

Linux

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwillhelm@hs-harz.de
- <http://www.miwilhelm.de>
- Raum 2.202
- Tel. 03943 / 659 338

Inhalt

1. Einführung, Literatur, Begriffe
2. Installation
3. Konsolen-Befehle, Shell, Manual
- 4. Komplexe Befehle (grep, find, tar, sed, cron, netcat)**
5. C-Programmierung (C, IO, Thread, make)
6. Python (Sprache)
7. Python (GUI)

a1:	abcd Abcd abccd abcccd abcabc abcabcabc	a6:	havefunwertvollwer hallo (hallo)a ((((O)))) (((((((abc(((((((acht
a2:	editor \$y=\$x	a7:	12345 56789 abcdef 234.56
a3:	abeditor \$x=3 \$x =42 ((abcd = 3 + a)) ((\tabcd\t= 3 + a))	a8:	12.11.10 1.1.1 01.01.2010 11.11.abc 0y. 127.123.168.02 127.2.168.002
a4:	12bfe.abcd ^12bFe 15bFe	a9:	^hallowelt hier ein wer-wolf hier noch ein werwerwolf hier ein weiterer wolf 0a.alpha 1z.beta
a5:	13f abcd 1214 y=2*(5*(x+z)) 12345\$		

Parameter:

- i ignore case
- c Anzahl, count
- n line number
- w Ausdruck als Wort
- v Negation
- s Unterdrückt Fehlermeldungen
- d Directory Optionen
 - d recurse oder -r
 - d skip
 - d read

Beispiele

```
grep 'ab' *
grep '^ab'
grep '^abc'
grep 'abcd$'
grep 'cd$'
grep 'l[23]'
grep 'l[23]f'
grep 'l[23][bf]'
grep '^l[23]'
grep '^[ae]'
grep -i '[Ae]'
grep '^^[ae],', a1
grep 'b.e'
grep '\.' '. ' liefert alles

grep -E 'abc+'
grep -E 'abc?'
grep -E '(ab|ed)'

egrep '[:alnum:]'
egrep '[:alphanum:]'
egrep '[:alpha: ]'
egrep '[:digit:]'
grep '([()]*a)'

egrep '([[:digit:]]{1,3}\.){3}[[:digit:]]{1,3}'
```

-E Extended Regular expression oder egrep

Beispiele

```
egrep '$x'
egrep '\$x'      korrekt, aber nun mit Space
egrep '\$x '      als Wort, tab
egrep '<$x>'      als Wort, tab, space, auch am Anfang und am Ende
```

Datum suchen:

```
egrep '<[0-9]{1,}\.[0-9]{1,}\.[0-9]{1,4}>' *
a8:12.11.10
a8:1.1.1
a8:01.01.2010
egrep '<[0-9]{1,}\.[0-9]{1,}\.[0-9]{2,4}>' *
a8:12.11.10
a8:01.01.2010
```

Beispiel zur Berechnung der Anzahl der Treffer

```
$ grep -c "a" a?
```

```
a1:3
```

```
a2:0
```

```
a3:3
```

```
a4:1
```

```
a5:1
```

```
a6:3
```

```
a7:0
```

```
a8:1
```

```
a9:1
```

Das Programm find: Suchen rekursiv

Das Kommando „find“ sucht alle Dateinamen, die bestimmten Bedingungen genügen.

Aufruf: `find [directory] bedingung aktion`

Parameter:

type -f Datei

 -d Verzeichnis

directory Beginn der Suche im Teilbaum

bedingung Es können folgende Bedingungen eingestellt werden:

 -name muster Ist erfüllt, wenn der Name dem Muster entspricht

 -atime name Ist erfüllt, wenn auf die Datei vor zahl Tagen zugegriffen wurde

 -mtime zahl Ist erfüllt, wenn die Datei vor zahl Tagen verändert wurde.

 -newer datei Ist erfüllt, wenn die untersuchte Datei nach der letzten Änderung von Datei geändert wurde.

