

Daten zwischen Notebook und Desktop-PC synchronisieren – mit rsync unter Linux

Dieses Tutorial beschreibt, wie man die Daten zwischen einem Notebook und einem Desktop-PC mit nur einem Mausklick und wenigen Tastendrücken synchronisiert. Verwendet wird das Konsolenprogramm „rsync“ unter Linux per ssh.

Copyright © Gregor Waluga (gregor@waluga.de)
Version vom 18. Februar 2003
Aktuelle Versionen unter: <http://linux.waluga.de>

Dieses Dokument unterliegt der GNU Free Documentation License und darf frei kopiert, verteilt und verändert werden, soweit keine Kosten entstehen und immer auf die GNU FDL verwiesen wird. Diese Seite darf gemäß GNU FDL nicht verändert werden und muss immer mit ausgehändigt werden. Die ganze Lizenz finden Sie unter www.gnu.org/copyleft/fdl.html

Inhalt :

Vorwort

1. Was brauche ich?
 - 1.1 Netzwerk
 - 1.2 rsync installieren
 - 1.3 ssh
 2. Crashkurs in rsync
 - 2.1 Ein paar Funktionen von rsync
 - 2.2 include- / exclude-Dateien
 - 2.3 ssh für rsync einrichten
 - 2.4 Der erste Test
 3. Synchronisation automatisieren
 - 3.1 Shellscripts schreiben – Erklärung von synclap und syncpc
 - 3.2 Geht's noch einfacher?
 4. Hinweise und weitere Informationen
-

Vorwort

„Wie nehme ich nun meine Arbeit mit?“ - Das war meine erste Frage, als ich mein Notebook auspackte. Mühsames hin- und herkopieren schloss ich als fauler Mensch aus. Das Konzept von Linux ein eigenes /home-Verzeichnis zu haben (auf beiden Rechnern), ist dabei sehr praktisch, da alle persönlichen Daten und Einstellungen in einem eigenen Ordner liegen. Schließlich fand ich auf Grund eines Tipps DAS geeignete Tool: rsync.

Es ermöglicht durch seine mehr oder weniger einfache Konfiguration eine perfekte Synchronisation gewünschter Daten / Verzeichnisse und das automatisch, ohne viel Aufwand. Doch ständig eine lange Befehlskette einzugeben, ist für mich ebenfalls zu umständlich gewesen, weshalb ich auch auf eine Automatisierung des Synchronisationsvorganges eingehe.

Dieses Tutorial versteht sich als eine Hilfe zum Kennenlernen des Tools, soll aber die Dokumentation nicht ersetzen. Außerdem ist ein bisschen Mitdenken gefordert, da ich hier nur Beispiele aufführe; jeder muss sich die jeweiligen Befehle individuell umschreiben.

Diese Anleitung ist auch noch nicht fertig. Jeder kann mir gerne neue Ideen und Verbesserungen schicken! Bitte nur konstruktive Kritik. Ich weiß auch, dass das Script nicht das Beste ist und auch diverse Sachen besser / einfacher gemacht werden könnten, doch auch ich bin nur ein Anfänger in dieser Sache.

Nun aber gehen wir schnell ans Werk...

1. Was brauche ich?

Es wird beschrieben, welche Voraussetzungen Sie zum Synchronisieren erfüllen müssen und wie Sie rsync installieren.

1.1 Netzwerk

Die Grundvoraussetzung für das Übermitteln von Daten zwischen zwei Rechnern ist natürlich ein Netzwerk, oder auch LAN genannt. Es muss korrekt eingerichtet und die Hostnamen und lokalen IP-Adressen müssen bereits vergeben sein. Ziehen Sie andere Dokumentationen zum Einrichten eines Netzwerkes zu Rate.

1.2 rsync installieren

Zuerst müssen Sie sich das Paket herunterladen und am besten kompilieren. Sie finden es unter: <http://rsync.samba.org> oder direkt auf <http://rsync.samba.org/rsync/download.html>.

Sie entpacken nun das Paket wie folgt:

```
tar -xvzf rsync-x.y.z.tar.gz
```

und installieren es mit folgenden Befehlen, die Sie nacheinander eingeben:

```
./configure  
make  
su  
[Passwort]  
make install
```

Zu Beachten ist, dass Sie es auf beiden Rechnern (also auf dem Desktop-PC und dem Notebook) installieren müssen. Also wiederholen Sie die Installationsschritte noch einmal.

1.3 ssh

Um eine sichere Datenübermittlung zu gewährleisten, hat sich die SecureShell (ssh bzw. openssh) bewährt. Das Script in Kapitel 3 verwendet ssh standardmäßig, darum wird es auf beiden Systemen vorausgesetzt. Normalerweise wird ssh heutzutage bei allen Distributionen mitinstalliert.

Wenn nicht, dann bekommt man die Pakete auf <http://www.openssh.org>. Entnehmen Sie die Installationsanleitung der beiliegenden README-Datei.

Sie haben nun die Grundvoraussetzungen zum Synchronisieren erfüllt. Natürlich können Sie rsync auf so vielen Rechnern installieren, wie Sie benötigen.

2. Crashkurs in rsync

Damit Sie im folgenden Verlauf Ihren Synchronisationsvorgang individuell gestalten können, werde ich Ihnen paar Grundkenntnisse im Umgang mit rsync vermitteln. Ich weise aber jetzt schon darauf hin, dass ich nur die wichtigsten Parameter erläutere. Wer mehr wissen will, muss sich die sehr gute Anleitung durchlesen, zu finden auf der rsync-Homepage, oder auch auf der Konsole durch die Eingabe von „man rsync“.

