Linux Tutorium

5. Archivmanagement

Version vom 11.06.2008 17:26:35

Archivmanagement

- Kompression von einer oder mehreren Dateien
- Unter Windows gängige Formate:
  - ZIP
  - RAR
  - ...
- Unter GNU/Linux gängige Formate:
  - TAR.GZ bzw. TGZ
  - TAR.BZ2 bzw. TBZ

Archivmanagement

- GZ
  - GZ steht für „gzip“ („gunzip“ zum Entpacken)
  - Verlustfreie Kompression von Dateien mittels Deflate-Algorithmus (Kombination aus LZ77 und Huffmann-Kodierung)
  - Relativ schnell bei gleichzeitig relativ guter Kompression

Archivmanagement

- BZ2
  - BZ2 steht für „bzip2“ („bunzip2“ zum Entpacken)
  - Verlustfreie Kompression von Dateien mittels Burrows-Wheeler-Transformation und anschließender Huffmann-Kodierung
  - Bessere Kompression als GZ, dafür aber langsamer

Archivmanagement

- TAR
  - Ist der Name des Programms „Tape ARchiver“ und die Endung der damit erstellten Dateien
  - „tar“ dient zum Erstellen von Archiven, sogenannten Tarballs (Containerdateien, in der alle Dateien enthalten sind) und zum Extrahieren von Dateien aus solchen Archiven
  - Tarballs können von „tar“ entweder auf die Festplatte oder direkt auf das Bandlaufwerk geschrieben werden

Archivmanagement

- In der Regel werden die auf Festplatte geschriebenen Tarballs mit „gzip“ oder „bzip2“ komprimiert
- Durch eine große (= den Tarball) statt vieler kleiner Dateien (= der Inhalt) kann man bei der Kompression die Redundanzen zwischen den Dateien ausnutzen und effizienter Komprimieren (=> kleinere Dateien als Ergebnis)

Archivmanagement

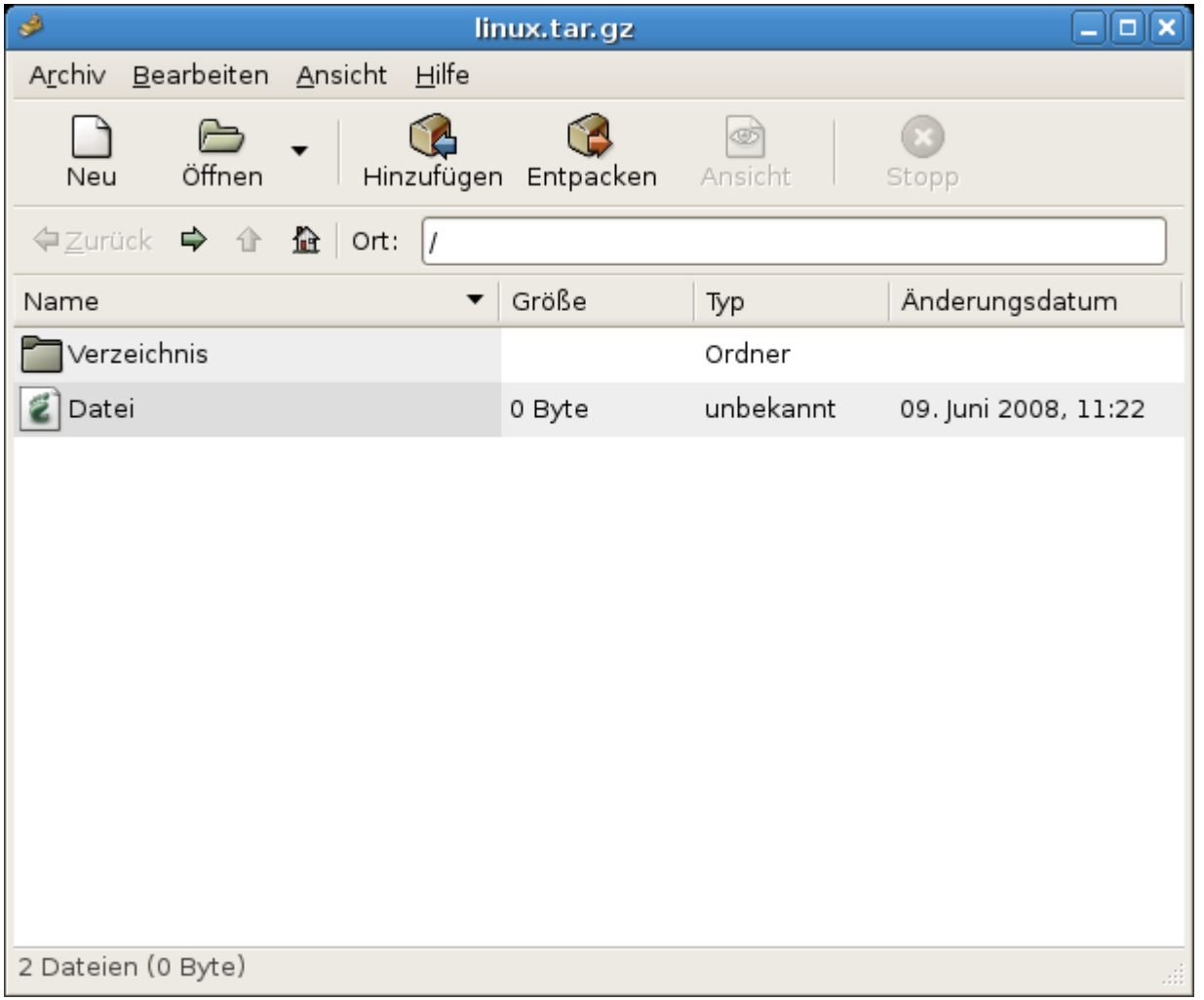
- Anlegen eines Archivs (1)