 -size zahl Ist erfüllt, wenn die Datei zahl viele Blöcke hat.

Das Programm find: Suchen rekursiv

Das Kommando „find“ sucht alle Dateinamen, die bestimmten Bedingungen genügen.

Aufruf: `find [directory] bedingung aktion`

aktion Bestimmt die Aktion

-exec kommando;

Auf jede gefundene Datei wird kommando ausgeführt. An derjenigen Stelle, des Kommandos, an der Dateiname steht, müssen die Klammern {} angegeben werden.

-ok kommando;

Auf jede gefundene Datei wird kommando ausgeführt. An derjenigen Stelle, des Kommandos, an der Dateiname steht, müssen die Klammern {} angegeben werden. Mit Abfrage des Benutzer (y oder yes)

-print

Ausgabe jeder gefundenen Datei.

Beispiele:

`find /home/paul -type f -name *.html`

Sucht alle Dateien mit der Endung html

`find /home/otto ! [-user otto] -exec ls -l {} \;`

Bewirkt die Anzeige aller Dateien, die nicht zu Otto gehören.

`find /home/otto -atime +10 -print`

Bewirkt die Anzeige aller Dateien , auf die in den letzten 10 Tagen nicht zugegriffen wurde.

`find . [-size +10 -o mtime +7] -ls -l {} \;`

Bewirkt die Anzeige aller Dateien, die eine Größe von mehr als 10 Blöcken haben oder in den letzten 7 Tagen nicht verändert wurden.

Beispiele:

`find -name a.out`

- Sucht alle Dateien mit dem Namen a.out, Programmieren mit g++

• `find -name a.out -o -name "*.o"`

- Sucht alle Dateien und Verzeichnisse mit dem Namen a.out oder mit *.o

• `find ./ -type f -name "a?"`

- Sucht alle Dateien mit der Maske a?

• `find -maxdepth 2 -type f -name "a? "`

Sucht alle Dateien mit der Maske a?, Maximal aber ein Unterverzeichnis

Zippen mit tar

- Archive mit Inhalt von /etc und /home erstellen:

- `tar cvf test.tar /etc/ /home/`
- `tar cvf - /etc /home | gzip > test.tar.gz`
- `tar cvf - /etc /home | bzip2 > test.tar.bz2`

- Mit GNU tar ist folgende Kurzform möglich, um gepackte Archive zu erstellen:

- `tar czvf test.tar.gz /etc/ /home/`
- `tar cjvf test.tar.bz2 /etc/ /home/`

Zippen mit tar

- Archive entpacken:
 - `tar xvf test.tar`
 - `gunzip < test.tar.gz | tar xvf -`
 - `bunzip2 < test.tar.bz2 | tar xvf -`
 - `zcat test.tar.gz | tar xvf -`
- Mit GNU tar ist folgende Kurzform möglich:
 - `tar xzvf test.tar.gz`
 - `tar xjvf test.tar.bz2`

Zippen mit tar

- Archivinhalt ansehen:
 - `tar tvf test.tar`
 - `gunzip < test.tar.gz | tar tf -`
 - `bunzip2 < test.tar.bz2 | tar tf -`
 - `zcat test.tar.gz | tar tf -`
- Mit GNU tar ist folgende Kurzform möglich:
 - `tar tzvf test.tar.gz`
 - `tar tjvf test.tar.bz2`

Zippen mit tar

- Archive von einem Verzeichnis packen und über 'pipes' in ein anderes Verzeichnis entpacken
 - `tar cvf - 'directory' | (cd 'directory'/; tar xpvf -)`
- **Beispiele:**
 - `tar -C sourcedir cvf - . | tar -C targetdir xpvf -`
 - und dies via SSH übers Netzwerk
 - `ssh user@host "tar cvf - /ursprung1/ /ursprung2/" | (cd /ziel/; tar xpvf -)`

Streameditor sed

- Oft müssen Dateien umgeformt werden.
 - Beispiel: Ändere alle Vorkommen von "html2 und "htm" innerhalb einer Datei. Dafür ist sed geeignet.
- Der Aufruf von sed hat folgende Form:
 - `sed [-n] [-e script] [-f scrfile] file1 file2`
 - Die Dateien file1, file2 usw. werden nacheinander behandelt. und im Ergebnis der Arbeit von sed in die Standardausgabe stdout transportiert.

