account aus gesendet worden wäre.

Hier ein Beispiel, wie du einige Usernamen umsetzen solltest:

[2]Der SMTP-Envelope ist vergleichbar mit einem Briefumschlag, auf dem u.a. der Absender und Empfänger steht. Mailserver betrachten nur diesen Envelope um Emails zuzustellen und nicht den Inhalt oder Header der Email. Wichtig ist, dass der Envelope nicht gleichzusetzen ist mit dem Mail-Header.

```
roland@internalname.cc      roland@linux-tin.org
root@internalname.cc        roland@linux-tin.org
cybertoon@internalname.cc   cybertoon@linux-tin.org
javier@internalname.cc      javier@linux-tin.org
@internalname.cc            @linux-tin.org
```

Hier habe ich einige Eventualitäten zusammengefasst. Obige Einträge wandeln Mailadressen in das "korrekte" Format um, damit im "Reply to:" - Feld der Antwortadressat steht. Ungünstig übrigens ist es, wenn du deinen Internalname z.B. auf `roland@localhost` mapst, denn damit erzeugst Du eine Mailschleife. Es gibt auch Spezies, die `roland@localhost` auf `roland@linux-tin.org` mappen. Damit würden alle auflaufenden Mails direkt zurück an den externen Mailserver geleitet.

Ist die canonical - Datei erstellt, musst Du diese noch in das Datenbankformat umwandeln mit dem Befehl:

```
linux: # postmap /etc/postfix/canonical
```

Im entsprechenden Verzeichnis gibt es dann neben der Datei `canonical` auch noch eine `canonical.db`.

3.2.5 Email - Aliases

Aliase sind Umleitungsanweisungen für lokale Emailadressen. Sie werden z.B. dazu verwendet, um Systemnachrichten an root in eine Benutzermailbox umzuleiten.

In der Datei `/etc/aliases` sind bereits einige Standardaliases vorgegeben. Diese Datei musst du noch für den eigenen Gebrauch anpassen. So leite ich z.B. alle root - Mails direkt in den Account `roland`, mit dem ich auch meine sonstigen Mails abrufe. Die Datei erklärt sich sehr gut von selbst, weswegen ich sie hier mal nicht komplett angebe. Siehe sie dir an und verändere die entsprechenden Aliases nach eigenem Gebrauch.

Anmerkung: Unter Postfix muss der Account root unbedingt einem eingetragenen User zugewiesen werden, der alle root-mails empfängt, da Postfix als nicht priveligiertes Programm läuft und auf keine Dateien des Benutzers root zugreifen darf.

```
postmaster      root
MAILER-DAEMON:  postmaster
root:           roland
```

Nachdem du die Datei gespeichert hast, führe das Kommando `newaliases` aus - dadurch wird die Datei `aliases.db` erstellt.

3.2.6 Rechtevergabe: /etc/postfix/access

Die Datei `/etc/postfix/access` ist optional und regelt, ob einzelne Adressen, oder ganze Domaenen zum Empfang freigegeben sind oder blockiert werden. Man kann hier nur angeben, ob Emails angenommen oder zurückgewiesen werden. Z.B. lässt sich durch den folgenden Eintrag ein Spamer blockieren:

```
@masterbatesknows.com REJECT
```

Falls du mit dieser Datei weitere Einschränkungen treffen willst, wird eine Menge Information in `/etc/postfix/access` geboten. Vergiß jedoch nicht nach dem anpassen der Datei eine Berkely-Database daraus zu generieren mit dem Befehl:

```
linux: # postmap /etc/postfix/access
```

3.2.7 Postfix (neu) starten

Bevor ich als Nächstes zum Cyrus IMAP Server übergehe, starte Postfix neu. Dazu gibt es (für den User: root) den einfachen Befehl:

```
linux: # rcpostfix reload
```

Läuft Postfix noch nicht, starte den MTA durch:

```
linux: # rcpostfix start
```

Sollte dies wieder erwarten zu einer Fehlermeldung führen, bzw. sich Postfix nicht starten lassen, hilft es meist die Systemstatusmeldungen durchzugehen. Dazu siehst du dir die Dateien `/var/log/messages` und `/var/log/mail` an, z.B. mit dem Befehl:

```
linux: # tail -fn20 /var/log/messages
```

(Das Shell-Programm `tail` zeigt die letzten Zeilen an und erneuert neu hinzugekommende Meldungen sofort. Die Anzeige wird mit einem Break-Signal beendet.)

3.2.8 Test von Postfix

Um den MTA auf Funktion zu testen, kannst du zunächst deine **Prozesse** nach postfix durchsuchen. Nur wenn dieser vorhanden ist, kannst du davon ausgehen, dass dein Mailsystem läuft. Überprüfe also durch:

```
linux: # ps ax | grep postfix
13438 ?          S        3:30 /usr/lib/postfix/master
14001 pts/4      S        0:00 grep postfix
linux: #
```

Falls also obige Ausgabe erscheint, kannst du schon mal sicher sein, dass syntaktisch nichts in deiner Konfiguration falsch gelaufen ist. Als nächstes geht es an die Überprüfung der **Erreichbarkeit**. Dies kann man am einfachsten mit einem simplen telnet auf den SMTP - Port erreichen. Gib also ein:

```
linux: # telnet localhost 25
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to linux.
Escape character is '^['.
220 mailserver.space.mil ESMTP
```

Sehr schön, damit ist klar, dass dein Mailserver korrekt auf Port 25 reagiert. Jetzt geht es noch darum, ob Postfix auch das macht, was du davon erwartest - Emails versenden. Also probiere es aus! Da du dich gerade noch in **Telnet** auf SMTP befindest, kannst du gleich damit testen. Ausserdem kann ich dir damit gleich zeigen, wie Outlook und Co. mit Postfix „sprechen". Gib also weiter folgendes ein:

```
helo roland
mail from: roland@linux-tin.org
rcpt to: christina@kittybond.com
data
Subject: Hello World!
.
quit
```

Würde man mit einem LAN-Sniffer die „Gespräche" eines Email-Clients über das Netzwerk abhören, könnte man in etwa obiges Geplauder wiederfinden. Nun denn, erreicht den Receipient (Empfänger) eine Email, hat alles geklappt und Postfix ist fertig eingerichtet. (Falls du defer_transport eingetragen hast, ist noch ein `sendmail -q` notwendig um die Queue rauszujagen.