Archivmanagement

- Anlegen eines Archivs (2)

Archivmanagement

- Entpacken eines Archivs



Linux Tutorium

6. Prozessmanagement

Prozessmanagement

- Prozesse erhalten zur Laufzeit immer die Rechte des aufrufenden Benutzers
- Weiterhin wird jedem Prozess eine PID („Process ID“) zugewiesen
- Über die PID kann der Prozess später eindeutig identifiziert werden

Prozessmanagement

- Es existieren 2 Arten von Prozessen:
  - Hintergrundprozesse, sogenannte Jobs oder Daemons
    - Interagieren nicht direkt mit dem Benutzer
    - Haben i. d. R. ein „d“ am Namensende, dies ist aber nicht zwingend vorgeschrieben
    - z. B. acpid, apache, cron, cupsd, udevd, hald, ...
  - Fordergrundprozesse
    - Interagieren direkt mit dem Benutzer

Prozessmanagement

- Beim Start eines neuen Prozesses wird dieser das Kind des startenden Prozesses
- Auf diese Weise bildet sich ein Prozessbaum, an dessen Spitze der init-Prozess steht
- Wird ein Elternprozess beendet, so reist er alle Kinderprozesse mit in den Tod
- Übersicht über alle laufenden Prozesse liefern die Befehle „top“, „ps aux“ und „pidof programmname“

Prozessmanagement

- Die Kommunikation mit Prozessen erfolgt über Signale, u. a.:
  - [SIG]HUP
    - Der Prozess soll sich beenden und neu starten, um z. B. die Konfigurationsdatei neu einzulesen
  - [SIG]KILL
    - Der Prozess wird sofort durch das System beendet
  - [SIG]TERM
    - Der Prozess soll sich selber beenden
- Signale werden mit dem Befehl „kill“ versendet