Dann bringen wir rsync bei ssh zu verwenden. Anschließend synchronisieren wir als ersten Test ein paar Daten.

2.1 Ein paar Funktionen von rsync

Die allgemeine Syntax von rsync ist wie folgt:

```
rsync [-Optionen] Quelle [Host:]Ziel
```

Optionen:

Wie schon gesagt beschreibe ich im Folgenden nur die wichtigsten bzw. nützlichsten Funktionen. Um das volle Potenzial von rsync auszuschöpfen, lesen Sie die Manpage („man rsync“).

Wie unter Linux üblich, werden die Optionen mit einem Bindestrich „-“ eingeleitet, gefolgt von den Optionskürzeln. Ausgeschriebene Optionen werden mit zwei Bindestrichen eingeleitet. Ich zähle die Kürzel, gefolgt von den ausgeschriebenen Optionsbezeichnungen auf.

-v / --verbose

Zeigt auf dem Bildschirm an, welche Datei gerade kopiert wird.

-a / --archive

Fasst die Optionen recursive (-r: kopiert Unterverzeichnisse), links (-l: kopiert symbolische Links), permissions (-p: behält Rechte bei), times (-t: behält Zeiten bei), groups (-g: behält Gruppenrechte bei), owner (-o: behält Besitzrechte bei) und devices (-D: behält Gerätedateien bei; nur root) zusammen. Es ist eigentlich empfehlenswert diese Option zu benutzen, da die Rechte auf dem Notebook gleich bleiben sollten.

-u / --update

Aktualisiert vorhandene, ältere Dateien und überschreibt keine neueren. Diese Option ist auch empfehlenswert, da Dateien vom Desktop-PC, die schon auf dem Notebook vorhanden sind (z.B. Mails), nicht unnötig kopiert werden, sondern nur, wenn diese neuer sind.

-n / --dry-run

Es testet nur die Synchronisation und zeigt an, welche Dateien kopiert bzw. aktualisiert worden wären. Es werden jedoch keine Daten kopiert.

--existing

Aktualisiert nur vorhandene Dateien, erstellt also keine neuen.

--delete

Bei einer „echten“ Synchronisation ist diese Option sehr nützlich. Bei Verwenden dieser Option werden Dateien auf dem Zielrechner gelöscht, die nicht auf dem Quellrechner vorhanden sind. Das heißt, wenn eine Datei test1 auf dem Zielrechner vorhanden ist, dagegen aber nicht auf dem Quellrechner, dann wird test1 auf dem Zielrechner gelöscht. Quell- und Zielrechner sind somit 100%ig synchronisiert.

--partial

Vor allem beim Synchronisieren über das Internet ist diese Funktion nützlich, da zum Teil übertragene Dateien beibehalten werden. Wenn also eine große Datei transferriert wurde, aber die Verbindung abbricht, dann wird beim erneuten Versuch die Datei zu übertragen, nicht von Anfang an begonnen, sondern macht da weiter, wo aufgehört wurde (ist praktisch das Gleiche wie das so genannte „resumen“ beim herunterladen).

--progress

Zeigt den Fortschritt der Übertragung an.

-P

Fasst die Befehle *--partial* und *--progress* zu einem Befehl zusammen.

-z/--compress

Ist eine nützliche Funktion, vor allem bei langsamen Datenverbindungen. Hier wird der Kompressionsstandard von *gzip* dazu verwendet, die anfallenden Daten zu verkleinern und damit die Übertragung zu beschleunigen.

--include=MUSTER

Schließt im MUSTER definierte Dateien in den Synchronisationsvorgang ein. Nähere Informationen in 2.2.

--exclude=MUSTER

Schließt im MUSTER definierte Dateien aus den Synchronisationsvorgang aus. Nähere Informationen in 2.2.

--exclude-from=DATEI

Schließt die in DATEI durch ein Muster festgelegten Dateien vom synchronisieren aus. Nähere Informationen in 2.2.

--include-from=DATEI

Schließt die in DATEI durch ein Muster festgelegten Dateien vom synchronisieren ein. Nähere Informationen in 2.2.

-e/--rsh=KOMMANDO

Hier kann man eine Alternative zu *rsh* auswählen. Meistens wird eben *ssh* verwendet, weil es die Daten verschlüsselt überträgt. Das Kommando *-e* muss gefolgt von einem Leerzeichen und der zu verwendenden Alternative benutzt werden, also z.B.: *-e ssh*

Vorgehensweise von *rsync*:

rsync vergleicht die vorgegebenen Dateien bzw. Verzeichnisse auf dem Quell- und Zielrechner miteinander. Dabei wird sowohl die Größe, als auch die Erstellungszeit beachtet. Falls die Option *-u* verwendet wird, wird insbesondere auf die Zeit geguckt und die ältere Datei durch die neue ausgetauscht, aber keine neuere Datei durch eine ältere ersetzt – somit stellt diese Option eine Ausnahme von der üblichen Vorgehensweise dar.

Wenn sich nun zwei Dateien voneinander unterscheiden, verwendet *rsync* einen speziellen Algorithmus, der nur den Unterschied zwischen den zwei Dateien überträgt. Somit wird die Synchronisationsgeschwindigkeit enorm gesteigert. Stimmen z.B. von einer 1 MB-Datei, 200 KB nicht mit der anderen Datei überein, werden nur die 200 KB übertragen. Ausgeschlossen sind logischerweise Dateien, die noch nicht auf dem Zielrechner existieren.