3.3 Abschließende Bemerkung

Dass dies noch lange nicht das Ende der Möglichkeiten von Postfix sind, lässt sich anhand der Beschreibungen rund um Postfix - z.B. bei http://www.postfix.org/basic.html erahnen.

4 Cyrus IMAP Server

4.1 Cyrus im Überblick

Dieser IMAP4 Daemon oder IMAP4 Server umfasst eine ganze Reihe von Features, u.a. IMAP Quota oder IMAP ACL-Erweiterungen. Ausserdem lässt sich dieser Server sehr gut skalieren, d.h falls weiterer Platzbedarf notwendig wird muss der Sysadmin nur eine weitere Partition für den Server freigeben.

Die nächsten Punkte sollen das Konzept hinter Cyrus und seine Eigenschaften durchleuchten. Zu tief möchte ich dabei nicht gehen, da dies sonst den Rahmen des HOW-TOs sprengen würde. Für das grundsätzliche Verständnis des Daemons habe ich jedoch alles nötige zusammengefasst.

4.1.1 Das Konzept hinter Cyrus IMAP4

Cyrus verwendet konsequent **hierarchische** Strukturen in seinen Mailboxen. Dies umfasst das gesamte Konzept und du solltest dies stets im Hinterkopf behalten, wenn du administrative Änderungen vornimmst. **Hierarchisch** bedeutet, dass eine Änderung in einer bestimmten Ebene alle tieferliegenden Ebenen ebenfalls davon betroffen sind. Z.B. beim Setzen eines Quotas gilt dies auch bei allen seinen Unterordnern.

Ein kleines Beispiel: Der User *roland*, welcher eine <INBOX> für eingehende Mail und die Unterordner <Drafts>, <Privat> sowie <Sent_Items> angelegt hat. Die Ordner würde Cyrus IMAP4 wie folgt anlegen:

- user.roland
- user.roland.Drafts
- user.roland.Privat
- user.roland.Sent_Items

Zusätzlich enthält jede Mailbox vier Binärdateien:

- cyrus.index
- cyrus.header
- cyrus.cache
- cyrus.seen

Diese Binärdateien dienen zur Performancesteigerung und Verwaltungsaufgaben, die der Server selbständig durchführt. So enthält **cyrus.index** einen Header mit Informationen über die komplette Mailbox sowie ein Feld pro Nachricht. Angaben sind enthalten, die Auskunft über die zuletzt hinzugefügten Nachrichten, den Datumsheader und Größe der Nachricht bereit hält.

Das File **cyrus.header** enthält den Namen des Verzeichnisses, von dem sich die Quota ableitet und eine Kopie der ACLs.

cyrus.cache ist eine reine Performance Datei, die Header jeder Nachricht die ein Client abfragt zwischenspeichert. So muss der Server nicht ständig die Nachrichten nach Headern durchsuchen. Einige weitere Daten, wie Zähler etc. werden ebenfalls gespeichert.

cyrus.seen enthält Daten darüber, wer wann zuletzt auf jene Mailbox zugegriffen hat, sowie Nachrichten, die bereits betrachtet wurden.

4.1.2 Möglichkeiten, die Cyrus bietet

Zugriffskontrollen: Eine Zugriffsliste oder ACL-Liste erlaubt es Benutzern oder Gruppen auf bestimmte Mailboxen zuzugreifen. Hierbei sind bereits zwei Gruppen bei jedem Cyrus Server vorgegeben: *anonymous* und *anyone*. *Anonymous* bezieht sich auf alle nicht authentifizierten Benutzer und kann z.B. spezielle Ordner der Öffentlichkeit freigeben. *Anyone* hingegen besteht aus allen Benutzern inklusive der *Anonymous*-Gruppe.

Bulletin-Boards: Cyrus erlaubt es private Mail-Ordner anderen Benutzern zur Verfügung zu stellen oder gemeinsam genutzte Ordner bestimmte Gruppen oder Benutzern zugänglich zu machen. So können Bulletin-Boards ähnlich Ordnern erstellt werden, bzw. die von Microsoft Exchange her bekannten *Public Folder*s simulieren.

Quotas (Einschränkung des verfügbaren Speicherplatzes): Wie bereits zuvor schon erwähnt ist es im Geschäftsumfeld außerordentlich wichtig, dass Mailspeicher eingeschränkt werden kann. Nicht nur als Sicherheitsmerkmal gegen Angreifer, die einen endlosen Datenstrom in die Mailbox schreiben könnten, sondern auch als Schutz gegen Mitarbeiter, die Emails horten. Übersteigt eine Mailbox einen vorgegebenen Rahmen, ist eine Wiederherstellung extrem aufwendig und zeitraubend. Außerdem wird z.B. die Headerdatei in cyrus.header extrem groß und dementsprechend fehleranfällig - nebenbei steigen die Ladezeiten für eine Mailbox stark an. Du siehst also - Quotas sind ein wichtiger Punkt zu dem ich später auch noch mehr sagen werde.

Partitionierung des Mailspeichers: Unter Cyrus kann der Mailspeicher über mehrere physikalische Platten / Partitionen verteilt werden. Jede Mailbox - Hierarchie wird als Partition bezeichnet. Damit kann der Mailserver mit den bedürfnissen mitwachsen. Sowohl das Auslagern bestehender Mailboxen als auch neue Mailboxen können dann auf das Dateisystem umgelagert werden.

Authentifizierung (z.B. mittels LDAP): Ein sehr wichtiges Merkmal, das ich auch im Folgenden benutzen werde ist der Authentifizierungsmechanismus. Das Übertragen von Passwörtern in Klartext ist an sich eine der schwersten Sünden eines Administrators. Cyrus unterstützt von sich aus Kerberos und die SASL-Bibliothek. Zu den Möglichkeiten von SASL gehören CRAM-MD5, DIGEST-MD5 und GSSAPI (MIT Kerberos 5). Kerberos wird jedoch nicht von allen Clients unterstützt. Auch die UNIX Authentifizierung kann benutzt werden, ich möchte dir jedoch zusätzlich ein PAM - Modul und die openLDAP Methode vorstellen. Natürlich musst du diese nicht verwenden. Hierfür ersetze ich den Cyrus-Authentifizierungsmodus-Daemon (`pwcheck`) durch einen anderen Daemon (`pwcheck_ldap`).

