



Linux - Grundlegende Nutzerprogramme

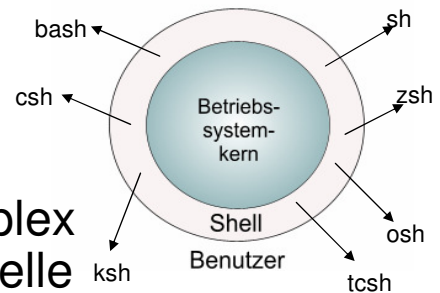
von
Klara Maria Cotarlea

Ruprecht-Karls-Universität Heidelberg
Proseminar: Linux für Umsteiger SS 08
Betreuer: Olga Mordvinova, Julian Kunkel
Heidelberg, den 27.05.08

Inhalt

- Shell
- Kommandos
- Arten des Kommandos
 - Verzeichnis/Dateienverwaltung
 - Datenverwaltung
 - Prozessverwaltung
 - Andere Kommandos
- Fazit
- Quellen

Shell



- Kommunikation Benutzer – Betriebssystem-Kern sehr komplex
→ vereinfachte Benutzer-Schnittstelle
- **Shell (Shalle)**
 - Schicht zwischen Betriebssystem und Benutzer
 - Interpretiert die Kommandozeile
 - Vorteile – schnell, ermöglicht mehrere Befehle gleichzeitig auszuführen
- **Terminal**
 - zuständig für Eingabe und Ausgabe der Daten

3

Aufgrund der komplizierten Kommunikation zwischen Betriebssystem und Benutzer ist Shell (Shalle) entwickelt, die eine Verbindung zwischen Benutzer und Betriebssystem ermöglicht.

Die Shell ist ein Interpreter, ein Programm mit dessen Hilfe das Betriebssystem die Benutzereingaben (Kommandozeile) verstehen kann.

Das Shellprogramm (z.B. bash) wird automatisch gestartet sobald Sie ein Terminalfenster öffnen.

Die Shell ist das Programm, das die Kommandos interpretiert, während in Terminal man die Kommandozeile eingibt und zuständig für die Eingabe und Ausgabe der Daten ist.

Kommandozeile ist eine Möglichkeit mit dem Interpreter (shell) zu kommunizieren.

Es gibt mehrere Arten von Shell: osh, bash, csh...usw.

Shell

- **osh** (Thompson-Shell) - die Standard-Shell des Unix-Systems
- **sh** (Bourne-Shell) - der Vorfahre der heutigen Shells von Stephen R. Bourne
- **bash** (Bourne-again-shell) - Teil des GNU-Projekts, entwickelt am Ende der 80er, Standard-Shell auf den meisten Linux-Systemen
- **csch** (C-Shell) - orientiert an C-Syntax
- **tcsh**, **ksh** (Korn-Shell), **zsh**....usw

4

Im Gegensatz zu anderen Shells ist die osh (Thompson-Shell) keine Programmiersprache, sondern die Steuerung von Programmabläufen war mit einem externen if- und goto-Kommando.

Die Bourne-Shell-Syntax ist Grundlage der meisten modernen Unix-Shells, die eigentlich eine Erweiterung dieser Shell darstellen.

Unter Linux ist die "Bourne Again Shell" ("bash") sehr bekannt.

Bash ist Teil des GNU-Projekts, basiert auf Bourne-Shell-Syntax. Diese Shell ist auf alle Unix-Systeme portierbar, sogar auf Windows. Die bekannteste Portierung für Windows ist cygwin.

C-Shell besitzt viele bekannte Features der bash, wie z. B. Aliase oder eine History. Heutzutage wird die C-Shell nur noch wenig benutzt.

ksh (Korn-Shell) orientiert sich an der Bourne-Shell und übernimmt die Neuerungen der C-Shell.

Inhalt

- Shell
- Kommandos
- Arten des Kommandos
 - Verzeichnis/Dateienverwaltung
 - Datenverwaltung
 - Prozessverwaltung
 - Andere Kommandos
- Fazit
- Quellen

Kommandos – Help

- Benutzung und Syntax des Kommandos
 - \$ befehl --help
 - \$ befehl -h
- Dokumentation des Kommandos
 - \$ man befehl
 - \$ info befehl
- Kurze Beschreibung des Kommandos
 - whatis befehl

6

Die Informationen über jedes Kommandos kann man über help fragen.