Linux Tutorium

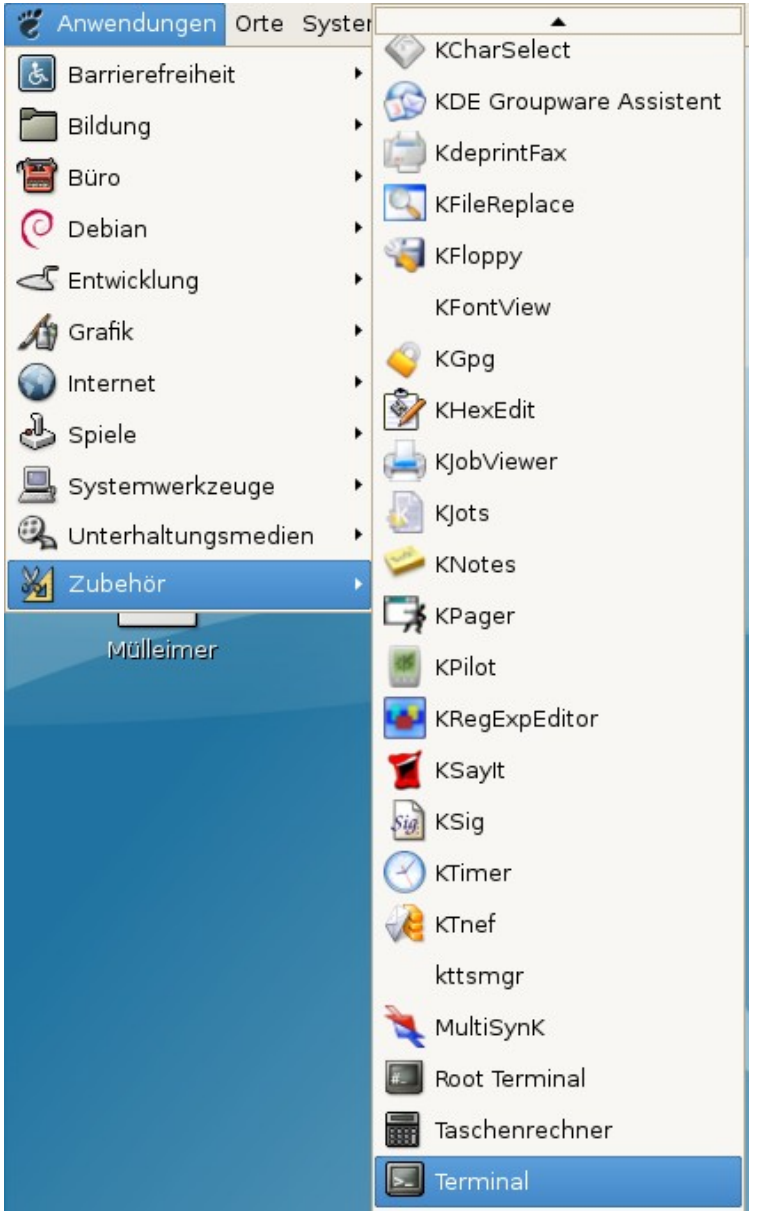
7. Konsole

Konsole

- Die Konsole – häufig auch „Shell“ oder „Terminal“ genannt – ist die CLI-Ebene (Command Line Interface) von GNU/Linux
- Am häufigsten eingesetzte Shell ist die Bourne-Again-Shell „bash“
- Es existieren viele weitere Shells wie z. B. die Kornshell „ksh“, die C-Shell „csh“, die Scheme-Shell „scsh“, ...

Konsole

- Starten des Terminals über „Anwendungen“ => „Zubehör“ => „Terminal“



Konsole

- Informationen, Onlinehandbücher und Hilfe (1)
  - „whatis schlagwort“
    - Liefert Informationen zum angegebenen Schlagwort
  - „whereis schlagwort“
    - Zeigt den Ort der Datei Schlagwort im Dateisystem an
  - „man schlagwort“
    - Öffnet das Onlinehandbuch zu Schlagwort
  - /usr/share/doc/
    - Enthält in den Unterverzeichnissen ebenfalls Dokumentation und Beispielkonfigurationsdateien

