rsync

Linux Programm zur Synchronisation von Laufwerken und Verzeichnissen

Inhalt:

1. Allgemeines

Mittels rsync ist es möglich, manuell und automatisch Datenbestände in verschiedenen Verzeichnisbäumen abzugleichen bzw. zu synchronisieren und somit auf dem gleichen Stand zu halten.

Der Begriff **Synchronisieren** ist allerdings nicht der Richtige, besser wäre die Bezeichnung **Replikation**, d.h. es werden neue oder veränderte Dateien von der Quelle auf das Ziel kopiert.

Zur **echten** Synchronisation sollte man das Programm unison (auch mit grafischer Oberfläche, siehe dazu unten unter Links) verwenden.

Dies kann lokal oder über das Netzwerk (z.B. mittels ssh) erfolgen. Zu bedenken ist hierbei jedoch, dass nicht mehr auf dem Quell-Laufwerk vorhandene Dateien und Verzeichnisse auch auf dem Ziel-Laufwerk gelöscht werden. Beide Verzeichnisse hätten also immer den gleichen Inhalt (synchron).

2. Installation

Falls rsync noch nicht installiert ist, sollte man zuerst die Paketquellen freischalten und anschließend das Paket rsync installieren. Per Terminal wäre der Befehl:

```
sudo apt-get install rsync
```

3. Syntax

Generell ist die rsync-Syntax wie folgt aufgebaut:

```
rsync Parameter Quelle Ziel
```

4. Kurzbeschreibung einiger Parameter

Eine vollständige Liste der Parameter gibt es hier.

-v	verbose	ausgiebiger Bericht der gerade laufenden Bearbeitung
-q	quit	keine Ausgabe von Fehlermeldungen
-a	archive	Zusammenfassung der Parameter -rlptgoD
-r	recursive	Abgleichen mit allen Unterverzeichnissen
-l	links	Kopieren der symlinks
-p	perms	übertragen der Zugrifssrechte
-t	time	Übertragen des Zeitstempels
-g	group	Übertragen der Gruppenrechte
-o	owner	übertragen der Benutzerrechte
-n	Testparameter	Testparameter, die Befehle werden ausgeführt aber es wird nicht kopiert (Testlauf)
-u	update	Bereits vorhadene Dateien werden nur bei neuerem Quelldatum kopiert
-E	execute	Attribut "ausführbar" bleibt erhalten
--delete	delete	löscht nicht mehr vorhandene Dateien auf der Quelle auch auf dem Ziel (synchron)
--exclude	exclude	schließt Verzeichnisse von der Sicherung aus

5. Beispiele zur lokalen Datensicherung

```
rsync -a /home/peter/daten/ /home/peter/sicherung/
```

Hierbei wird das Verzeichnis "daten" des Users "peter" inkl. des Verzeichnisnamens in das Verzeichnis "sicherung" in dessen eigenem Homeverzeichnis kopiert.

```
rsync -a /home/peter/daten /home/peter/sicherung/daten/
```

Hierbei werden nur die Daten aus dem Verzeichnis daten in den Zielordner daten kopiert.

Dateien vom Kopieren ausschließen

Damit nur gewünschte Nutzdaten übertragen werden, können bestimmte Dateitypen von der Verarbeitung ausgeschlossen (exclude) werden:

```
--exclude=*.bak --exclude=.tmp --exclude=*~ Quelle Ziel
```

Um hierbei die Übersicht zu behalten, kann dies in eine Liste der nicht zu kopierenden Dateien ausgelagert werden.

```
--exclude-from="Name und Ort der exclude Datei"
```

6. Automatische Datensicherung

Manuelle Datensicherung ist ja schön und gut, aber warum, wenn es auch automatisch geht.

Linux verfügt über viele hilfreiche Features, um zeitgesteuerte Aktionen auszuführen.

Hierzu gehört der crond, welcher über eine Konfigurationsdatei (/etc/crontab) gesteuert werden kann.