Es gibt mehrere Möglichkeiten: wenn man eine ausführliche Dokumentation möchte, dann ist --help, oder man/info geeignet.

Für eine kurze Definition ist whatis geeignet.

Meist ist die ausgegebene Hilfe mit dem --help nicht so ausführlich in Vergleich mit man Dokumentationen die hingegen auch Hintergründe des Kommandos beschreiben und wann man welchen Parameter wie einsetzt.

Kommandos – Syntax

- befehl [Optionen] [Argumente]
- ls -l demo.txt

- **Kommando** – erste Zeichenkette (ls)
- **Optionen** – ändern die Funktionsweise von Kommandos, beginnen oft mit einem - (-l für long list)
- **Argumente** – Dateien (demo.txt)
- **Übergabeparameter** – Optionen und Argumente

- Kommandos sind **casesensitive!**

Kommandos kombinieren

- `Befehl1;Befehl2` – Befehle hintereinander ausführen
- `Befehl1&&Befehl2` – Ausführen von Befehl2 wenn Befehl1 erfolgreich beendet wurde
- `Befehl1||Befehl2` – Ausführen von Befehl2 wenn Befehl1 scheiterte
- `Befehl1|Befehl2` – Ausgabe von Befehl1 als Eingabe für Befehl2 verwenden

8

Es kann komplizierte Kommandos ausgeführt werden indem man mehrere Kommandos kombiniert.

Kommandos kombinieren

- Befehl<Datei – Verwendet Datei als Eingabe für Befehl
- Befehl>Datei – Leitet die Ausgabe von Befehl in Datei um
- Befehl>>Datei – Hängt die Ausgabe von Befehl an Datei an

Inhalt

- Terminal – Shell – Bash
- Kommandos
- Arten des Kommandos
 - Verzeichnis/Dateienverwaltung
 - Datenverwaltung
 - Prozessverwaltung
 - Andere Kommandos
- Fazit
- Quellen

Verzeichnisverwaltung

- **Pwd** – der ganze Pfad des Verzeichnisses wo man gerade sich befindet

- **Cd** – Verzeichnis wechseln
 - `cd ..` für das übergeordnete Verzeichnis
 - `cd ~` oder `cd` für das home Verzeichnis
 - `cd /` für root
 - `cd [DIRECTORY]`

11

Folgende Kommandos sind bei der Arbeit mit Verzeichnisse verwendet.

`pwd`

Das Kommando ‚Print Working Directory‘ zeigt den ganzen Pfad des aktuellen Verzeichnisses.

`cd`

Um in ein anderes Verzeichnis zu wechseln benutzt man das Kommando ‚Change Directory‘.

Bei der absoluten Darstellung geht man immer von der Wurzel (root) aus. Diese Pfadangaben beginnen immer mit einem Slash /

Verzeichnis/Dateienverwaltung

- **mkdir** – erzeugt ein Verzeichnis
 - **mkdir** [OPTION] [DIRECTORY]
 - **mkdir -v** DATEI
- **ls** – zeigt den Inhalt eines Verzeichnisses
 - **ls** [OPTION] [DIRECTORY]
 - **ls -l** DIRECTORY
 - **ls -a** DIRECTORY/*
- **touch** – erzeugt eine leere Datei
 - **touch** [DATEI]

12

Mkdir

Erzeugt ein Verzeichnis, falls es nicht gibt.

Mit der Option `-v` (verbose) es wird eine Nachricht ausgegeben für jedes neu kreierte Verzeichnis. Mit der Option `-p` können Unterverzeichnisse in nicht existierenden Verzeichnissen erstellt werden.

```
mkdir -p -v ss08/Proseminar/LinuxFuerUmsteiger
```

Es wird zuerst das Verzeichnis `ss08` erzeugt, danach innerhalb dessen `Proseminar` und darunter `LinuxFuerUmsteiger`. Ausserdem gibt jeweils eine Nachricht für jedes erstellte Verzeichnis.