4.2 Konfiguration des Cyrus Servers

4.2.1 Konfigurationsdatei /etc/cyrus.conf

Da der Daemon seine Sockets nicht aus `/etc/inetd.conf` bezieht, gibt es u.a. diese Datei in der alle Informationen hierfür zusammengefasst sind. Normalerweise kannst du diese Datei so weiterverwenden, wie SuSE diese vorgibt. Mitunter ist mir jedoch schon berichtet worden (bei SuSE Version 8.0), dass bei manchen Installationen kein LMTP-Pfad angegeben ist. Suche in der Datei nach dem Kürzel lmtp. Sollten diese auskommentiert sein, so kommentiere die folgende Zeile ein.

```
lmtpunix cmd="lmptd" listen="/var/lib/imap/socket/lmtp" prefork=0
```

Unter früheren Versionen musst du keine Änderung an dieser Datei vornehmen.

4.2.2 Konfigurationsverzeichnise des Servers

Da ich davon ausgehe, dass du SuSE aus der 7er oder 8er - Reihe verwendest, kannst Du die Standardeinstellungen beibehalten und damit sind die Config-Dateien im Verzeichnis `/var/imap` (7.x) bzw. `/var/lib/imap` (8.x) zu finden.

Im Folgenden werde ich der Einfachheit halber die 8.x Pfadangaben verwenden, solltest du SuSE 7.x verwenden, so entferne das `lib` - Verzeichnis (wie oben erwähnt).

`/var/lib/imap/msg/shutdown:` Ist die Datei vorhanden, sendet der imapd die erste Zeile dieser Datei als IMAP ALERT-Nachricht an den Client und beendet die Verbindung. Es können keine neuen Verbindungen aufgebaut werden, bis die Datei gelöscht wurde. (Wird benutzt bei Wartungsarbeiten etc.)

`/var/lib/imap/msg/motd:` Ist die Datei vorhanden, sendet imapd die erste Zeile dieser Textdatei bei jedem Login. Hiermit kann man z.B. auf die Regeln zur Benutzung des Emailsystems hinweisen. (Funktioniert nicht mit jedem Client).

`/var/lib/imap/mailboxes:` Die „mailboxes" - Datei ist entscheidend für die Funktion des Cyrus-Servers und enthält Informationen der einzelnen angelegten IMAP-Folder. Bis zur Version 1.6.22 handelte es sich um eine Klartextdatei. Ab 2.0 wurde diese ersetzt durch eine transaktionsorientierte Datenbankdatei (db-Format). Diese Datei wird sehr häufig aktualisiert und kann mit wachsender Benutzerzahl stark anwachsen. In unserem Fall stellt dies kein Problem dar, da diese Konfigurationsanleitung die Useranzahl auf ein Maximum von 50 voraussetzt.

`/var/lib/imap/log:` Innerhalb dieses Verzeichnises können Sitzungsinformationen und Telemetriedaten in einem Logfile angelegt werden. Der Name der Ausgabedatei ist die Prozessnummer des Servers.

`/var/lib/imap/quota:` Hier werden die Quotas eingetragen und zwar für jedes Wurzelverzeichnis.

`/etc/imapd.conf:` Diese Datei ist die zentrale Konfigurationsdatei für den IMAP Daemon. Im nächsten Punkt gehe ich gerade darauf näher ein.

4.2.3 Konfigurationsdatei /etc/imapd.conf

Der Cyrus Server selbst, bzw. dessen Grundeinstellungen werden in der Datei `/etc/imapd.conf` festgelegt. Insgesamt ist es eine recht einfach aufgebaute Config-Datei, die nach folgendem Schema funktioniert:

Option: Wert.

Man sollte die Anzahl der Einträge so gering wie möglich halten um die Übersicht wahren zu können. Im Folgenden liste ich die wichtigsten Optionen auf, die von SuSE 7.x und 8.x standardmäßig vorgegeben werden:

Option	Standardwert	Wert
`configdirectory`	keiner	Verweis auf das Config - Verzeichnis `/var/imap` (7.x) oder `/var/lib/imap` (8.x)
`defaultpartition`	default	Name der Standardpartition, auf der neue Mailboxen angelegt werden (nicht der Verzeichnisname)
`partition-xxx`	keiner	Erforderlich. Vollständiger Verzeichnisname der Partition xxx. Z.B. `defaultpartition: default` und `partition-default: /anywhere`
`admins`	leer	eine Leerzeichen-getrennte Liste mit Benutzern, die administrative Rechte haben.
`srvtab`	/etc/srvtab	Nur für Kerberos notwendig
`allowanonymous-login`	no	Hier kann man festlegen, ob man sich auch ohne Angabe von Benutzernamen u. Passwort anmelden darf.
`quotawarn`	90	Prozentwert der Quota-Auslastung, ab der der Server dem entsprechenden User eine Warnung sendet.
`timeout`	30	Zeit in Minuten, die eine Sitzung inaktiv sein darf, bevor der Server sie automatisch beendet. Für einen lokalen Server sollte man diese Zeit auf ca. 480 Minuten einstellen.
`autocreatequota`	0	In unserem Fall ist der Standardwert 0 eine gute Wahl
`reject8bit`	no	Ist diese Option nicht gesetzt, werden alle 8-Bit Zeichen im Header durch ein X ersetzt

Im Folgenden gebe ich eine `/etc/imapd.conf` - Datei vor, so wie sie bei dir aussehen sollte (für SuSE Linux ab 8.0 - solltest Du eine frühere Version verwenden, ist jeweils das Verzeichnis `/var/imap` anstatt `/var/lib/imap` einzutragen):

```
configdirectory: /var/lib/imap
partition-default: /var/spool/imap
admins: cyrus root
srvtab: /var/lib/imap/srvtab
allowanonymouslogin: no
autocreatequota: 50000
reject8bit: no
quotawarn: 90
timeout: 480
sasl_pwcheck_method: pam
lmtpsocket: /var/lib/imap/socket/lmtp
```

4.2.4 Test von Cyrus IMAPd

Unter normalen Umständen sollte es dir nun möglich sein, den Daemon zu starten. Benutze also das Kommando:

```
rccyrus start
```

Gab es keine Probleme beim Start des Servers, überprüfe die Erreichbarkeit mit einem telnet wie folgt:

```
linux: # telnet localhost imap
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^['.
* OK localhost Cyrus IMAP4 v2.0.16 server ready
. logout
* BYE LOGOUT received
. OK Completed
```

Antwortet der Server wie oben (oder ähnlich) hast du eine weitere Hürde gepackt. Der IMAP Server läuft und wartet auf Anweisungen. Vorzüglich zunächst administrative Anweisungen.