Benutzung:

Wie oben beschrieben, hat *rsync* eine ganz bestimmte Syntax. Anhand von Beispielen zeige ich Ihnen jetzt, wie Sie die Quelle und das Ziel genau definieren.

*rsync *.c foo:src/*

Hier wird jede Datei, die mit „*c*“ aufhört aus dem aktuellen Verzeichnis in das Verzeichnis „*src*“ auf den Computer „*foo*“ übertragen.

rsync -avz foo:src/bar /data/tmp

Hier werden alle Dateien rekursiv aus dem Verzeichnis *src/bar* von der Maschine *foo* in das Verzeichnis */data/tmp* auf die lokale Maschine übertragen. Zu beachten ist, dass nur die Dateien im Verzeichnis selbst übertragen werden, nicht jedoch Daten aus Unterverzeichnissen.

*rsync -avz /src/bar/. *foo:/data/tmp/*

Hier werden nur die versteckten Dateien aus dem lokalen Verzeichnis */src/bar/* in das Verzeichnis */data/tmp/* auf die Maschine *foo* übertragen. Der Slash „*/*“ hinter „*bar*“ bedeutet, dass der gesamte Inhalt des Verzeichnisses kopiert wird, was soviel heißt, dass auch Unterverzeichnisse einbezogen werden.

```
rsync -avz /src/bar/* foo:/data/tmp/
```

Hier werden alle Dateien (nicht jedoch versteckte Dateien) aus dem lokalen Verzeichnis /src/bar/ in das Verzeichnis /data/tmp/ auf die Maschine foo übertragen.

Dieses Beispiel zeigt, dass es ziemlich praktisch ist. Will man z.B. das ganze /home-Verzeichnis übertragen, jedoch ohne die persönlichen Einstellungen, die meistens als versteckte Datei abgelegt sind, muss man das mit einem sogenannten Wildcard „*“ definieren. Versteckte Dateien müssen explizit erwähnt werden („.*“). In diesem Beispiel werden auch die Unterverzeichnisse und deren Dateien (außer den versteckten) kopiert.

```
rsync -avz --include=* --include=.* /src/bar/ foo:/data/tmp/
```

Kopiert alle Dateien und alle versteckten Dateien aus dem lokalen Verzeichnis /src/bar/ in das Verzeichnis /data/tmp/ auf die Maschine foo. Unterverzeichnisse werden ebenso einbezogen.

2.2 include- / exclude-Dateien

Diese Möglichkeit bestimmte Dateien ein- oder auszuschließen ist besonders praktisch, da alle Muster in einer Datei gespeichert sind. Es geht auch komplizierter und unüberschaubarer, aber das lassen wir heute lieber! ;-)

Wenn die Option, wie in 2.1 beschrieben, gewählt ist, gleicht rsync jede Datei mit dem Muster in den include- / exclude-Dateien ab und erst dann wird entweder übertragen oder auch nicht.

Wird eine exclude-Datei aufgerufen, werden die darin aufgeführten MUSTER abgearbeitet; somit werden die in MUSTER definierten Dateien beim Synchronisationsvorgang übersprungen. Es ersetzt sozusagen mehrere exclude-Parameter. Das selbe Prinzip gilt bei den include-Dateien, da werden aber die definierten Dateien einbezogen.

Die Regeln für includes / excludes sind eigentlich alle verständlich; im Prinzip sind es die selben, wie bei den Beispielen in 2.1 beschrieben:

- der Wert „/foo“ würde die Datei „foo“ am Anfang des Verzeichnisbaumes treffen
- „foo“ selbst würde jede beliebige Datei treffen, egal in welchem Verzeichnis sie ist
- „foo/“ würde nur ein Verzeichnis treffen, jedoch keine Dateien
- „*.o“ würde alle Dateien mit „.o“ am Ende ein-/ausschließen
- mit „*“ werden alle Dateien, außer den versteckten, ein-/ausgeschlossen
- mit „.*“ werden nur versteckte Dateien ein-/ausgeschlossen

So, nun zur Praxis: Sie erstellen einfach eine Datei, in die Sie untereinander die jeweiligen in- bzw. exclude Muster hineinschreiben, wie etwa hier:

```
--include "/home/gregor/Musik"  
--include "/home/gregor/.kde/share/apps/kmail"  
--exclude ".*"  
--exclude "/home/gregor/.kde/Autostart"
```

Oder aber, Sie machen es sich einfacher und nutzen die Kürzel:

Ein „-“ (Minuszeichen) bedeutet, dass das folgende Muster ausgeschlossen wird (exclude).

Ein „+“ (Pluszeichen) bedeutet, dass das folgende Muster eingeschlossen wird (include).