Ls

Ermöglicht sich den Inhalt eines Verzeichnisses anzusehen. Mit `-l`, werden sich die Autoren der Dateien anzusehen. Um die versteckten Dateien auch anzuzeigen, muss man zusätzlich als Argument ein `-a`, eingeben.

```
ls -a ss08/Proseminar/LinuxFuerUmsteiger
```

gibt auch die versteckten Dateien die unter Verzeichnis `LinuxFuerUmsteiger` liegen.

Mit der Option `-t` wird eine sortierte Liste anhand der letzten Änderung ausgegeben. Neueste werden zuerst angezeigt. Diese Option ist wichtig bei der Suche nach Dateien in dem Fall dass nur das Änderungsdatum bekannt ist.

touch

ist benutzt bei der Erstellung einer oder mehreren Dateien.

```
touch ss08/stundenplan.txt ss08/Proseminar/LinuxFuerUmsteiger/befehle.txt
```

Das Kommando erstellt 2 leere `txt` Dateien, eine in `ss08` und die andere in `LinuxFuerUmsteiger`.

```
touch text{1,2,3}.txt;ls
```

Erstellt `text1.txt`, `text2.txt`, `text3.txt` und zeigt danach den Inhalt des Verzeichnisses.

Verzeichnis/Dateienverwaltung

- **rmdir/rm** – löscht ein Verzeichnis oder Datei
 - **rm** [OPTION] DIRECTORY
 - **rm -v** DIRECTORY
- **cp** – kopiert eine oder mehrere Dateien
 - **cp** DIRECTORY SOURCE
- **mv** – verschiebt eine Datei oder benennt sie um
 - **mv -v** SOURCE DEST

13

Rm

löscht Dateien. Normalerweise werden die Verzeichnisse nicht mitgelöscht. Um Dateien löschen zu können, benötigt man Schreibrechte in dem jeweiligen Verzeichnis. Wenn diese für das aktuelle Verzeichnis fehlen, muss man für jede Datei das Löschen mit **y** bestätigen oder mit **n** ablehnen. In Verzeichnissen, bei denen das Stickybit gesetzt ist, kann eine Datei nur von ihrem Eigentümer gelöscht werden.

Es besteht nicht unter Linux die Wiederherstellung der Dateien. Man kann die Option **-i** einsetzen um die Kommandos bestätigen zu müssen (mit **y** Taste bestätigen, mit **n** Taste ablehnen).

```
rm -i *.txt
```

Für jede **.txt** Datei muss man das Löschen mit **y** bestätigen oder mit **n** ablehnen.

```
rm -r ss08/Proseminar
```

Löscht das Verzeichnis **Proseminar** mit dazugehörigen Unterverzeichnisse.

Cp

Kopiert eine oder mehrere Dateien von Quellen nach Ziel.

Mv

Bewegt Quellen nach Ziel. Wenn Ziel und Quelle im selben Verzeichnis ist, wird die Datei mit dem neuen Name unbenannt.

```
mv -v *.txt ZielVerzeichnis/ > move.log
```

Dieses Kommando bewegt alle Dateien mit der Endung **"txt"** in das **ZielVerzeichnis**. In das Verzeichnis wo die ***.txt** Dateien waren würde eine Datei **move.log** erstellt und alle Vorgänge aufgezeichnet. Damit kann man immer anschauen, welche Dateien würden bewegt und wo sie zu finden sind.

Inhalt

- Terminal – Shell – Bash
- Kommandos
- Arten des Kommandos
 - Verzeichnis/Dateienverwaltung
 - Datenverwaltung
 - Prozessverwaltung
 - Andere Kommandos
- Fazit
- Quellen

Daten ausgeben

- **more, less** – Datei seitenweise auf der Standardausgabe ausgeben
- **more**
 - **more** [DATEI]
 - **more -d** DATEI
 - **more -d -s** DATEI
- **less**
 - **less -s** [Datei]
 - startet schneller große Dateien als andere Editors (vi)
 - less ist more 😊
- **Unterschied: less** erlaubt auch das Zurückblättern in Texten

15

More und less zeigt den Dateiinhalt in der Standardausgabe seitenweise an.