Konsole

- Informationen, Onlinehandbücher und Hilfe (2)
  - „whoami“
    - Zeigt den Benutzernamen, mit dem man momentan unterwegs ist
  - „id“
    - Zeigt die Benutzer-ID und die Gruppen, bei denen der aktuelle Benutzer Mitglied ist

Konsole

- Ein Onlinehandbuch – welches mit „man schlagwort“ gelesen werden kann – besteht aus wenigstens folgenden Elementen:
  - NAME
    - Name und Kurzbeschreibung
  - SYNOPSIS
    - Syntaxbeschreibung
  - DESCRIPTION
    - Ausführliche Beschreibung des Kommandos

Konsole

- Arbeiten mit Dateien & Verzeichnissen (1)
  - „cd pfad“
    - Wechselt in den angegebenen Pfad
  - „pwd“
    - Zeigt das Verzeichnis, in dem man sich aktuell befindet
  - „ls“
    - Zeigt den Inhalt des aktuellen Verzeichnisses

Konsole

- Arbeiten mit Dateien & Verzeichnissen (2)
  - „cp quelle ziel“
    - Kopiert eine Datei von der Quelle zum Ziel
  - „mv quelle ziel“
    - Verschiebt die Datei von der Quelle zum Ziel
    - mv muss auch zum Umbenennen verwendet werden
  - „rm datei“
    - Löscht die angegebene Datei
  - Achtung: Sind die Quellen Verzeichnisse, so benötigen „cp“ und „rm“ zusätzlich den Parameter „-r“ (siehe man-Seite)

Konsole

- Arbeiten mit Dateien & Verzeichnissen (3)
  - „touch datei“ bzw. „touch verzeichnis“
    - Ändert den Zeitstempel für die Datei bzw. das angegebene Verzeichnis
    - Ist die Datei nicht vorhanden, so wird sie mit einer Größe von 0 Byte angelegt
  - „mkdir verzeichnis“
    - Legt ein neues Verzeichnis mit dem angegebenen Namen an
  - „ln -s ziel verknüpfungsname“
    - Erzeugt eine symbolische Verknüpfung

Konsole

- Arbeiten mit Dateien & Verzeichnissen (4)
  - „chmod oktalrechte datei-oder-verzeichnis“
    - Ändert die Benutzerrechte
  - „chown benutzer:gruppe datei-oder-verzeichnis“
    - Ändert den Besitzer

Konsole

- Suche nach und in Dateien
  - „find pfad -name "begriff" -ls“
    - Durchsucht den Verzeichnisbaum beginnend beim angegebenen Pfad nach Dateien und Verzeichnissen, die „begriff“ im Namen enthalten und zeigt diese an
  - „grep -slr "begriff" pfad“
    - Durchsucht den Verzeichnisbaum beginnend beim angegebenen Pfad nach Dateien und Verzeichnissen, die „begriff“ als Inhalt enthalten und zeigt diese an

Konsole

- Arbeiten mit (klassischen GNU/Linux) Archiven (1)
  - „tar -cvf archiv.tar datei1 verzeichnis/ datei2 ...“
    - Anlegen eines Tarballs
  - „tar -cvzf archiv.tar.gz datei1 verzeichnis/ datei2 ...“
    - Anlegen eines gzip-komprimierten Tarballs
  - „tar -cvjf archiv.tar.bz2 datei1 verzeichnis/ datei2 ...“
    - Anlegen eines bzip2-komprimierten Tarballs

Konsole

- Arbeiten mit (klassischen GNU/Linux) Archiven (2)
  - „tar -xvf archiv.tar“
    - Extrahieren aus einem Tarball
  - „tar -xvzf archiv.tar.gz“
    - Extrahieren aus einem gzip-komprimierten Tarball
  - „tar -xvjf archiv.tar.bz2“
    - Extrahieren aus einem bzip2-komprimierten Tarball

Konsole

- Ansehen / Editieren von Dateien
  - „less datei“
    - Zeigt die Datei in einem scrollbaren Fenster an
  - „nano datei“
    - Öffnet die Datei in einem relativ einfachen Editor
  - „vi datei“
    - Öffnet die Datei im Profieditor ;-)