Streameditor sed: Optionen

- -n
 - Die Option -n unterdrückt die Ausgabe, nur besondere Kommandos geben aus
- -e
 - Bei der Option -e steht Skript für eine Folge von Editorkommandos, eines je Zeile. Mehrere Kommandos müssen durch Apostroph eingefasst werden.
- -f
 - Die Option -f besagt, dass eine Folge von Editorkommandos der Datei name entnommen werden soll.

Streameditor sed: Beispiele

- Beispiel für die Kurzform (vgl. -e):
 - sed '/^\$/d' file
- Erläuterungen:
 - Das Kommando beseitigt alle Leerzeilen aus der Datei file und schreibt das Resultat auf die Standardausgabe:
 - ^ Anfang der Zeile, siehe grep
 - \$ Ende der Zeile, siehe grep
 - /^\$/ aktuelle Zeile die von Anfang bis Ende nichts enthält
 - d lösche aktuelle Zeile

Streameditor sed: Optionen

- `s/reg/new/flags`
 - Ersetze die Zeichenkette aus `reg` durch `new`
- `y/string1/string2`
 - Transformiere alle Zeichen aus `string1` in Zeichen aus `string2`, `string1` und `string2` sind gleich lang
- `a\ text`
 - Anhängen von Text. Das Argument besteht aus einer oder mehreren Zeilen. Alle Zeilen außer der letzten enden mit einem Backslash \
- `i\ text`
 - Einfügen von Text
- `d`
 - lösche den Vergleichsspeicher, Fortsetzung

Streameditor sed: Optionen

- `D`
 - Lösche erste Zeile im Vergleichsspeicher
- `n`
 - kopiere den Vergleichsspeicher nach stdout
- `N`
 - anhängen der nächste nEingabezeile an den Vergleichsspeicher
- `p`
 - Drucke nach stdout
- `P`
 - drucke die erste Zeile des Vergleichsspeichers
- `l`
 - wie `p` aber zeige Sonderzeichen

Streameditor sed: Optionen

- r file
 - liest die Datei file und gib sie nach stdout
- w file
 - Anhängen des Vergleichsspeichers nach Date file
- q
 - beenden
- !
 - Anwenden der Funktion auf Zeilen die nicht adressiert wurden
- =
 - Ausgabe der Zeilennummer

Automatisches Starten mit cron

- Mit dem cron-daemon kann man Programme zeitgesteuert starten.
- crontab ist eine **Tabelle**, in der die einzelnen cronjobs definiert sind.
- crontab ist ein **Programm**, das die Tabelle verwaltet.

■ Dateien

- /var/spool/cron/
- /etc/cron.d

■ Befehle

- more crontab
- crontab -l listet alle „user“-Jobs auf
- crontab -u user1 -l listet alle Jobs des Benutzers user1 auf
- crontab -r löscht alle Jobs
- crontab -e Starten des Programms

■ Alternativen

- at
- anacron
- fcron

Crontab Syntax

■ Format

- W M T Std Min Befehl
- W: 0-7, Sonntag 0st 0 oder 7
- Ein * bedeutet, dass die Ausführung immer erfolgt.
- */n bedeutet, dass die Ausführung nach n Zyklen erfolgt.
- n,m,o bedeutet, dass die Ausführung nach n,m,o Einheit erfolgt.