Sollte wider Erwarten eine sasl - Fehlermeldung auftauchen, musst du das sasl-Paket nachinstallieren bzw. mit dem Befehl `saslpasswd` den User `root` und `cyrus` noch aktivieren.

Im kommenden Abschnitt gehe ich dann auf die Administration des Servers ein und gehe davon aus, dass dein Cyrus-Server lauffähig ist.

Zuvor sollte dir jedoch klar sein, dass ein Administrator bei nur 25 bis 50 Benutzern eine ganz schön umfangreiche Aufgabe zu bewältigen hat. Besonders die Userverwaltung nimmt einen nicht unerheblichen Teil ein. Deshalb empfehle ich eine zentrale Userdatenbank, die wesentlich leichter gepflegt werden kann, da Cyrus die Konten aus dem Linux-System bezieht.

Mein Tipp ist openLDAP zu verwenden (*jedoch nicht zwingend erforderlich*). Einen geeigneten LDAP Browser vorausgesetzt (wie z.B. GQ), ist es ein Leichtes, dass dieser Teil der Aufgabe leichter von der Hand geht. Eine Anleitung und weiterführende Informationen findest du auf http://www.linux-tin.org unter dem Punkt openLDAP.

4.3 Administration des Mailservers

In diesem Abschnitt - nachdem also der Cyrus - Serverdienst von dir installiert und konfiguriert wurde - möchte ich auf jene Dinge eingehen, die zum täglichen Betrieb notwendig sind. Du wirst den Umgang mit dem "Werkzeug" **cyradm** kennenlernen, mit dem du die meisten Aufgaben erledigen kannst. Dazu zählt das Anlegen, Löschen und Bearbeiten von Mailboxen.

Ein paar kurze Worte zu **cyradm** - es handelt sich dabei um ein Tcl - basiertes Clientprogramm, das sowohl interaktiv als auch im Batchmodus eingesetzt werden kann. Ein Mailadministrator kommt nahezu einzig mit disem Tool aus.

4.3.1 Anlegen von Benutzern

Das Anlegen von neuen Benutzern für deinen Cyrus ist in zwei Schritte aufgeteilt. zunächst musst du sicher gehen, dass der Benutzer tatsächlich als UNIX-User existiert. Dies geschieht vorzüglich via LDAP, kann aber auch lokal auf dem Linuxserver mit Shadow/Cleartext - Passwörtern durchgeführt werden.

Lege als den Musterbenutzer `mark` für deine Tests an.

Im zweiten Schritt behandeln wir die Mailbox. Starte hierzu als Benutzer root das Programm `/usr/bin/cyradm`:

```
cyradm -user root localhost
```

Nun sollte sich nach der Passworteingabe der Prompt ändern zu:

```
localhost>
```

Das Kommando um eine neue Mailbox, sprich einen neuen Benutzer anzulegen nennt sich `createmailbox` oder kurz `cm`. Da alle User bzw. deren Mailboxen den Prefix `user.` erhalten, solltest du dies beherzigen und die folgende Befehlsfolge ansehen:

```
localhost> createmailbox user.mark
localhost> listmailbox
user.mark
localhost>
```

Ja, richtig. Das war es auch schon :-). Wenn du die cyradm - Konsole noch mit `exit` verlässt, ist eine Mailbox **mark** für eben diesen Benutzer angelegt und du kannst mit dem entsprechenden Mailclient wie Kmail, Evolution oder Outlook darauf zugreifen. Wie du diese Programme konfigurierst erzähle ich noch im späteren Verlauf.

Bevor ich nun direkt zum nächsten Punkt komme, möchte ich dir die möglichen Kommandos von **cyradm** vorstellen. Dazu kannst du auch am **cyradm** - Prompt ein Fragezeichen (?) eingeben, doch hier eine Tabelle der wichtigsten Funktionen:

Befehl	Erklärung
`listmailbox` oder **`lm`**	Gibt den Namen aller Mailboxen aus. Wildecards sind erlaubt
`createmailbox` oder **`cm`**	Erzeugt eine neue Top_level-Mailbox
`deletemailbox` oder **`dm`**	Löscht eine Mailbox und alle darunterliegenden Mailboxen dieser Hierarchie
`renamemailbox` oder **`renm`**	Umbenennen einer Mailbox
`setaclmailbox` oder **`sam`**	Fügt ACL einer Mailbox neue Zugriffskontrollen hinzu
`deleteaclmailbox` oder **`dam`**	Löscht Einträge aus den Zugriffskontrollen einer Mailbox
`listaclmailbox` oder **`lam`**	Listet die Zugriffskontrollen einer Mailbox auf
`setquota` oder **`sq`**	Setzt Quota für ein Element - Unterordner sind ebenfalls von dieser Quota betroffen
`listquota` oder **`lq`**	Zeigt die Quota für ein Element/Mailbox
`listquotaroot` oder **`lqr`**	Gibt nur die Quotas für die übergeordneten Ordner einer Mailbox aus.

4.3.2 Löschen von Benutzern

Das Löschen von Benutzern ist vermeintlich eine ganz simple Angelegenheit, gibt es doch den befehl dm! Wenn du jedoch in cyradm versuchst - selbst als root - eine Mailbox zu löschen, erhältst du folgende Meldung:

```
localhost> deletemailbox user.mark
deletemailbox: Permission denied
localhost>
```

Sollte da mehr dahinterstecken? Eigentlich nicht - es wurde als Schutz vor Falscheingaben eingebaut. kein Administrator hat eine grundsätzliche Löschberechtigung auf Mailboxen. Diese muss er sich erst expliziet zuweisen. Dazu müssen einige ACLs gesetzt sein. Siehe dir folgende Befehlsfolge an:

```
localhost> setaclmailbox user.mark cyrus cd
localhost> deletemailbox user.mark
localhost>
```

Mit dieser Befehlsfolge machst du eigentlich nichts anderes, als dem Cyrus - Administrator (könnte auch root sein), das Recht zum Löschen (`=d`) und Anlegen einer Mailbox (`=c`) für `user.mark` zu erteilen. Danach stellt das Löschen kein Problem mehr dar.

