

noch ein paar Details zum Schluß ...

Inhalt

- ▶ **FTP**
 - freischalten
 - automatischer Upload
 - Windows-Client
- ▶ **Backup**
 - Archivieren
 - Mirror-Verfahren
 - Clonen
- ▶ noch ein paar Skripte
- ▶ Kylix-Installation

FTP

▶ Freischalten

- in `/etc/inetd.conf` vor entsprechender `ftp`-Zeile das Kommentarzeichen `#` entfernen
- dann noch den `inetd` neu starten:
`/etc/init.d/inetd restart`

▶ Optionen

- evtl. Datei `/etc/ftpuser` anpassen:
Wer darf `ftp` benutzen?
- evtl. anonymes `ftp` einrichten,
aber potenzielles Sicherheitsloch

Windows-Clients

▶ Problem

- wollen einerseits individuelle Profile für FTP-Client
- andererseits wenig administrativen Aufwand
- Vorkonfiguration zur komfortablen Nutzung
- nach Möglichkeit nicht Freigabe von `c:` o.ä.

▶ Lösung?

- Konfigurationsdateien automatisch kopieren
 - Hin-Richtung ganz einfach:
Bei der Anmeldung kann in der `logon`-Batchdatei schon alles wichtige auf die lokale Platte geschrieben werden
 - Rück-Richtung?
Von lokalem System in individuelles Benutzerverzeichnis?
Gibt keine `logoff`- oder `logout`-Batchdatei

here-Operator

- Die **bash** bietet noch ein paar Möglichkeiten mehr als bisher bekannt:
- Mit sog. **here-Operationen** kann „von dieser Stelle“ gelesen werden, nicht aus Datei, nicht aus **stdin** und nicht interaktiv
- Funktioniert als würde Eingabe später manuell (von Tastatur) erfolgen
- Aufruf: `kommando <<Begrenzer
Text, Eingabe, ...
Begrenzer`
- **Begrenzer** ist ein beliebiger Text, häufig **ENDE** oder **EOF**

Beispiel zu here

```
#!/bin/bash
# Beispiel zum here-Operator
cat <<EOF
Das ist der Text,
der sich über viele Zeilen erstreckt
und ausgegeben werden soll.
Normalerweise vor jede Zeile ein „echo“.
EOF
```

- Der Aufruf des Shell-Skriptes gibt den Text aus
Das ist der Text,
der sich über viele Zeilen erstreckt
und ausgegeben werden soll.
Normalerweise vor jede Zeile ein „echo“.

Automatischer FTP-Upload

- Shell-Skript mit folgendem Inhalt:

```
ftp -in domino.informatik.uni-kl.de <<EOF
user d_joniet passwort
binary
cd www
mput *.html
quit
EOF
```

-i	schaltet interaktiv-Modus aus
-n	verhindert automatische Abfrage nach Benutzernamen und Passwort

- lädt automatisch alle `html`-Dateien aus dem aufrufenden Verzeichnis auf den angegebenen Server in das Verzeichnis `www` hoch
- Nachteil: **Passwort im Klartext**

aufräumen

▶ regelmäßig „durchfegen“ per cronjob

- beispielsweise im Tausch-Verzeichnis
- alle Dateien löschen, auf die seit 30 Tagen nicht mehr zugegriffen wurde
- leere Verzeichnisse schon nach 10 Tagen löschen

▶ Idee:

- Dateien suchen, auf die Bedingungen zutreffen
 - `find /daten/tausch -atime 30 \! -type d`
 - `find /daten/tausch -atime 10 -type d -empty`
- dann in Schleife über die Treffer dieselben löschen
 - `rm $i`
 - `rmdir $i`

Archive - gzip

▶ gzip und gunzip

- universell, unterstützt Pipes, Standard
- Aufruf: `gzip datei`
 - Komprimiert `datei`, hängt Endung `.gz` an und löscht Original
- Auspacken: `gunzip datei.gz`
 - Dekomprimiert `datei` und stellt Namen und Rechte wieder her
- Wichtige Optionen:
 - `-c` schreibt auf `stdout`
 - `-d` decompress (wie `gunzip`)
 - `-q` unterdrückt Warnungen
 - `-r` alle Unterverzeichnisse rekursiv bearbeiten
 - `-t` testet komprimierte Datei durch CRC-Test Unversehrtheit
 - `-v` geschwätzig

Archive - zip

▶ zip und unzip

- unhandlich, im Prinzip nur zum Austausch mit Windows-Nutzern
- Aufruf: `zip datei.zip Quelldatei1 ...`
 - packt wie gewohnt alle Quelldateien zusammen in die Zieldatei
 - es können u.U. Dateiname oder Zugriffsrechte verloren gehen
- Auspacken: `unzip datei.zip`
 - Entpackt den Inhalt des Archives
- Nützliche Optionen
 - `-r` die Unterverzeichnisse rekursiv packen

Archive - tar

▶ tar (tape archive)

- universelles Austauschformat, unterstützt inkrementelle Backups, hohe Betriebssicherheit

- *ganz wichtige* Optionen:

-c	create	neues Archiv anlegen
-x	extract	Dateien aus Archiv extrahieren
-t	table	Übersicht über Archivinhalt ausgeben
-d	difference	Unterscheiden sich die Dateien im Archiv von denen im aktuellen Verzeichnis?
-u	update	Dateien im Archiv aktualisieren
-f	filename	Archivname, ohne: Wert von \$TAPE
-v	verbose	geschwätzig
-vv	verbose	noch geschwätziger

Archive - tar

- weitere *wichtige* Optionen:

-z	zip	Archiv mit gzip komprimieren Endung: tar.gz oder .tgz
--use-compress-program	programm	Komprimierungsprogramm oder Verschlüsselungssoftware oder ... verwenden
-N	newerdatum	nur Dateien einpacken, die neuer sind als Datum
-M	multivolume	Wenn sich das Archiv über mehrere Datenträger erstrecken soll
-O	stdout	Auf Standardausgabe schreiben
--same-owner		Besitzer der Dateien beibehalten
--same-permissions		Zugriffsrechte beibehalten
--interactive		Rückfragen an Benutzer stellen

Platten-Mirror

▶ Image eines Gerätes anlegen

```
dd if=/dev/gerät of=/pfad/dateiname
```

- bei Bedarf Plattenplatz sparen durch on-the-fly-Komprimierung

```
gzip < /dev/gerät > /pfad/dateiname
```

– Nachteile

- recht zeitaufwändig
- einzelne Dateien lassen sich kaum rekonstruieren
- umständliches Verfahren beim Rückspielen
- **gefährlich**, wenn Parameter vertauscht werden

– Vorteile

- Eins-zu-Eins-Kopie
- einfach anzufertigen

Kylix-Installation

– Vorbereitung

- Kylix laden, dann:

```
gunzip kylix_oe.tar.gz    dekomprimieren
```

```
tar -xvf kylix_oe.tar    Archiv auspacken
```

```
borpretest                wird's Probleme geben?
```

– Eigentliche Installation starten

- als **normaler Benutzer** oder **root**, je nach Intention

```
setup.sh                  Installation beginnt
```

– Installation abschliessen

```
startkylix                Kylix startet erstmalig!
```

- **Probleme mit SuSE 7.x:**

Entweder Kernel aus der Reihe **2.2.x** installieren oder einfach das „**Generating font matrix**“-Fenster schließen, dann geht's weiter ;-)

- **Product-Key** eingeben und registrieren