Dann noch eine Vereinfachung: Die „-“ (Minuszeichen) können auch in include-Dateien aufgeführt sein, und umgekehrt. So können Sie z.B. in einer zentralen Datei alle include-/exclude-Muster ordentlich festhalten. An obiges Beispiel angelehnt sähe das wie folgt aus:

```
+ "/home/gregor/Musik"  
+ "/home/gregor/.kde/share/apps/kmail"  
- ".*"  
- "/home/gregor/.kde/Autostart"
```

Hier noch ein Beispiel, das zeigt, wie man z.B. das ganze /home-Verzeichnis ausschließt, aber trotz des „Verbots“ einzelne Dateien bzw. Verzeichnisse einschließt. Der verwendete Befehl ist

`rsync -e ssh -avzuP --include-from=includefile /home/ 192.168.0.1:/home/` mit folgender include-Datei. Zu

beachten ist, dass man das auszuschließende Verzeichnis am Schluss aufführt, nachdem man die Ausnahmen definiert hat:

```
+ gregor/  
+ gregor/Documents/  
+ gregor/Documents/*  
+ gregor/.kde/  
+ gregor/.kde/share/  
+ gregor/.kde/share/apps/  
+ gregor/.kde/share/apps/kabc/  
+ gregor/.kde/share/apps/kabc/*  
+ gregor/.kderc  
- *
```

Merken Sie was? Sie haben das gesamte Homeverzeichnis ausgeschlossen, wollen aber die Dateien in `/home/gregor/Documents/` einbeziehen. Es reicht nicht aus, ein simples `„/home/gregor/Documents/“` einzutippen, denn dann beziehen Sie nur das Verzeichnis ein, nicht aber die Dateien darin. Außerdem ist es notwendig die übergeordneten Verzeichnisse ebenfalls einzubeziehen. Denn `rsync` hat alle Unterverzeichnisse geblockt und kennt diese daher nicht. Diese müssen von Grund auf wieder „freigeschaltet“ werden. Der Pfad `/home/gregor/.kde/share/apps/kabc/` muss, wie oben gezeigt, Verzeichnis für Verzeichnis eingebunden werden, anschließend die Dateien in dem Verzeichnis.

Des Weiteren haben Sie oben als Stammverzeichnis `„/home/“` gewählt, daher müssen und dürfen Sie in der `include-/exclude-Datei` nicht `/home/` vor jedes Muster schreiben.

2.3 ssh für rsync einrichten

Auch wenn Sie nur zwischen Ihrem privaten Desktop-PC und Notebook Daten verschieben: Sicher ist sicher! Drum wird nun erklärt, wie man `rsync` dazu bringt über `ssh` die Daten zu synchronisieren, damit diese von keinem anderen Benutzer im (W)LAN oder übers Internet abgefangen bzw. manipuliert werden können. Wenn Sie das Script `synclan` und `synclocal` (bzw. `synclap` und `syncpc`) verwenden wollen, müssen Sie diesen Abschnitt nachvollziehen, da `synclan` und `synclocal` standardmäßig `ssh` verwenden.

Zunächst einmal müssen Sie `ssh` so konfigurieren, dass Sie möglichst einfach Daten hin- und her verschieben können. Dazu eignet sich die „Host basierende Authentifizierung“, die allerdings den Nachteil hat, dass sie nicht ganz sicher ist. Sie müssen Ihren privaten Schlüssel unbedingt schützen!!!

Diese Anleitung wurde vorbehaltlich eigener Änderungen, Anpassungen und Verbesserungen mit freundlicher Genehmigung von www.vertexnet.de übernommen. Wer an dem immer noch etwas zu verbessern hat (insbesondere bei der Steigerung der Sicherheit), der kann mir gerne mailen.

Nehmen wir an, Sie wollen als Benutzer „gregor“ Dateien synchronisieren (diese müssen Ihnen auch gehören), dann können Sie eventuell nicht fremde Daten verschieben. Sicher ist es aber allemal, eine Synchronisation NICHT als `root` durchzuführen.

Also zurück zum Beispiel: „gregor“ will vom Desktop-PC Dateien zu einem Benutzer (z.B. auch „gregor“) auf das Notebook verschieben.

So müssen wir auf dem Desktop-PC einen persönlichen Schlüssel generieren und diesen dann dem Notebook bekannt machen. So werden die Desktop-PC-Dateien nur zum passenden Notebook, das den Desktop-PC eindeutig kennt, übertragen. Ist der passende Schlüssel gefunden, ist kein Login auf dem Notebook mehr nötig.

Um das zu erreichen öffnen Sie eine Konsole und gehen wie folgt vor:

Auf dem Desktop-PC wird zunächst `„ssh-keygen -t dsa“` als derjenige Benutzer ausgeführt, der Synchronisieren will (Beispiel „gregor“). Hierbei werden die Schlüssel zur späteren Übertragung generiert, wobei das SSH-Protokoll 2 verwendet wird. Es wird die Datei `„/home/gregor/.ssh/id_dsa“` zum Speichern des privaten Schlüssels vorgeschlagen. Um einen speziellen Schlüssel für das Synchronisationssystem einzurichten, sollte man diesen Pfad ändern z.B. in `„/home/gregor/.ssh/desktop-pc.id_dsa“`. Damit schafft man eine eindeutige Zuordnung der Schlüsselpaare. Als Kennwort für diesen Schlüssel wird `„<ENTER>“` gedrückt, also keins vergeben.

Unter „*/home/gregor/.ssh*“ wird nun eine Textdatei namens „*config*“ mit folgendem Inhalt angelegt:

```
Host notebook
User gregor
Compression yes
Protocol 2
RSAAuthentication yes
StrictHostKeyChecking no
ForwardAgent yes
ForwardX11 yes
IdentityFile /home/gregor/.ssh/desktop-pc.id_dsa
```

Bei „Host“ wird der andere Rechner eingetragen, auf den ohne Login zugegriffen werden soll. Dabei kann man eine IP-Adresse nehmen, oder aber einen Domainnamen (hier eben notebook, weil die IP 192.168.0.2 im lokalen Netzwerk laut */etc/hosts* als „notebook“ definiert ist; auf dem Notebook muss in */etc/HOSTNAME* der Name notebook ebenfalls festgelegt sein).