Auf die vielseitigen Zugriffskontrollen komme ich im nächsten Punkt zu sprechen und gehe dort auch etwas genauer auf die Möglichkeiten ein.

4.3.3 ACLs - Zugriffskontrolle auf Mailboxen

Eine der herausragendsten Eigenschaften des Cyrus IMAP Servers ist die Möglichkeit mit Zugriffsrechten bis hinunter auf unterste Ebene zu arbeiten. Damit lassen sich sogenannte *Public Folders*, also öffentliche Ordner anlegen und z.B. als *schwarze Bretter*

benutzen. Diese wiederum lassen sich so einrichten, dass geschlossene Benutzergruppen mehr Möglichkeiten haben, als anonyme Benutzer.

Das Beste an der Sache jedoch ist die Transparenz und Einfachheit der Verwaltung für den Administrator. Dazu möchte ich dir zunächst mal eine Rechtetabelle vorstellen.

Rechte-Tabelle:

Rechtekürzel	Beschreibung
`l`	Recht, den Namen der Mailbox zu ermitteln, jedoch nicht ihren Inhalt
`r`	Recht, den Inhalt einer Mailbox einzusehen
`s`	Der Status einer Meldung bleibt erhalten. Dies ist z.B. sinnvoll um das *gelesen* oder *neu* - Flag zu erhalten. Der User kann dies nicht abändern, falls dieses Flag gesetzt ist.
`w`	Recht zu schreiben, also das Ändern der Nachrichtenflags *neu*, *beantwortet* und *draft*.
`i`	Recht einzufügen, verschieben oder kopieren einer Nachricht in eine Mailbox
`p`	Senden einer Nachricht an eine Mailbox. Z.B. eine Nachricht an die Cyrushelp Mailbox (sysadmin+cyrushelp@domain.net)
`c`	Anlegen einer Mailbox unterhalb des Top-Levels (was normale Benutzer eigentlich nicht dürfen)
`d`	Löschen einer Nachricht oder der Mailbox selbst
`a`	Administrationsrecht - z.B. für einen Bereichsadministrator, der die Rechte (ACLs) für einzelne schwarze Bretter betreut.

Für häufig verwendete Einstellungen gibt es auch leicht zu merkende Abkürzungen. Auch dafür möchte ich kurz eine Tabelle vorstellen, in der dies zusammengefasst ist.

Abkürzung	Zugriffsrechte	Beschreibung
none	leer	Benutzer vefügt über keine Rechte - wird eingefügt um spezielle Benutzer aus einer allgemeinen Liste auszuschliessen.
read	lrs	Der Benutzer darf den Inhalt der mailbox lesen, ansonsten jedoch besitzt er keinerlei Rechte.
post	lrps	Der Benutzer darf den Inhalt der mailbox einsehen und zudem Nachrichten über IMAP oder deliver direkt senden
append	lrsip	Der Benutzer darf den Inhalt der Mailbox einsehen und zudem Nachrichten über IMAP oder deliver direkt an diese Mailbox anhängen.
write	lrswipcd	Der Benutzer darf den Inhalt der Mailbox einsehen, Nachrichten an diese senden, anhängen und löschen. Einzig die Zugriffsrechte darf der Benutzer nicht ändern.
all	lrswipcda	Hier darf der Benutzer im Gegensatz zu *none* alles. Das heisst selbst die Mailbox löschen und Rechte dieser zu verändern. Mit *all* sollte sehr sparsam umgegangen werden.

Mit diesen Rechten kannst du nun so ziemlich alles anstellen, was dir in den Sinn kommt, sofern du dich als Administrator angemeldet hast natürlich. Wie breits erwähnt, besitzt ein Administrator stets die Rechte `l` (ermitteln des Namens der Mailbox) und `a` (Administration der Mailbox) auf alle angelegten Ordner.

Wird ein neuer Benutzer angelegt, so erhält dieser sämtliche Rechte auf seine eigene Mailbox - alle Ordner unterhalb *erben* die Rechte der übergeordneten.

Natürlich können nicht nur Benutzer mit Rechten ausgestattet werden, sondern auch Gruppen. Diese verhalten sich exakt wie einzelne Benutzerrechte. Was aber passiert, wenn ein User auf eine Mailbox bestimmte Rechte hat, dieser aber einer Gruppe angehört, die andere Rechte besitzt? Der Cyrus IMAP Server rechnet einfach die Vereinigung der Rechte zusammen. Dies werde ich dir aber an einem Beispiel zeigen:

Der Benutzer `mark` gehört der Gruppe `serveradmin` an und folgende Rechte sind auf eine Mailbox vergeben:

```
mark lrs
group: serveradmin lrsp
```

Mark erhält also, da er ja der Gruppe **serveradmin** angehört zusätzlich das Recht `p`.

Dies hat einige Folgen für das weitere Vorgehen. Du musst bei der Rechtevergabe acht geben, welchen Gruppen ein User angehört. Allein mit dieser Art der Rechtevergabe würde jeder Administrator schnell den Überblick verlieren, wenn es nicht noch eine weitere Methode gäbe - den **Rechteentzug**. Sieh dir zunächst folgendes Beispiel an:

```
group: serveradmin lrsp
-mark lrs
```

Dies gibt zunächst der Gruppe serveradmin die Rechte lrsp - entzieht daraufhin jedoch mark das Recht p. Mit Hilfe dieser Methode kannst du dir schnell einen Überblick über die gesamte Rechtestruktur verschaffen.

Wie behandelt Cyrus diese Art der Rechtevergabe? Zunächst wird eine Vereinigung aller Zugriffsrechte berechnet und vergeben. Danach rechnet Cyrus die Vereinigung des Rechteentzugs aus und entzieht der Gesamtmenge wieder die entsprechenden Rechte.

Ohne konkrete Anwendungsbeispiele ist es jedoch schwer, dies verständlich zu erklären, deshalb folgen in den nächsten Punkten einige Möglichkeiten der Verwendung.