Bindet man nun das Syncronisationsscript in diese Datei (/ect/crontab) ein, kann der Sicherungslauf automatisch Zeit- oder Aktionsgesteuert im Hintergrund angestossen werden.

Erklärungen zu crontab: Die 5 ersten Felder eines Eintrags sind

1. Minute
2. Stunde
3. Tag des Monats
4. Monat
5. Wochentag (0 oder 7 ist Sonntag)

Hiernach würde dieser Crontabeintrag

```
*/15 * * * * peter /home/peter/sicherung_smb
```

alle 15 Minuten an allen Tagen der Woche das Script sicherung_smb mit den Rechten des Users peter aufrufen und ausführen.

Dies eignet sich z.B. für die permanente Datensicherung alle 15 Minuten. Da hierbei nur die veränderten Daten übertragen werden, ist der Zeitaufwand für die Datensicherung relativ gering.

Es eignet sich für permanent eingebundene Ziele genauso wie für Wechselspeichermedien, auf die kopiert werden soll. Was genau passieren soll, wird in dem jeweiligen aufzurufenden Script geregelt.

Man kann auch nur ein Script zur Datensicherung verwenden, in dem alle Arten der gewünschten Datensicherungen auf verschiedene Ziele geregelt werden, je nachdem welche Laufwerke bei der Scriptaktivierung gerade gemountet sind.

Dies hat den Vorteil, dass, egal ob man nun einen Memorystick einsteckt, auf dem eine Verzeichnis-Synchronisation mit der Arbeitsstation erfolgen soll, oder das externe Festplattenlaufwerk eingeschaltet wird, um eine Datensicherung durchzuführen.

Das Script prüft ab, was eingesteckt wurde, und stösst dann alle 15 Minuten die notwendigen Synchronistaionen an.

Externe Laufwerke

Durch externe Speichermedien wie z.B externe Festplattenlaufwerke und Memorysticks kann man es auch dahingehend automatisieren, dass, wenn ein bestimmter Datenträger eingeschaltet oder angesteckt wird automatisch eine Datensicherung (oder Synchronisierung) durchgeführt wird.

Fest gemountete Datensicherungsverzeichnisse

Verfügt man z.B. über einen mount auf einem Server, welcher bereits bei der Anmeldung aktiviert wird, könnte ein solcher **Crontabeintrag** lauten:

```
15 1  * * 1,2,3,4,5,6,7 peter /home/peter/sicherung_smb
```

Dieser Eintrag würde 15 Minuten nach jeder vollen Stunde das aufgeführte Script abarbeiten.

Anmerkungen:

Der Benutzer von rsync muss auf dem Ziel über die notwendigen Rechte zur Dateiablage verfügen (read+write). rsync kann nur Dateien übetragen, auf die der Benutzer von rsync auch die notwendigen Rechte besitzt.

Enthält die Quellangabe als letztes Zeichen einen Slash (/), wird der letze angegebende Verzeichnisname nicht mitkopiert (auf dem Ziellaufwerk neu angelegt), sondern es wird nur dessen Inhalt in das angegebene Ziel kopiert.

Beispiel:

```
rsync -a /home/peter/daten/ /home/peter/sicherung/daten/
```

Kopiert nur die Daten (Dateien und Verzeichniss) aus dem Verzeichnis /daten in den Unterordner /daten des Ziellaufwerks.

```
rsync -a /home/peter/daten /home/peter/sicherung/daten/
```

Dies würde im Unterschied zum ersten Beispiel im Zielordner /daten ein neues Verzeichnis /daten/daten erstellen und den Inhalt dort hinkopieren.

Beim Kopieren (Replikation) verbleiben natürlich Dateien und Verzeichnisse auf den Ziellaufwerk, welche bereits auf der Quelle gelöscht wurden.

Möchte man die **Verzeichnisse syncron** halten, lohnt sich die Verwendung des Programms unison (mit grafischer Oberfläche).

Zu bedenken ist hierbei jedoch, dass nicht mehr auf dem Quelllaufwerk vorhandene Dateien und Verzeichnisse auch auf dem Ziellaufwerk gelöscht werden.