Sollten für mehrere PCs oder Notebooks unterschiedliche Konfigurationen benötigt werden, muss für jeden Rechner ein zusätzlicher „Host“-Block eingerichtet werden.

Nun noch die Rechte setzen, damit es etwas sicherer ist:

- `chmod 700 /home/gregor/.ssh`
- `chmod 600 /home/gregor/.ssh/*`

Nun bringen wir dem Notebook bei, dass der Desktop-PC auf ihn zugreifen kann:

Der zuvor erstellte ÖFFENTLICHE (!) Schlüssel „*/home/gregor/.ssh/dektop-pc.id_dsa.pub*“ wird nun auf das Notebook übertragen und an die Datei „*/home/gregor/.ssh/authorized_keys2*“ auf dem Notebook (Pfad kann natürlich anders sein) angehängt. Eventuell muss die Datei „*authorized_keys2*“ neu erstellt werden.

Nun noch die Rechte wieder setzen (diesmal auf dem Notebook):

- `chmod 700 /home/gregor/.ssh`
- `chmod 600 /home/gregor/.ssh/*`

So, nun können Sie Daten von Ihrem Desktop-PC auf ihr Notebook ohne Login per ssh übertragen. Wenn Sie auch eine entgegengesetzte Synchronisation durchführen wollen – das heißt Sie sitzen am Notebook und starten von da aus die Synchronisation – müssen Sie angelehnt an das obige Beispiel, einen Schlüssel auf dem Notebook erzeugen und diesen dann dem Desktop-PC bekannt machen. Also einfach die letzten Schritte noch einmal durchführen. Dabei sollten Sie den Schlüssel z.B. „*notebook.id_dsa*“ nennen. Außerdem sollte die config-Datei auf dem Notebook die andere IP bzw. den anderen Domainnamen enthalten. Den ÖFFENTLICHEN (!) Schlüssel des Notebooks („*notebook.id_dsa.pub*“) an die Datei „*authorized_keys2*“ auf dem Desktop-PC anhängen.

2.4 Der erste Test

Es ist simpel und einfach. Wir versuchen einmal, beide */home*-Verzeichnisse zu synchronisieren, aber natürlich nur zum Test, und durch die Option „*-n*“ von *rsync* werden auch keine Daten übertragen. Tippen Sie mal folgende Zeile in die Konsole (natürlich mit Ihrem Homeverzeichnis):

```
rsync -e ssh -avzn /home/gregor/ notebook:/home/gregor/
```

Nun müssten beide Platten losrattern, ohne ein Login-Prompt! Sollte es nicht gehen, ist was schief gelaufen. Suchen Sie den Fehler, indem Sie diese Anleitung noch einmal durchgehen oder mal die FAQ von der *rsync*-Homepage (siehe Kapitel 4) zu Rate ziehen.

Sie können schon loslegen und anhand von Abschnitt 2.1 und 2.2 Ihre ersten Synchronisationen durchführen.

3. Synchronisation automatisieren

So, nun kommt der spannendste Teil an der Sache! Natürlich ist das Eintippen dieser ellenlangen Befehle unpraktikabel. Da ich dazu auch zu faul war, habe ich nach Möglichkeiten gesucht, das Ganze so einfach und komfortabel zu machen, wie nur möglich. Trotzdem kann es sein, dass ich noch nicht alles herausgekitzelt habe, was möglich ist, drum bin ich für Verbesserungsvorschläge dankbar!

3.1 Shellscripits schreiben – Erklärung von synclap und syncpc

Da eine punktuelle Synchronisation erforderlich ist, bieten sich Scripts an, da sie häufig benutzte Befehle bzw. Abläufe automatisieren. Man gibt nur einen kleinen Befehl ein und schon wird eine Befehlskette abgearbeitet.

Dann stand ich da vor dem Problem, dass ich nicht immer wirklich alles synchronisieren will. Z.B. soll meine Musiksammlung, da sie sich selten ändert, nicht synchronisiert werden, wobei man das ab und zu doch braucht. OK, der Zeitgewinn ist minimal, da rsync ohnehin merken würde, dass sich nichts verändert hat, aber trotzdem kam mir die Idee, sich selbst aussuchen zu können, was ich synchronisieren will, und was nicht. So, dann hatte ich alles wunderbar aufgeteilt in Mails, Dokumente, Webserver, Musik usw., aber ein Script könnte ich daraus nicht machen, höchstens ein so genanntes Alias; Nachteil wäre aber, dass ich mehrere Befehle hätte – zu viel für einen faulen Menschen! ;-)

Da kam mir die Idee eines zentralen Scripts, das mich fragt, ob ich meine Musik synchronisieren will, oder nicht, und es soll mich weiter fragen, ob ich meinen Webserver synchronisieren will, oder nicht, usw. Danach soll es die Befehle selbst ausführen und alles komplett schön abarbeiten. Wichtig war mir zudem die leichte Veränderbarkeit, leichte Konfiguration und schnelle Erweiterungsmöglichkeit.

Heraus gekommen sind zwei Scripts, die in ihrer Arbeitsweise eigentlich identisch sind:

- synclan: synchronisiert festgelegte Verzeichnisse oder Dateien vom lokalen System ins LAN („SYNChronize LAN“)
- synclocal: synchronisiert festgelegte Verzeichnisse oder Dateien vom LAN auf die lokale Maschine („SYNChronize LOCAL machine“)

Sie finden diese zwei Scripts in einem Paket auf meiner Internetseite <http://linux.waluga.de> im Verzeichnis „general“.