Einzelnen Benutzern Rechte gewähren

Stelle dir vor, der Benutzer `roland` möchte seinem Kollegen `mark` Leserechte auf die Mailbox `user.roland.linuxtin-mails` gewähren.

Der Benutzer Roland müsste die Rechte für seine Mailbox wie folgt einstellen:

```
localhost> listaclmailbox user.roland.linuxtin-mails
roland lrswipcda
localhost> setaclmailbox user.roland.linuxtin-mails mark read
localhost> listaclmailbox user.roland.linuxtin-mails
roland lrswipcda
mark lrs
```

Gruppe von Benutzern Rechte gewähren

Um Mitarbeitern einer bestimmten Gruppe - sagen wir mal allen Administratoren den lesenden und schreibenden Zugriff auf eine Gruppenmailbox mit dem Namen admin-news zu erlauben ist die untenstehende Herangehensweise zu verwenden.

Übrigens, alle Gruppen sind normale Unix-Gruppen (z.B. aus der LDAP-Authentifizierung oder `/etc/group`)

```
localhost> listaclmailbox user.admin-news
admin-news lrswipcda
localhost> setaclmailbox user.admin-news group:administrator write
localhost> setaclmailbox user.admin-news tux all
localhost> listaclmailbox user.admin-news
admin-news lrswipcda
group: administrator lrswipcd
tux: lrswipcda
```

Du hast also der Gruppe **`administrator`** und damit allen Benutzern darin volle Rechte (bis auf die Administration) auf diese Mailbox erteilt. Zusätzlich hast du einem Benutzer **`tux`** aus der Gruppe **`administrator`** zusätzlich die Administrationsrechte gegeben.

4.3.4 Quota - Festlegen von Mailboxgrößen

Eine wichtige Instanz für einen Administrator ist das Ausloten der Belastung des Mailservers. Mitunter kann der Mailspeicher selbst eines mittleren Unternehmens imens wertvoll sein. Heutzutage werden bis zu 95% aller Geschäftskorrespondenz über Email abgewickelt - dieser dient dann auch als Informationsquelle für Geschäftsabschlüsse, Verträge und andere wichtige Daten.

Um diesen "wertvollen" Speicher sicher verwalten zu können, muss besonders für seine Ausfallsicherheit gesorgt werden. Eine Methode ist es, Mailboxgrößen so zu beschränken, dass der Speicher des Servers nicht überläuft und eventuell die Datenbank mit sich reisst.

Quotas sind definitiv ein Moment, das jedem Mailserver zu Teil werden sollte um in diesem Bereich gesichert zu sein.

Sicher, dies ist nur ein kleiner Teil um den Mailserver abzussichern, aber ein sehr wichtiger. In der Einleitung habe ich bereits zum Thema Mailboxgrößen mehrere Informationen geliefert - jetzt geht es an die Anwendung.

Zunächst setze ich für die Mailbox des Users **`mark`** eine Maximalgröße von 100MB an:

```
localhost> setquota user.mark 102400
localhost> listquotaroot user.mark
user.mark STORAGE 5120/102400 (5%)
```

Ist einmal eine Quota für eine Mailbox gesetzt kannst du diese nicht mehr entfernen - zumindest nicht auf normalem Wege. Aber du solltest dir im Klaren sein, dass du keine produktiv eingesetzte Mailbox einrichtest ohne Quota. Wird mehr Platz benötigt kannst du diese durch heraufsetzen der Quota problemlos bereitstellen.

So viel zum Thema Quota - nur noch eine Anmerkung: *da Quotas auch dem hierachischem Prinzip gehorchen, kann eine tieferliegende Quota nicht größer sein, als eine höherliegende. Es hat sich eingebürgert, dass Quotas nur auf oberster Ebene eingerichtet werden.*

4.3.5 Bulletin Boards und abteilungsweit offene Ordner

Sowohl Bulletin Boards, als auch offene Ordner sind im Grunde ganz normale Mailboxen, die über entsprechende ACLs mehreren Benutzern einen Zugriff darauf erlauben. Dennoch gibt es gewisse Unterschiede, die ich im Folgenden aufzeigen möchte.

Stelle dir vor, es gibt einen Ordner *shared folde*r oder *offener Ordner*. Dabei handelt es sich meist um einen speziellen Unterordner, der einer speziellen Person gehört die mehreren den lesenden Zugriff darauf erlaubt. Daneben hat dieser meist noch eine weitere signifikante Eigenschaft: Die Statusinformation der enthaltenen Nachrichten werden nicht für jeden Benutzer vermerkt. Dies ist zum Beispiel wichtig, wenn eine Arbeitsgruppe an ein und demselben Fall arbeitet und Nachrichten, auf die bereits geantwortet wurde nicht ein weiteres mal bearbeitet werden. Stattdessen ist der Zustand der Nachricht *Important* oder *Deleted* für alle Benutzer gleich.

Soll der Ordner hingegen als Bulletin Board oder schwarzes Brett eingerichtet werden handelt es sich um eine Mailbox, die keinem speziellem Benutzer gehört, sondern dem System. In diesem Fall sollte für jeden Benutzer einzeln die Statusinformation verwaltet werden. Z.B. wird bei jeder Sitzung und bei jedem einzelnen User der Status *seen* verwaltet. Bulletin Boards werden oft für Foren oder Mailinglisten verwendet.

Einrichten eines *Shared Folder* (*Offener Ordner*):

```
localhost> lam user.public.linuxtin-news
public lrswipcda
mark p
roland p
anyone lr
```

Mit dieser Rechtevergabe besitzen alle User das Recht lesend auf den Unterordner `linuxtin-news` zuzugreifen, die beiden Benutzer `mark` und `roland` zusätzlich das Recht Nachrichten hinzuzufügen. Um z.B. direkt Nachrichten an die Mailbox zu senden, benutzt man die Adresse: `public+linuxtin-news@linux-tin.org`

Aber gehen wir einen Schritt zurück und richten zunächst die Mailbox `public` ein, auf die nur der user `public` selbst Rechte besitzt:

```
localhost> createmailbox user.public
localhost> createmailbox user.public.linuxtin-news
```

Setze nun die Rechte für den Unterordner:

```
localhost> setaclmailbox user.public.linuxtin-news anyone lr
localhost> setaclmailbox user.public.linuxtin-news mark p
localhost> setaclmailbox user.public.linuxtin-news roland p
```

Nun ist der ***Shared Folder*** eingerichtet, der wie oben beschrieben funktioniert.