Less

muss nicht den ganzen Input der Datei vor starten, also bei der großen Datenmengen startet schneller als andere Editoren wie z.B. Vi.

Die grundlegenden Optionen von less und more sind gleich:

-p suchtext zeigt die erste Zeile an, in der der zu suchende Text gefunden wurde.

-s fasst mehrere Leerzeilen zu einer zusammen.

-d Benutzer kann per Leertaste in der Datei zu blättern.

Beide Programme können mit der Taste q beendet werden. Der Unterschied zwischen die beide Kommandos ist dass man mit less zurückblättern kann. Daher ist less eine bessere Alternative zu more.

Daten ausgeben

- **head** – Anfangszeilen einer Datei ausgeben
 - **head** [Optionen] [Datei...]
 - **head -3** [Datei]
- **tail** – Letzte Zeilen einer Datei ausgeben
 - **tail -3** [Datei]
- **nl** – Daten mit Zeilennummern ausgeben
 - **nl** [Datei]

16

Head

schreibt die ersten (10) Zeilen von der Datei auf der Standardausgabe. Wenn keine Datei oder '-' angegeben wird, liest head von der Standardeingabe. Wird mehr als eine Datei angegeben, so wird der Dateiname, in '==>' und '<==>' eingeschlossen, der Ausgabe vorangestellt.

Option: -n - gibt die erste N Zeilen der Dateien aus.

```
head -2 -q text1.txt text2.txt > text3.txt
```

Erstellt eine Datei text3.txt, schreibt in text3.txt die erste zwei Zeilen von text1.txt und von text2.txt ohne die Dateinamen auszugeben -> Option -q (quiet).

Tail

schreibt die letzten (10) Zeilen von der Datei auf der Standardausgabe. Wenn keine Datei oder '-' angegeben wird, liest head von der Standardeingabe. Wird mehr als eine Datei angegeben, so wird der Dateiname, in '==>' und '<==>' eingeschlossen, der Ausgabe vorangestellt.

Option -n - gibt die letzte N Zeilen der Dateien aus.

```
tail -2 text1.txt text2.txt
```

Schreibt auf der Standardausgabe die Dateinamen und die letzte zwei Zeilen von jeder Datei.

```
tail -2 - << ENDE
```

Input wird von der Standardausgabe gelesen bis man ENDE schreibt. Gibt die letzte zwei Zeilen aus.

Daten ausgeben

- **cat** – Daten nacheinander ausgeben
 - **cat** [Optionen] [Datei1] [Datei2] ... [DateiN]
 - **cat -s** [Datei1] [Datei2] ... [DateiN]
- **tac** – Daten in umgekehrter Reihenfolge ausgeben
 - **tac** [Datei1] [Datei2] ... [DateiN]
- **Od, hd/hexdump** – Daten in oktal/hexadezimal ausgeben
 - **od** [Datei]
 - **hd** [Datei]

17

Cat

hängt die Dateien aneinander und schreibt die in Standardausgabe.

Optionen für cat:

-n - die Zeilen der Textausgabe erhalten eine Zeilennummer

-s -reduziert die Anzahl der Leerzeilen, es wird maximal eine Leerzeile angezeigt

Die Dateien sind optional, weil cat auch von der Standardeingabe lesen kann (z.B. in einem Pipe-Kontext).

```
cat -n 1.txt 2.txt > out.txt
```

Schreibt in out.txt die Dateien 1.txt und 2.txt mit durchnummerierten Zeilen.

```
cat > listTast.txt << ENDE
```

Es wird eine txt Datei erzeugt. Die Tastatureingaben werden in die Datei geschrieben, bis man ENDE (oder was nach dem << folgt) eingibt.

```
ls -l *.mp3 | cat -n
```

Ausgabe der mp3 Dateien. Das Ergebnis des ls-Kommandos wird an cat weitergereicht (gepiped). Cat liest in diesem Fall von der Standardeingabe. Diese wird dann durch der Option -n durchnummeriert. Das Ergebnis ist eine durchnummerierte Liste aller mp3-Dateien im Verzeichnis.