Konsole

- Prozessmanagement
  - „pidof programm“
    - Zeigt die PID eines laufenden Programms an
  - „ps aux“
    - Zeigt alle Prozesse des Systems
  - „top“
    - Ein Systemüberwachungstool für die Konsole
  - „kill -signal pid“
    - Schickt das angegebene Signal an die spezifizierte PID

Konsole

- Benutzermanagement
  - „adduser benutzername“
    - Neuen Benutzer anlegen
  - „deluser benutzername“
    - Benutzer löschen
  - „su benutzername“
    - Nach Passworteingaben zum angegebenen Benutzer werden, ohne Angabe des Benutzernamens wird zum Superuser (= root) gewechselt

Konsole

- Dateisystemmanagement
  - „mount“
    - Dient zum Einhängen / Einbinden von Dateisystemen
  - „umount“
    - Hängt eingebundene Dateisysteme wieder aus
  - z.B. mount /dev/sda1 /mnt

Konsole

- 

(I/O – Umleitungen)
  - „ > Datei “
    - Ausgabeumlenkung erfolgt durch ein Größer-als-Zeichen, wobei die Nummer des Deskriptors vorangestellt werden sollte. Wird keine Nummer vorangestellt, wird automatisch die Standardausgabe verwendet.
    - Standartausgabe nach *Datei* umleiten
    - die *Datei* wird wenn noch nicht vorhanden neu angelegt

    z.B. cat /proc/cpuinfo > info.txt oder ls 1> list

Konsole

- „ >> Datei“
  - Manchmal kommt es vor, dass die Ausgabe in eine *Datei* umgelenkt werden soll, in der sich bereits ein Inhalt befindet, der jedoch nicht überschrieben werden soll
  - die Standartausgabe wird an die Datei an gehangen
  - z.B. uname -a >> info.txt
- „ < Datei “
  - Standarteingaben nach *Datei* umleiten
  - z.B. mail root < bericht.xml

Konsole

- Standarddeskriptoren
  - Jedem Prozess sind unter Unix standardmäßig **drei** sogenannte Deskriptoren zugewiesen
    - die Standardeingabe (**0**, STDIN)
    - die Standardausgabe (**1**, STDOUT)
    - die Standardfehlerausgabe (**2**, STDERR)
  - Über diese drei Deskriptoren kann die Ein- und Ausgabe des Programms erfolgen
  - Die Zahlenwerte in den Klammern sind die dem Deskriptor zugewiesenen Nummern, mit denen in der Shell gearbeitet werden sollte

Konsole

- z.B. eine Fehlerumlenkung

```
user$ ls /root
ls: root: Permission denied
user$ ls /root 2>log
user$ cat log
ls: root: Permission denied
```

- Ein spezieller Trick zur Unterdrückung von Fehlermeldungen bietet sich übrigens durch die Umlenkung der Fehlerausgabe auf den »Mülleimer« /dev/null
  - z.B. „find / -name Datei >Ergebnisse 2>/dev/null“

Konsole

- Pipes
  - es ist möglich die Standardausgabe eines Kommandos direkt einem zweiten Kommando als Standardeingabe zu übergeben
  - deshalb stellen Pipes eine Art der Interprozess-Kommunikation dar
  - im Gegensatz zur Ausgabeumlenkung wird die Ausgabe eines Programms nicht in eine Datei geschrieben, sondern direkt an ein Programm weitergeleitet
  - Diese Weiterleitung erfolgt mit einem Pipe-Operator (|). Dieser Strich wird zwischen die beiden Programme gesetzt:

    „Befehl1 | Befehl2“

Konsole

- z.B. „apt-cache show coreutils | grep Size“

```
root@ubuntu:~# apt-cache show coreutils | grep Size
Installed-Size: 11576
Size: 2319994
```