Einrichten eines Bulletin Boards:

Bulletin Boards werden eigentlich wie obige *Shared Folders* eingerichtet mit dem Unterschied, dass Nachrichten geposted werden dürfen. Dazu setzen wir zusätzlich das p-Flag für alle User, sowie das s-Flag, damit jeder einzelne User sehen kann, welche Nachrichten er bereits gelesen hat und welche noch offen sind.

5 Fetchmail

Bis hierher hast du gehört, wie du mit Hilfe von Postfix über das SMTP ganzen Benutzgruppen den Emailversand ermöglichst. Weiterhin hast du erfahren, wie mit dem Cyrus IMAP Daemon Benutzern ein Werkzeug in die Hand gegeben wird, mit dem sie ihre Emails verwalten können. Was also noch fehlt ist der Empfang von externen Emails. (Die Internen werden ja über Postfix / Deliver verteilt)

Sofern alle Emails über externe ISPs[3] in POP3 - Boxen auflaufen, benötigst du ein zusätzliches Programm, das diese Emails abholt und dem entsprechenden Benutzer zuordnet. Hierfür springt Fetchmail ein und übernimmt diese Aufgabe.

5.1 Grundsätzliches zu Fetchmail

Fetchmail ist allgemein gesagt nichts anderes, als ein Programm, das Emails von einem entfernten Mailserver abholt und auf die lokale Maschine (was auch ein Mailserver sein kann) weiterleitet. Zudem ist es in der Lage auch ganze Mailboxgruppen in Intervallen abzufragen (z.B. einer Domäne) und diese weiterzuleiten. Es versteht die Protokolle POP2, POP3, IMAP2bis, IMAP4 und IMAPrev1.

Ursprünglich wurde es entwickelt um über temporäre Internetverbindungen (Modem, ISDN, DSL usw) Mailboxen abzufragen und lokal abzulegen. Wir benutzen es hier als Schnittstelle zur Außenwelt. Sofern du keine feste IP Adresse besitzt mit eigener, erreichbarer Domäne, benötigst du einen Emailserver der diese Voraussetzungen inne hat. So ein Server wird meist durch einen ISP bereitgestellt bzw. durch einen Enterprise-Server innerhalb eines Unternehmens.

Die Funktionsweise des Programs ist recht einfach erklärt. Es holt die Email nacheinander aus den diversen Mailboxen und verteilt diese über SMTP auf der lokalen Maschine über Postfix - gerade so, als würden die Emails direkt über Port 25 (SMTP) kommen.

[3] ISP = Internet Service Provider

5.2 Konfiguration

Die Konfiguration von Fetchmail erfolgt relativ simpel. Damit du dir eine Vorstellung machen kannst, habe ich hier eine kurze Konfigurationsdatei abgebildet (`/root/.fetchmailrc`):

```
# Configuration - demo for Linux-Tin.ORG
set postmaster "postmaster"
set no bouncemail
poll pop.linux-tin.org with proto POP3
    user 'roland' there
    with password 's3cr3t001' is roland here warnings 3600
    antispam 554
    user 'darkcruix' there
    with password 's3cr3t010' is roland here warnings 3600
    antispam 554
    user 'mark' there
    with password 's3cr3t011' is mark here warnings 3600
    antispam 554
    user 'cybertoon' there
    with password 's3cr3t100' is mark here warnings 3600
    antispam 554
...
poll pop.btx.dtag.de with proto POP3
    user 'roland.huber@t-online.de' there
    with password "s3cr3t101" is roland here warnings 3600
    antispam 554
    user 'darkcruix@t-online.de' there
    with password "s3cr3t110" is roland here warnings 3600
    antispam 554
```

Was du oben siehst ist ein Ausschnitt aus meiner (vereinfachten) `/root/.fetchmailrc`. (Nur so als Anmerkung: Es ist sinnlos, dass du mit Hilfe der Passwörter auf meine Accounts kommst - ich benutze andere :-)

Aber gehe mit mir mal diese Einstellungen durch:

Zunächst setze ich einen **postmaster** fest, der als letzte Instanz jene Mails bekommt, die nicht zugestellt werden konnten (**`set postmaster = "postmaster"`**). Es ist wichtig, dass so ein Benutzer eingestellt wird, damit die Email nicht verworfen wird und zumindest noch der Administrator benachrichtigt wird. Unter **3.3** habe ich dir gezeigt, wie man mit Hilfe der Aliases den Postmaster auf einen tatsächlichen Benutzer umlenkt.

Die folgende Option (**`set nobouncemail`**) legt fest, dass eine nicht - zustellbare Email statt zurück an den Sender an den **postmaster** ausgeliefert wird, der dann entsprechend handeln kann. Bei größeren Sites ist es angebracht diese Option nicht zu verwenden.

Nach diesen Optionen kommen wir zum tatsächlichen Abholen der Email. Um von einem bestimmten Mailserver mehrere unterschiedliche Mailboxen auszulesen beginnst du mit dem poll - Befehl: **`poll pop.dein-provider.de with proto POP3`**.

Damit gibst du an, wo dein Provider die Mails zwischenlagert (Domänenname oder IP) und welches Protokoll dieser verwendet (hier: POP3).

Unterhalb dieses “poll” - Kommandos folgen nun die einzelnen Accounts, die abgeholt werden sollen. Nochmal ein Beispiel daraus:

```
user 'mark' there
with password 's3cr3t011' is mark here warnings 3600
antispam 554
```

mit der Angabe in der ersten Zeile sagst du, dass der Benutzer mit dem Mailkontonamen[4] mark ausgelesen werden soll. In der darauffolgenden Zeile folgt dann das Passwort[5] und die Angabe an welchen lokalen Benutzer die Mail ausgeliefert werden soll. Des weiteren folgt die Option **`warnings`** mit einer Sekundenzahl. Diese Option habe ich gesetzt, damit nach dem Überschreiten der Zeitspanne von 3600 Sekunden dem postmaster eine Nachricht zugesand wird, falls eine zu große Nachricht eintreffen will, die nicht entgegengenommen werden kann.