Tac

hängt die Dateien aneinander wie das Kommando cat, aber schreibt das Ergebnis in umgekehrter Reihenfolge in der Standardausgabe.

Hexdump und od zeigt Dateien in hexadezimalen, dezimalen, oktalen oder ascii Zeichenformat an.

Dieser Befehl wird meistens zur Ausgabe von Binärdateien eingesetzt.

Daten – umformen

- **split** – Zerteilen von Dateien
 - **split** [OPTION] [INPUT]
- **sed** – transformiert die Zeichenketten
 - **sed** [OPTION] [DATEI]
 - **sed** 's/Nr/Nummer/g' [SOURCE] > [DEST]

18

Split

zerteilt eine Datei in mehrere gleich große Teile.

```
split -15 adressen.txt newadress.txt
```

Die Datei adressen.txt wird in Teilstücke zu jeweils 15 Zeilen zerlegt werden. Die originale Datei bleibt unverändert, die neue Dateien heißen newadress.txtaa, newadress.txtab usw.

Sed

Der Name Sed kommt von Stream Editor und ist ein mächtiges Unix-tool, mit dem Texte geändert werden können. Sed wird benutzt indem man einen regulären Ausdruck in zwei Slashes schreibt und den ganzen Muster in einfache Hochkommata, damit die Shell nicht die Metazeichen interpretiert. Modifiziert nicht das Original, sondern schreibt das Ergebnis auf die Standard-Ausgabe oder in einer Datei. Sed ist oft benutzt als suchen und ersetzen tool.

Sed kann man bei der Umsetzung der Texten in verschiedene Fonologien, wie z.B. CP-5, Sampa,..usw. verwendet (bei der Spracherkennungssysteme), indem alle Buchstaben durch andere Zeichen ersetzt wird. Man schreibt eine Liste mit einer Regel pro Zeile. zB: in der Datei sedrulesCP5.txt

```
s/hio/_C2_Y. O_S /g
```

```
s/iu/l_S_W. /g
```

```
s/ng/_NC_G./g
```

```
s/ k/ K_R/g
```

```
s/x/_KS /g
```

#Man kann in der Datei Kommentare fügen die mit '#' anfangen.

```
sed -f sedrulesCP5.txt in.txt > out.txt
```

Die Datei in.txt enthält ein normalen Text der in CP5 übersetzt wird und dannach das Ergebnis in der Datei out.txt geschrieben.

```
sed -e "s/([ ])\1 /g" in.txt > out.txt
```

Nach jeder Buchstabe wird ein leeres Zeichen finzugefügt. zB. KLARA -> K L A R A

```
sed -i "s/^/| /g" in.txt
```

Fügt eine | vor jeder Zeile und speichert das Ergebnis in der originalen Datei -> -i (in place)

Info über Daten

- **file** – Daten über Datei: z.B. Kodierung.
 - **file** [DATEI]
- **wc** – gibt die Zahl der Zeilen, Wörter und Zeichen
 - **wc** [DATEI]
 - **wc -l** [DATEI]
 - **wc -w** [DATEI]
 - **wc -m** [DATEI]

19

File

Informationen über Datei erfahren wie zB: Datei-empty, Kodierung (UTF-8), usw.

```
file pack.tar.gz
```

Output:

```
pack.tar.gz: gzip compressed data, from Unix, last modified: Tue May 26 12:50:05 2008
```

Wc

In dem Kommando ,wc' die Option -l steht für Zeilen, -w für Wörter und -m für Zeichen.

```
wc -w Demo/*.txt
```

Gibt die Wörteranzahl jeder txt-Datei aus dem Verzeichnis Demo und auch die Gesamtanzahl der Wörter.

Suchkommandos

- **find** – sucht nach Datei
 - **find** [DATEI]
 - **find** *.txt
- **grep** – sucht in Datei
 - **grep** = **g**lobal **r**egular **e**xpression **p**rint
 - **grep** [SUCHMUSTER] [DATEI]
 - **grep -v** "DE" adressen.txt
 - **Egrep** - Extended Grep, **grep**, **Fgrep** - Fixed Grep

20

Find
find . temp.txt
Sucht in aktuellem Verzeichnis und in untergeordneten Verzeichnissen die Datei temp.txt

find Demo/t*.txt
Gibt die txt-Dateien die mit 't' anfangen und in Verzeichnis Demo sind.