Ich selbst habe diese Scripts erfolgreich benutzt, bis ich merkte, dass es sehr häufig zur Verwirrung zwischen synclan auf dem PC ausgeführt und dem synclan auf dem Notebook ausgeführt, kommt. Als Folge hatte ich dann paar Mails ins Datennirvana geschickt. :-)

Dies zeigt zum einen, dass man die Scripts konzentriert einrichten MUSS, zum anderen habe ich die Scripts umbenannt, um Verwechslungen auszuschließen:

- synclap: synchronisiert IMMER das Laptop. Egal, ob das Script auf dem Desktop-PC oder auf dem Laptop selbst ausgeführt wird. Auf dem Desktop-PC steckt eigentlich synclan unter der Haube; beim Laptop ist es eigentlich das synclocal Script. Prägen Sie sich aber lieber „synclap“ ein!
- syncpc: synchronisiert IMMER den Desktop-PC. Egal, ob das Script auf dem Laptop oder auf dem Desktop-PC selbst ausgeführt wird. Auf dem Laptop steckt eigentlich synclan unter der Haube; beim Desktop-PC ist es eigentlich das synclocal Script. Prägen Sie sich aber lieber „syncpc“ ein!

Sie sehen, dass es durchaus zu Verwechslungen kommen kann. Daher habe ich in dem Script-Paket das Ganze in Ordner gepackt: In dem Ordner LAPTOP ist alles drin, was auf das LAPTOP drauf soll; in dem Ordner PC ist alles drin, was auf den Desktop-PC kopiert werden soll. Natürlich müssen Sie die Scripts selbst anpassen (und zwar separat). Näheres dazu gleich.

Weil diese Scripts, so denke und hoffe ich, auch für Anfänger geeignet sind, möchte ich die hier etwas beschreiben.

Technisch gesehen fragt das Script, ob etwas synchronisiert werden soll, oder nicht. Dies legt man per Eingabe von „j“ oder „n“ fest. Ist „Ja“ als Standardantwort im Script festgelegt, kann man auch nur die Eingabetaste drücken. Diesen Vorgang wiederholt das Script so lange, bis alle Gebiete, die synchronisiert werden sollen, abgearbeitet sind. Dann eine Frage, ob mit der Synchronisation angefangen werden soll, oder nicht. Bei „j“ werden sofort alle angesammelten Informationen abgearbeitet. Anschließend wird man noch gefragt, ob eine entgegengesetzte Synchronisation mit „syncpc“ bzw. „synclap“ stattfinden soll. Das war's

auch schon zum Prinzip.

Die Programmieretechnik ist, naja, sehr einfach, aber es erfüllt seinen Zweck.

Wert habe ich auf eine zentrale Konfiguration gelegt, wo schnell und einfach z.B. die Synchronisationsverzeichnisse ausgetauscht werden können oder aber die IP des Notebooks verändert werden kann. Hier ein Beispiel, wie so eine Konfiguration für synclap aussieht:

```
IP_LAP=192.168.0.2

# Mails
NAME1=Mails
DATADIR_PC1=/home/gregor/Mail/
DATADIR_LAP1=/home/gregor/Mail/
RS_OPTIONS1="-avuzP --delete"

# Webserver
NAME2=Webserver
DATADIR_PC2=/home/webserver/
DATADIR_LAP2=/webserver/
RS_OPTIONS2="-avuzP --delete"
```

Hier werden mit der IP-Adresse 192.168.0.2 (mein Notebook) meine Mails und mein lokaler Webserver abgeglichen. Dabei befindet sich der Webserver auf dem Notebook an anderer Stelle im Verzeichnisbaum.

Hier nun zur Abfrage und der Funktionsweise des Scripts:

```
# Synchronisationsvorgang Mails; Standard: JA

echo "${NAME1} soll/sollen vom lokalen PC zum Laptop übertragen werden. Fortfahren?"
echo "VON ${DATADIR_PC1} NACH ${IP_LAP}:${DATADIR_LAP1}"
echo -n "J / n : "
read datadirsync1

echo ""
if [ -z "$datadirsync1" ]; then
  datadirsync1=J
fi

case "$datadirsync1" in
  "J" | "j" | "" ) EXECUTE1="rsync -e ssh ${RS_OPTIONS1} ${DATADIR_PC1}
                    ${IP_LAP}:${DATADIR_LAP1}"
                    echo "Daten werden gleich synchronisiert..."
                    echo ""
                    echo
                    "=====
                    echo ""
                    echo ""
                    ;;
  * ) echo "Diese Daten werden NICHT synchronisiert"
      echo ""
      echo
      "=====
      echo ""
      echo ""
esac
```

So, wie man sieht habe ich mit Variablen gearbeitet, wodurch die einfache Konfiguration des Scripts an zentraler Stelle zustande kommt. Nun aber eine Beschreibung der Variablen, damit auch Sie das Script ganz schnell konfigurieren können – es ist jedenfalls einfacher als Sie denken:

LAP = Variable ist auf das Laptop bezogen

PC = Variable ist auf den Desktop-PC bezogen

j und **J** = 'j' ist nicht standardmäßig gesetzt, 'J' ist als Standard festgelegt

n und **N** = 'n' ist nicht standardmäßig gesetzt, 'N' ist als Standard festgelegt

IP_LAP = die IP des Laptops - wahlweise auch mit gültigem Hostnamen

IP_PC = die IP des lokalen PCs – wahlweise auch mit gültigem Hostnamen

NAME_x = Name für den Synchronisationsvorgang (Was wird synchronisiert?)