Mit der Angabe **`antispam 554`** wird fetchmail darauf aufmerksam gemacht, dass Postfix diese Mail als Spam erkannt hat und fetchmail diese somit nicht ausliefern darf - sondern vernichten. (Andere MTAs besitzen etwas andere Fehlercodes hierfür).

Auf diese Weise kannst du nun alle Accounts eintragen - es ist auch erlaubt mehrere Mailboxen auf einen lokalen Benutzer zuzuweisen - auch aus unterschiedlichen Domänen.

Und damit hätten wir **Fetchmail** eigentlich so weit behandelt, dass du in der Lage bist externe Mails abzuholen und auszuliefern.

5.3 Fetchmail manuell starten

Sofern du deine `/root/.fetchmailrc` erstellt hast, kannst testweise die Emails abholen. Dazu gibst du als root das Kommando ein:

```
linux: # fetchmail -d0
```

Diese Eingabe bedeutet, dass alle Accounts, die in `.fetchmailrc` angegeben sind sofort heruntergeladen werden und auf der Konsole Statusmeldungen angezeigt bekommst. Ging dies ohne Fehlermeldungen vonstatten, sollten die entsprechenden Emails auch schon in den lokalen IMAP Mailboxen vorhanden sein.

Nun möchtest du natürlich nicht ständig auf den Server wechseln und als root obiges Kommando eingeben nur um die Emails abzuholen. Deshalb starte Fetchmail als Daemon, der selbständig nach einem angegebenen Zeitraum nach neuen Emails sieht. Das Kommando ist das gleiche, wie oben, nur dass du anstatt der 0 einen Zeitraum in Sekunden angibst.

Also Beispielsweise, wenn du alle 120 Sekunden nach neuen Emails sehen möchtest, benutzt du:

[4] Hier ist der Kontoname des Benutzers gemeint, wie er vom ISP eingestellt wurde

[5] Auch hier gilt wieder das Passwort das benötigt wird um Emails vom Provider abzuholen.

```
linux: # fetchmail -d120
```

Dies bewirkt, dass Fetchmail im Hintergrund startet und automatisch alle 120 Sekunden deine Emails abholt.

5.4 Fetchmail über ein init-Skript starten

So selten man einen Server auch neu startet, aber es ist schon ärgerlich, müsste man jedesmal, wenn dieser hochfährt das Kommando aus 5.3 eingeben, oder?

Also empfehle ich dir ein einfaches Startskript zu verwenden. Hier gleich mal mein Beispiel:

```
#! /bin/sh # Fetchmail start script
# 2002 Roland Huber
#
# /etc/init.d/fetchmail
#
### BEGIN INIT INFO
# Provides: fetchmail
# Required-Start: $remote_fs cyrus postfix $syslog
# Required-Stop:
# Default-Start: 3 5
# Default-Stop:
# Description: start the fetchmail daemon
### END INIT INFO
case "$1" in
start)
echo "Starting Fetchmail Daemon" /usr/bin/fetchmail -a -v -d 120
>>/var/log/fetchmail 2>&1
;;
stop)
echo -n "Shutting Down Fetchmail Daemon" /usr/bin/fetchmail -q
>>/var/log/fetchmail 2>&1
echo
;;
*)
echo "Usage: $0 {start|stop}"
exit 1
esac
exit 0
```

Obiges Skript speicherst du als root unter `/etc/init.d/fetchmail` ab und stattest es mit den gleichen Rechten wie z.B. die Datei postfix im selben Verzeichnis aus. Jetzt musst du nur noch die entsprechenden Links in `/etc/init.d/rc3.d` und `/etc/init.d/rc5.d` erzeugen. Z.B. mit folgenden Befehlen:

```
ln -s /etc/init.d/fetchmail /etc/init.d/rc3.d/S10fetchmail
ln -s /etc/init.d/fetchmail /etc/init.d/rc3.d/K13fetchmail
ln -s /etc/init.d/fetchmail /etc/init.d/rc5.d/S10fetchmail
ln -s /etc/init.d/fetchmail /etc/init.d/rc5.d/K13fetchmail
```

Zu guter letzt noch nach alter SuSE - Manier ein Link auf /sbin mit dem Namen `rcfetchmail`:

```
ln -s /etc/init.d/fetchmail /sbin/rcfetchmail
```

Damit wird Fetchmail beim Systemstart automatisch gestartet. Das stoppen und starten des Dienstes erfolgt entsprechend dem Befehl:

```
linux: # rcfetchmail start
Starting Fetchmail Daemon
linux: #
```

Was es sonst noch zu sagen gibt

Diese Dokumentation sollte es dir ermöglichen einen lokalen Mailserver einzurichten und zu verwalten. Mitunter gibt es vielerlei Tools - ob grafische, webbasierte oder auf der Shell - die das Einrichten erleichtern, aber ich wollte mich da nicht festlegen.

Um diese Dokumentation zu verbessern würde ich mir wünschen, dass du dein Wissen mit mir teilst und zusätzliche Information, bzw. Fehlerkorrekturen zukommen lässt. (roland@linux-tin.org)

Da es sich dabei um Teil I der Dukumentation handelt, sollte es ja auch ein zweiten Teil geben. In dieser Form ist das leider noch nicht der Fall. Den zweiten Teil, der viel weiter in die Tiefe geht, werde ich erst zu einem späteren Zeitpunkt veröffentlichen. Hierzu muss noch viel getestet werden und ich warte noch auf Anregungen von Lesern.

Viel Spass schon mal beim Einrichten,

Roland Huber
April 1999 (Version 0.1) bis Juli 2002 (1.0.9)

Anhang A

Literaturverzeichnis

DIANA MULLET & KEVIN MULLET
Mailmanagement mit IMAP, O'Reilly, 2001, ISBN 3-89721-285-4

PEER HEINLEIN
Das Postfix Buch, SuSE Press, 2002, ISBN 3-935922-41-8

Weblinks

Linux-Tin.org - Seite des Autors:
[http://www.linux-tin.org]
CYRUS IMAP SERVER:
[http://asg.web.cmu.edu/cyrus/]
POSTFIX:
[http://www.postfix.org]
FETCHMAIL:
[http://www.tuxedo.org/~esr/fetchmail/index.html]

Index