Beide Verzeichnisse hätten also immer den gleichen Inhalt (synchron).

Kleines Script zum Einbinden in die crontab zur automatischen Datensicherung

```
#!/bin/bash
# einfaches Script zur automatischen Datensicherung auf externe Medien
# mittels des Befehls rsync zur Datenreplikation
# gibt es eine Datei unter /media/usbdisk/datensicherung/usb_disk
if test -e /media/usbdisk/datensicherung/usb_disk
# dann führe das Script unter == /home/Benutzername/script/backup_usb_disk ==
aus
then
echo "gefunden"
# Auswahl der Backupfunktion externes File oder direkt mittels rsync
# ausführen eines externen Scripts
# exec /home/Benutzername/script/scriptname
# oder direktes Ausführen des rsync Befehls zur Replikation der Dateien
# rsync -a /vollständige_quelle /vollständiges_ziel
else
echo "nicht gefunden"
fi
```

Automatisches Backup mit dem o.a. Script

Zunächst benötigen wir ein Verzeichnis auf dem Zieldatenträger hier:

```
/media/usbdisk/datensicherung
```

Als nächstes benötigen wir eine Datei in diesem Verzeichnis:

```
touch /media/usbdisk/datensicherung/usb_disk
```

Der Befehl touch legt diese Datei an, jedoch ohne Inhalt, ggf. könnte man hier Infos reinschreiben.

Das eigentliche Sicherungsscript legt man in das Homeverzeichnis von root (/root/script/backup_usb_disk)

oder eben in das eigene Homeverzeichnis

```
/home/Benutzername/script/backup_usb_disk
```

Bitte beachtet hierbei wer das Script ausführen darf (im crontab kann man festlegen mit welchen Benutzerrechten dies geschehen soll) und macht das Script anschließend noch ausführbar.

```
chmod u+x backup_usb_script
```

Im Script selbst muss man ggf. noch das Quell und das Ziellaufwerk sowie die Quell- und Zielverzeichnisse abändern.

Man könnte jetzt verschiedene externe Laufwerke mit verschiedenen Backupfunktionen einrichten und beim Einstecken oder Anschalten des Laufwerks würde beim nächsten Abarbeiten des crond

bzw. der /etc/crontab die gewünschten Datenreplikationen durchführen (je nach dem welche Aktivierungszeit eingestellt wurde).

Man könnte dies alles in ein Script einbauen, wäre einfacher zu pflegen und alles in nur einem Script integriert.

Ich habe hierbei bewusst auf das Einbinden der UID der externen Laufwerke verzichtet, das dies für den Anfänger durchsichtiger erscheint was eigentlich passiert.

Zusammenfassung was benötigt wird:

Auf dem Ziellaufwerk - zum Feststellen was angesteckt wurde:

```
1. ein Verzeichnis   /media/usbdisk/datensicherung
2. eine Datei        /media/usbdisk/datensichrung/usb_disk
```

Im eigenen Home oder Home des root- Benutzers - eigentliches Sicherungsscript:

```
3. das Script        /home/Benutzername/script/backup_usb_disk
```

crontab Eintrag in der Datei:

```
4. /etc/crontab
```

Falls man die automatische Variante nicht will kann man auch einen Starter auf der Desktopoberfläche anlegen um das Sicherungscript zu starten, oder man startet es halt von Hand aus der Konsole heraus.

Hierbei entfällt der notwendige crontab Eintrag.

Die Echoeinträge im Script können entfernt werden, dienen während des Testens nur zur Anzeige ob die gesuchte Datei vorhanden ist oder nicht.

Ebenso muss festgelegt werden ob ein externes Script angestossen werden soll oder ob man hier direkt den rsyn-Befehl benutzen möchte.

Hierzu muss man lediglich vor dem eigentlichen Befehl das Gatter (#) entfernen
(# = auskommentiert, Kommentarzeile ohne Verarbeitung).

7. Links

rsync Homepage

grsync: Grafische Oberfläche für rsync

unison - echte Datensychronisation