DATADIR_PC_x = Lokales Verzeichnis auf dem PC, von dem aus bzw. in das die Daten kopiert werden

DATADIR_LAP_x = Verzeichnis auf dem Laptop, das mit dem Verzeichnis auf dem PC (**DATADIR_PC_x**) abgeglichen wird, bzw. von dem aus abgeglichen wird.

RS_OPTIONS_x = Optionen für rsync, müssen mit '-' bzw. '--' angegeben werden (vgl. man rsync).

Die gesamte Variable sollte in Anführungsstrichen " stehen.

EXECUTE_x = Wird am Ende als auszuführender Synchronisationsvorgang eingefügt (siehe unten)

Anmerkung: Das x steht für eine beliebige reale Zahl und symbolisiert die Nummer des Synchronisationsvorganges, die jeweils um 1 zunimmt. Alle Variablen mit einem x in der Vorlage müssen durch das gleiche x ausgetauscht werden. Am Ende muss die Variable \$EXECUTE_x, passend zum Synchronisationsvorgang, gesetzt werden.

So, wenn Sie nun die Bedeutung der einzelnen Variablen verstanden haben, dann gehen Sie ganz einfach ganz runter an das Scriptende. Dort habe ich Vorlagen erstellt, die Sie praktisch nur kopieren und den eigenen Wünschen anpassen müssen. So gehen Sie im einzelnen vor:

1. Im Scriptkopf bei der Variablendefinition müssen Sie die IP des Notebooks bzw. des Desktop-PCs festlegen. Alternativ kann auch ein gültiger Hostname verwendet werden.
2. Für jeden Synchronisationsvorgang benötigen Sie eine Variablendefinition, die Sie als Vorlage unten am Scriptende finden. Diese kopieren Sie oben in den Scriptkopf zu den anderen Variablendefinitionen. Entfernen Sie nun die „#“-Zeichen. Nach fortlaufender Zahl nummeriert, ersetzen Sie alle „x“ bei den Variablennamen durch jeweils die gleiche Zahl. Wählen Sie nun die passenden Werte für die Variablen. Ein Beispiel sehen Sie weiter oben. Beachten Sie bitte, dass die rsync-Optionen aus technischen Gründen in Anführungszeichen stehen müssen.
3. Zu einem Synchronisationsvorgang gehört natürlich die passende Abfrage. Überlegen Sie sich, was Sie gerne als Standard haben möchten: Entweder ein „J“ oder ein „N“, also: Immer aktualisieren oder nie automatisieren. Der Vorteil liegt darin, dass man, wenn man z.B. seine Mails immer synchronisieren will, ein „J“ als Standard definiert. Ist dies gemacht, kann man praktisch nur die Eingabetaste drücken, was gleichzusetzen mit einer Eingabe von „J“ oder „j“ ist. So sparen Sie sich „unnötige“ Tastendrucke. :-)

Kopieren Sie nun die gewählte Abfragen-Vorlage vom Scriptende unter die letzte Abfrage im Script. Entfernen Sie die „#“-Zeichen. Nach fortlaufender Zahl nummeriert, ersetzen Sie alle „x“ bei den Variablennamen durch jeweils die gleiche Zahl; doch hier bitte sehr genau vorgehen! Wenn meistens etwas nicht funktioniert, dann liegt der Fehler in diesem Block. Man vergisst sehr schnell ein „x“ zu ersetzen. Zur Kontrolle: Das „x“ müssen Sie genau 12 Mal durch jeweils die selbe Zahl ersetzen.

Die zur Abfrage passende Variablendefinition muss selbstverständlich jeweils die selbe Zahl am Ende jeder Variable haben, was soviel heißt, dass die Variablennamen identisch sein müssen.

4. Damit das Script nun wirklich schluckt, dass es diese Abfrage auch ausführen soll, müssen Sie die Variable „\$EXECUTE_x“, wobei das „x“ die selbe Zahl sein muss, wie bei passender Variablendefinition und Abfrage, setzen. Die Variable setzen Sie im Abschnitt Synchronisation ziemlich am Scriptende an gekennzeichnete Stelle ein.

Tipp: Auch wenn es mühsam erscheint: Konfigurieren Sie jedes Script einzeln! Das hilft sehr den Überblick zu behalten und vermeidet so Fehler! Denken Sie, dass Fehler in der Konfiguration einen Datenverlust zur Folge haben könnten!

Bevor Sie nun mit der Synchronisation anfangen, ein Rat: Benutzen Sie bei der Variable „RS_OPTIONS_x“ unbedingt die Option „-n“ beim ersten Versuch! So sehen Sie, ob die gewünschten Dateien synchronisiert werden und kein Datenverlust auftritt.

Wenn synclap nun bei Ihnen richtig konfiguriert ist, können Sie nun an syncpc gehen. Dort müssen Sie die

lokale IP eingeben, denn syncpc holt ja aus dem LAN vom Laptop die Datei auf den Desktop-PC. Die Funktionsweise und die Konfiguration ist die gleiche, wie bei synclap.

Am Ende von synclap bzw. syncpc findet eine Abfrage statt, ob das andere Script gestartet werden soll (also wenn synclap läuft, wird gefragt, ob syncpc aufgerufen werden soll). So können Sie eine entgegengesetzte Synchronisation durchführen.