Grep
ist ein Unix tool, verwendbar auch in Linux. Grep durchsucht die angegebenen Dateien (oder die Standardeingabe) nach einem regulären Ausdruck (Muster) und gibt die entsprechenden Zeilen aus.

Extended Grep (erweitertes grep) wird verwendet wenn mehrere reguläre Ausdrücke ausgeführt werden müssen. Es muss die Option -E eingesetzt werden und die regulären Ausdrücke mit | (oder) verknüpfen.

grep -E -n Hans*|*221 adressen.txt
Gibt die Zeile wo das Muster gefunden ist. Der Zeilennummer wird mit ausgegeben.

Fixed Grep sucht nur nach festen Zeichenketten, womit sich die Suche vor allem in großen Dateien erheblich beschleunigen lässt.

grep -F -n Hansi adressen.txt
Gibt die Zeilen aus die 'Hansi' enthält.

grep -v Hans* adressen.txt
Gibt die Zeilen aus wo das Muster: 'Wörtern die mit Hans anfangen' nicht zutrifft (-v).

Die Ausgabe "grep" herausfiltern:
klara@klara-laptop:~/Desktop/Demo\$ ps ax | grep firefox | grep -v grep
10338 ? Sl 0:01 /usr/lib/firefox-3.0b5/firefox

Einpacken und Entpacken

■ Einpacken

- **tar** cvfz [ArchivName].tgz [Verzeichnis] [Datei]
- **bzip2** [Datei]
- **zip** [ArchivName].zip [Verzeichnis] [Datei]

■ Entpacken

- **tar** -xvfz [ArchivName].tgz
- **bunzip2** [Datei].bz2
- **unzip** -l [ArchivName].zip

21

1. Obwohl tar ein Programm zur Bandsicherung ist, wird es häufig verwendet, um Archivdateien zu erzeugen und verarbeiten.

```
tar cvfz archivName.tgz Verzeichnis1 Datei1.txt Datei2.txt Verzeichnis2
```

Das Kommando sichert alle Daten in ein komprimiertes Archiv. Wenn Sie ein komprimiertes Archiv (-z Option) erzeugen wollen, sollten Sie als Dateiendung .tgz oder .tar.gz benutzen, um dies anzuzeigen. Der -f Option in Verbindung mit einem Archivnamen ist erforderlich, wenn Sie in eine Datei sichern, oder Dateien aus einer Datei extrahieren möchten.

```
tar xfvz archivName.tgz
```

Das Kommando entpackt das Archiv.

```
tar -tvf archivName.tgz
```

Mit der Option -v werden detailliertere Informationen über die im Archiv enthaltenen Dateien angezeigt.

2.

```
gzip -dc [DATEI].tar.gz | tar xfv -
```

Das Entpacken einer *.tar.gz oder *.tgz Datei mit dem gzip-Kommando

Gzip ist eine Methode für Komprimierung und Dekomprimierung der Dateien im Lempel-Ziv Verfahren. Der erste Befehl dekomprimiert die Datei und schickt das Ergebnis auf die sogenannte "Pipe". Das tar-Kommando greift sich von dieser "Pipe" die Daten und entpackt diese. Die "Pipe" kann man sich wie eine Zwischenablage vorstellen.

Siehe nächste Notizen-Folie für ‚Einpacken und Auspacken‘ ->

Inhalt

- Terminal – Shell – Bash
- Kommandos
- Arten des Kommandos
 - Verzeichnis/Dateienverwaltung
 - Datenverwaltung
 - Prozessverwaltung
 - Andere Kommandos
- Fazit
- Quellen

22

Notizen ‚Einpacken und Auspacken‘ ->

2. Das Archiv mit der Endung "bz2" verhält sich ähnlich wie ein Archiv mit einer "gz"-Endung. Der Vorteil von "bz2" sind die bedeutend kleineren Dateien. Beim Linux-Kernel oder ISO-Images sind es einige MB.

`bzip2 [DATEI]`

Das Kommando komprimiert eine Datei. Das Ergebnis ist eine Datei namens "[DATEI].bz2".