- Die Bash-History
  - sie speichert die zuletzt eingegebenen Befehle
  - man kann diese Befehle abrufen und braucht sie nicht erneut einzugeben, um sie wieder aufzurufen
  - die Bash bietet die Möglichkeit, diese Befehle zu editieren

Konsole

- eine Übersicht über die in der History enthaltenen Befehle liefert ein bloßer Aufruf von „history“
- Cursor-Tasten erleichtern den Umgang mit der History
  - durch die Cursor-Taste »Nach oben« kann der zuletzt eingegebenen Befehl in der Kommandozeile erneut aufgerufen werden, so ist das durchlaufen der History bis an den Anang möglich
  - invers dazu ist dies auch mit der »Nach unten« Taste möglich, wenn man sich in der History befindet, um den gewünschten Befehl zu finden
- auch eine Suche in der History ist relativ einfach möglich

Konsole

- [Strg-r] = Rückwärtiges Suchen im Historybuffer
- [Strg-s] = Vorwärts im Historybuffer suchen

- Kommandos aneinander reihen
  - unter Unix/Linux ist es generell möglich, mehr als nur ein Kommando pro Befehl auszuführen
  - ein Befehl endet erst, wenn er mit der Eingabetaste an die Shell geschickt wird
  - ein Kommando kann von einem anderen jedoch simpler weise durch ein Semikolon (;) getrennt werden

Konsole

- ungeachtet dessen, ob es Probleme mit einem der Kommandos in der Kommandoreihenfolge gibt, die der Shell übergeben werden, laufen alle Kommandos **hintereinander** ab
  - z.B. „ls Verzeichnis ; uname ; find / -name Datei“

- „dmesg“
  - anzeigen der Kontrollmeldungen des Systems aus dem Ringbuffer des Kernels
  - in diesem Buffer sind alle Kernelmeldungen seit dem letzten Hochfahren (wenn der Buffer voll war nur die neusten)

Konsole

- Tab-Vervollständigung
  - 1. Eine Befehlszeilen Oberfläche bedeutet viel Tippen
  - 2. Tippen ist Arbeit
  - 3. Niemand mag Arbeit
  - Aus 2. und 3. können wir ermitteln das (4.) niemand Tippen mag
  - Tab-Vervollständigung funktioniert in etwa so:
    - den Befehl/Datei/Verzeichnis/Pfad beginnen einzugeben
    - drückt man nach ein paar Zeichen <TAB>, versucht die Shell automatisch die gewünschte Eingabe aufzufüllen

Konsole

- 3 mögliche Situationen treten nun ein:
  - 1. es kann keine Vervollständigung durchgeführt werden, der PC-Speaker gibt einen Signalton aus
  - 2. Die Vervollständigung funktioniert aufgrund der Eindeutigkeit der bisherigen Eingabe durch den User und wir sind glücklich
  - 3. es liegt keine Eindeutigkeit vor, es passiert nichts. Wird <TAB> nun doppelt betätigt werden alle in Frage kommenden Möglichkeiten aufgelistet
- Shell-Aliase
  - erlauben es, einen neuen Befehlsnamen zu vergeben, der stellvertretend für ein Kommando oder eine Kommandoreihe steht

Konsole

- ein Alias wird über das gleichnamige Kommando „alias“ erstellt und modifiziert
- einen Alias erstellt man, indem man den neuen Befehlsnamen sowie den Befehl, der hinter diesem verborgen sein soll, dem alias-Kommando übergibt
  - z.B. alias dir="ls -lsa"
- möchte man einen erstellten Alias löschen, ist dies einfach durch Eingabe des Kommando „unalias“ möglich, welchen man den entsprechenden Namen des Alias übergibt welcher entfernt werden soll
  - z.B. unalias dir

Konsole

- eine Liste der bereits vorhandenen Alias-Einträge kann ebenfalls durch den Befehl „alias“ (ohne weitere Parameter) ausgeben werden
- Um den Alias dauerhaft zu speichern, muss man beim Aufruf die Option „-p“ anfügen:
  - alias -p dir=“ls -lsa“