■ Beispiele

- 0 5 * * * /home/user1/backup.sh
- Jede Nacht um 5 Uhr morgens
- */10 * * * * /home/user1/play_helene_fischer.sh
- Alle zehn Minuten Horror
- 0 8,17 * * * /home/user1/play_helene_fischer.sh
- Nur um 8 und um 17 Uhr Horror
- 59 23 31 12 * /home/user1/play_helene_fischer.sh
- Sylvester Horror: 31.12.XXXX 23:59

Crontab Ausgabe

■ Ausgabe

- Die Ausgabe der Cronjobs wird standardmässig per Mail an den jeweiligen System-User der den Cronjob eingerichtet hat gesendet. Um dies zu unterdrücken, könnte man die Ausgabe in eine Datei umleiten oder mit Umleitung zu /dev/null komplett verwerfen:.

■ Beispiele

- 0 8,17 * * * /home/user1/play_helene_fischer.sh >> /var/log/cron/payinglist 2>&1
- Ausgabe in einen logfile umleiten
- 0 8,17 * * * /home/user1/play_helene_fischer.sh > /dev/null 2>&1
- Ausgabe in den Papierkorb umleiten

Automatisches Starten mit anacron S.509

- Startet asynchron.
- Verwendet man cron sollte der Rechner ständig an sein.
- Anacron startet die Jobs perodisch. Meist nach dem Starten.
- Man sollte den Befehl „nice“ verwenden (geringere Priorität)
 - nice -20 /home/user1/backup hohe Priorität
 - nice 19 /home/user1/backup niedrige Priorität
- **Speichert die Dateien in**
 - /etc/anacrontab
- **Optionen**
 - -s Serialize. Jobs starten nacheinander.
 - -n now, startet immer den Job, achtet nicht auf die Zeit (mit -s)
 - -f force, startet immer den Job, achtet nicht auf die Zeit
 - -d Geht nicht in den Hintergrund.
 - -q Fehlermeldungen werden unterdrückt (mit -d).
 - -u Setzt den neuen Zeitstempel den anacron-Jobs. Sonst nichts
 - -v -V Ausgabe der Version
 - -h Ausdruckt von kurzen Meldungen.

Automatisches Starten mit systemd-timer

- Mit dem cron-daemon kann man Programme zeitgesteuert starten.
- crontab ist eine **Tabelle**, in der die einzelnen cronjobs definiert sind.
- crontab ist ein **Programm**, das die Tabelle verwaltet.
- **Dateien**
 - /var/spool/cron/
 - /etc/cron.d
- **Befehle**
 - more crontab
 - crontab -l listet alle „user“-Jobs auf
 - crontab -u user1 -l listet alle Jobs des Benutzers user1 auf
 - crontab -r löscht alle Jobs
 - crontab -e Starten des Programms
- **Alternativen**
 - at
 - anacron
 - fcron

Netzwerkverwaltung: Konfiguration

- Ubuntu: System->Administration->Networking.
- RedHat: grafisch und per shell
- Suse: YAST oder YAST2
- Mandrake: Wizard (Mandrake's Control Center)
- Mandriva: Wizard
- Gnome: gnome-network-preferences.
- KDE: knetworkconf.
- Allgemein: sbin/ifconfig

Netzwerkverwaltung: Dateien

- `etc/resolv.conf`
 - The `/etc/resolv.conf` file configures access to a DNS server, see Section 10.3.7. This file contains your domain name and the name server(s) to contact:
- `/etc/nsswitch.conf`
 - The `/etc/nsswitch.conf` file defines the order in which to contact different name services. For Internet use, it is important that dns shows up in the "hosts" line:
 - `grep hosts /etc/nsswitch.conf`
 - `hosts: files dns`
 - This instructs your computer to look up hostnames and IP addresses first in the `/etc/hosts` file, and to contact the DNS server if a given host does not occur in the local hosts file.

Netzwerkverwaltung: Befehle

- `ip addr show`
 - Anzeige der IP-Adressen
- `ip route show`
 - Anzeige der Routing-Informationen
- `ifconfig`
 - Haupttool
- `netstat`
 - Überprüfen der Konfiguration
- `ping`
 - Testen des Kontaktes zu anderen Rechnern
- `traceroute`
 - Anzeige der Routing-Informationen