Beispiel: Sie synchronisieren Dokumente mit synclap ohne die Option „--delete“; so werden die auf dem Laptop vorhandenen Dokumente, die aber nicht auf dem Desktop-PC sind, nicht gelöscht. Dann haben Sie auf dem Laptop praktisch mehr Dateien, als auf dem PC. Daraus folgt eine ungleiche Synchronisation. Um nun wieder ein Gleichgewicht herzustellen, werden Sie gefragt, ob syncpc aufgerufen werden soll. Dort nämlich können Sie die Dokumente vom Laptop, die nicht auf dem PC sind, rüberkopieren. Heraus kommt ein völliges Gleichgewicht, da nun auf beiden Computern gleich viele Dateien vorhanden sind. Das nur als Begründung, wieso die Abfrage drin ist.

Beachten Sie bitte, dass synclap und syncpc im „\$PATH“ drin ist, z.B. in „/usr/local/bin“. Ansonsten müssen Sie den Pfad zu den Scripten im Script selbst verändern.

Im Script selbst ist auch beschrieben, wie Sie es konfigurieren können.

Wenn Sie Verbesserungsvorschläge zu dem Script haben, dann bin ich für konstruktive Kritik sehr dankbar!

Sie verwenden das Script auf eigene Gefahr!!!
Bei unsachgemäßer Benutzung kann es zum Datenverlust kommen, für den ich nicht verantwortlich bin!

3.2 Geht's noch einfacher?

Bestimmt! ;-)

Zumindest wollte ich schon immer mit nur einem Mausklick alles in Einklang bringen. In KDE kann man z.B. auf dem Desktop ein Icon erstellen. Dazu einfach mit der rechten Maustaste auf den Desktop klicken, „Neu erstellen...“ und „Verknüpfung mit Programm...“ auswählen. Im Registerreiter „Ausführen“ im Feld „Befehl“ einfach „synclap“ bzw. „syncpc“ eingeben. Wenn syncpc und synclap nicht im \$PATH sind, dann den direkten Pfad eingeben (z.B. /home/gregor/scripts/synclap). Dann weiter unten „In Terminal starten“ ankreuzen. So können Sie schon einfacher eine Synchronisation durchführen.

Man kann die beiden Scripte aber auch per Tastenkombination aufrufen. Ich habe mir einen Trick überlegt: Drücken Sie Alt + F2 oder öffnen Sie eine Konsole. Geben Sie nun „kmenuedit“ ein. Fügen Sie im KDE-Startmenü (oder bei einer SuSE-Distribution im SuSE-Menü) z.B. im Unterordner System ein „Neues Element“ hinzu. Geben Sie nun einen Namen ein, z.B. syncpc bzw. synclap. In der Zeile „Befehl“ geben Sie „synclap“ bzw. „syncpc“ ein. Weiter unten kreuzen Sie „In Terminal starten“ an. Klicken Sie nun auf „Ausgewählte Taste“ in das Feld, wo „Keine“ drin steht. Wählen Sie nun eine Tastenkombination nach Belieben aus. Ich z.B. habe „Alt + Strg + rechts“ für synclap und „Alt + Strg + links“ für syncpc (auf dem PC selbst). Vergessen Sie nicht unten auf „Anwenden“ zu klicken. Jetzt können Sie immer mit dieser Tastenkombination die Scripts aufrufen.

Noch nicht einfach genug?

Nunja, ich habe mir mal das Programm „ifplugd“ (<http://freshmeat.net/projects/ifplugd/>) angeschaut. Es ist speziell für Notebooks gedacht, weil es, direkt wenn das Notebook an den Switch angeschlossen wird, das Netzwerk selbstständig konfiguriert. Ich versuche es mal zu missbrauchen, denn man kann auch, wenn der Stecker in den Switch kommt, eine Synchronisation automatisch durchführen? Oder zumindest eine Abfrage, ob synchronisiert werden soll. Wer das mal ausprobieren will und von Programmieren Ahnung hat, der kann mir gerne eine Mail schreiben.

Wer noch Ideen zur Vereinfachung hat, der kann mir ebenfalls mailen.

So, nach dieser ausführlichen Anleitung sollte es nun jedem gelingen nach einer gewissen Konfigurationsorgie, sein lokales Netzwerk mit Notebook (oder aber auch „festen“ Maschinen) auf gleichem Stand zu halten. Selbst wenn man einen Fileserver benutzt, braucht man die Scripts und vor allem rsync, um seine Arbeit auf dem Notebook mitnehmen zu können. Ich wünsche viel Spaß beim Arbeiten unterwegs!

4. Hinweise und weitere Informationen

Wer intensiver in die Materie einsteigen will, kann sich mal auf folgenden Internetseiten umschauen:

http://rsync.samba.org	Die offizielle Homepage von rsync, DEM Synchronisationstool. Hier findet man auch eine sehr gute FAQ, wenn man Probleme mit rsync hat.
http://www.linux-magazin.de/Artikel/ausgabe/2002/04/rsync/rsync.html	Hier lernt man, wie man einen Server komplett spiegelt.
http://www.linux-user.de/ausgabe/2003/02/082-rsync/index.html	Ein allgemeiner Artikel zur Verwendung von rsync

Das Tutorial und auch die Scripts werden bestimmt mal verbessert... daher sollte man öfters nach einer neuen Version dieses Tutorials schauen, zu finden auf meiner Homepage.

Diese Anleitung wurde nach bestem Wissen und Gewissen geschrieben. Sollten dennoch Fehler enthalten sein, so bitte ich, mir diese mitzuteilen. Für Schäden an Hard- und Software sehe ich mich nicht verantwortlich.