`bunzip2 [DATEI].bz2`

Das Dekomprimieren. Das Ergebnis ist wieder der ursprüngliche Dateiname.

3. ZIP-Dateien sind hauptsächlich unter DOS und Windows.

`zip demos.zip Demo1 b.txt Demo2`

Das Kommando komprimiert die Verzeichnisse Demo1 und Demo2 und die txt-Datei in demos.zip

`unzip -l demos.zip`

Der Inhalt des demos.zip-Archivs ausgeben lassen

`unzip demos.zip`

Man kann demos.zip-Archiv auch wieder entpackt werden

Prozessverwaltung

- **ps** – gibt eine Liste mit aktuellen Prozesse
 - **ps -A | less**
 - **ps -aux**
- **top** – Auflisten der Prozesse nach CPU-Last
 - **top**
- **kill** – Prozess beenden
 - **kill <prozess_id/prozessname>**

23

ps

Gibt kurze Informationen über laufenden Prozesse aus.

ps -A|less

Die Verwendung des Kommandos less ermöglicht es die Ausgabe seitenweise anzuzeigen. Per Leertaste kann man das Ergebnis ansehen und mit ,b' geht es eine Seite zurück.

top

top -d 1

zeigt laufende Prozesse mit 1 s Update.

kill

Mit ,kill' werden die Prozesse beenden die außerhalb der Terminal laufen.

Die Kommandos die in Terminal laufen werden mit CTRL+c beenden.

ps -A | grep <prozessname>

Um ProzessID eines Prozesses zu ermitteln

kill <ProzessID>

Beenden eines Prozesses

Inhalt

- Shell
- Kommandos
- Arten des Kommandos
 - Verzeichnis/Dateienverwaltung
 - Datenverwaltung
 - Prozessverwaltung
 - Andere Kommandos
- Fazit
- Quellen

Andere Kommandos

- **[Alt] + [F1 bis Fn]** – Terminal wechseln
- **Clear, CTRL-L** – Bildschirm löschen
- **Q, D, quit, exit** – „exit“ Kommandos
- **last, lastlog** – Letzte Logins anzeigen
- **logout, exit** – Beenden einer Sitzung
- **CTRL-C, kill** – Kommandoabbruch

25

Weitere nützliche Kommandos:

```
head -c6 /dev/random | uuencode -m - | sed -n '2s/=*$//;2p,
```

Das Kommando ermittelt zufällige Zeichenketten, die sich sehr gut als Passwörter eignen. Für längere Zeichenketten kann die Anzahl der aus dem Random Device ausgelesenen Zeichen vergrößert werden.

Inhalt

- Shell
- Kommandos
- Arten des Kommandos
 - Verzeichnis/Dateienverwaltung
 - Datenverwaltung
 - Prozessverwaltung
 - Andere Kommandos
- Fazit
- Quellen

Fazit

- Shell - mächtig und schneller gegenüber grafischen Benutzerschnittstellen
- Es gibt mehrere Arten von shell
- Shell interpretiert die Kommandozeile
- Die Kommandos sind casesensitive
- Man muss nicht unbedingt die Syntax der Kommandos merken → --help, man, info
- Man kann mehrere Kommandos kombinieren

- Zeitsparen: schneller Zugriff an der Daten, schnelles Ergebnis

Inhalt

- Shell
- Kommandos
- Arten des Kommandos
 - Verzeichnis/Dateienverwaltung
 - Datenverwaltung
 - Prozessverwaltung
 - Andere Kommandos
- Fazit
- Quellen

Quellen

- <http://fibel.org/linux/lfo-0.6.0/lfo.html>
- <http://doc.ubuntu-fr.org/terminal>
- <http://wiki.ubuntu-forum.de/index.php>
- <http://proteino.de/index.php/4102017/>
- <http://www.raw.to/rubriken/linux-faq/linux-faq3.htm>
- http://www-lehre.inf.uos.de/~ainf/2006/dokumentation/SelfLinux-0.12.1/html/was_ist_shell01.html
- <http://www.linux-user.de/ausgabe/2007/07/094-zubefehl/index.html>