Computer & GNU/Linux Einführung
Teil 2

Simon Haller, Sebastian Stabinger

Inst. für Informatik
[IFI]

September 28, 2015

Rechte I

Rechte

- Userrechte
- Grouprechte
- Rechte für alle (world, other)

Rechte II

Anzeigen von Rechten

ls -l Dateiname

Die Rechte:

```
- rw- r-- r-- 1 c102159 c102 4387 Jan 08 13:38 Dateiname
- --- --- --- | ------- ---- ---- ------------ ---------
|  |   |   |   |   |     |      |     |          |          |
|  |   |   |   |   |     |      |     |          |          Dateiname
|  |   |   |   |   |     |      |     |          Änderungsdatum
|  |   |   |   |   |     |      |     Dateigröße
|  |   |   |   |   |     |      Gruppe
|  |   |   |   |   |     Eigentümer
|  |   |   |   |   Hardlinks
|  |   |   |    Rechte aller anderen
|  |   |  Rechte der Gruppe
|  Rechte des Eigentümers (r...Read, w...Write, x...eXecute)
Dateiart (-...Datei, d...Directory, l...Link)
```

?

Rechte III

Setzen von Rechte mit chmod

$ chmod *ugo* = ±*rwx* [DATEI]

u... User, *g*... Group und *o*... Other

Alternativ: $ chmod [RECHTE] [DATEI]

Beispiel:

```
$ chmod 650 [DATEI]
        |||  rwx
        |||- 000
        ||-- 101
        |--- 110
```

Setzen des Eigentümers mit chown

- $ chown USER.GROUP [DATEI]

Rechte IV

AFS Rechte

Die Rechte können mit:

`fs listacl <dirname>`

abgefragt werden.

Achtung:

Die AFS Rechte werden nicht mit chmod geändert und sind nicht mit `ls -l` sichtbar. Falls AFS ACLs geändert werden sollen, bitte `http://www.uibk.ac.at/zid/systeme/linux/lpccs3/afs_krb_benutzerdoku.html` sorgfältig lesen.

Zeichensätze

Die Umgebungsvariablen LC_ALL und LANG

Der Zeichensatz kann mit

- LANG=C7-Bit ASCII d.h. keine Umlaute, kein ß
- LANG=en_US8-Bit ISO Latin 1 (ANSI) Englisch
- LANG=de_DE oder de_AT Deutsch
- LANG=de_AT.UTF8 Unicode UTF-8

festgelegt werden.

Designunterschied UNIX/Linux zu Windows

Im Unterschied zu Windows ist die Kodierung nicht Eigenschaft eines Objektes (Datei, Dateisystem etc.). In Windows ist der Zeichensatz durch 2-4 Bytes im Header der Datei kodiert.

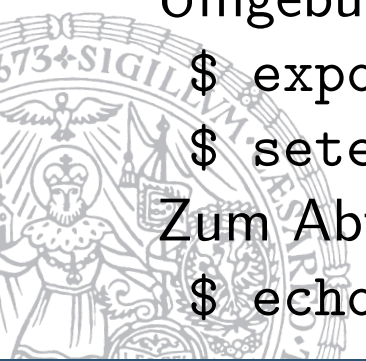
Variablen I

Wichtige Umgebungsvariablen

- *PATH* Legt den Suchpfad für ausführbare Dateien fest
- *HOME* Homeverzeichnis des Users
- *USER* Aktuell angemeldeter User
- *DISPLAY* Legt die IP-Adresse für den X-Server fest
- *SHELL* Legt die Standard-Shell fest
- *EDITOR* Legt den Standard-Editor fest

Setzen und ausgeben von Umgebungsvariablen

Umgebungsvariablen werden mit dem Befehl
`$ export UMGEBUNGSVARIABLE=Wert` (je nach Shell auch mit:
`$ setenv UMGEBUNGSVARIABLE=Wert`) gesetzt.
Zum Abfragen reicht:
`$ echo $UMGEBUNGSVARIABLE`



Variablen II

Variablen löschen

der Inhalt kann mit

- `$ set Variable`
- `$ unset Variable1 Variable2`

gelöscht werden.

Mit `$ env` koennen alle gesetzten Variablen aufgelistet werden.

Anstatt der Umgebungsvariable $HOME kann das $\sim$ Zeichen in Kommandos verwendet werden.

Welche Befehle werden wirklich verwendet

Welcher Befehl wird ausgeführt

Da die PATH Umgebungsvariable u.U. ziemlich lang und unübersichtlich wird ist es oft praktisch zu wissen welcher Befehl wirklich ausgeführt wird.

Mit dem Befehl: `$ type Befehl`

kann der verwendete Befehl ausgegeben werden.

AFS und Kerberos

Kommandos

```
kinit [USER]
klist
kdestroy
aklog
```

Bash–Konfiguration

Konfigurationsdateien

- .bashrc
- .bash_profile
- .bash_logout

Die Datei .bash_history speichert die zuletzt verwendeten Kommandos.

History

die Geschichte der Geschichte

Das Kommando `history` ermöglicht es, bereits einmal eingegebene Kommandos erneut aufzurufen. Mit `history` lassen sich die letzten Befehle auflisten. Mit

!!	kann der letzte
!*n*	der n-ten Befehl
!*string*	der letzte Befehl der mit 'string' anfängt
!?*string*	letzter Befehl der 'string' im Befehlstext enthält

ausgeführt werden. Mit *strg* + *r* kann die History rückwärts durchsucht werden.

Bash–Shortcuts

Kommandozeile

strg + *a*	springt an den Anfang der Zeile
strg + *e*	an das Ende der Zeile
strg + *h*	löscht ein Zeichen
strg + *w*	löscht ein Wort
strg + *u*	löscht eine Zeile
strg + *c*	bricht eine Kommandoausführung ab.
strg + *d*	Dateiende
strg + *z*	Unterbricht ein Kommando

Bash-Shortcuts

Praktisches

tab `vervollständigt einen Befehl/Argument bzw. zweimal gedrückt wird eine mögliche Auswahl angezeigt.`

alt + . `liefert das letzte Argument des zuletzt ausgeführten Befehls.`

Kommandozeile wiederherstellen

Nach einem Programmabbruch reagiert die Konsole nicht mehr oder nur noch chaotisch

strg + *j* `reset` *strg* + *j* oder
`stty sane` oder
`reset`

stellen die Standardeinstellungen wieder her.

Wichtige Kommandos

echo *(Anzeigen einer Textzeile)*

Optionen:

- -n keine neue Zeile
- -e aktiviert \ Optionen:
  - \ n neue Zeile
  - \ t horizontaler Tabulator
  - \ v vertikaler Tabulator

Hello World

Hello World in einer Shell kann folgendermaßen realisiert werden:

```
$ echo Hello World
```

Wichtige Kommandos

ls [OPTIONEN] [DATEI] *(list)*

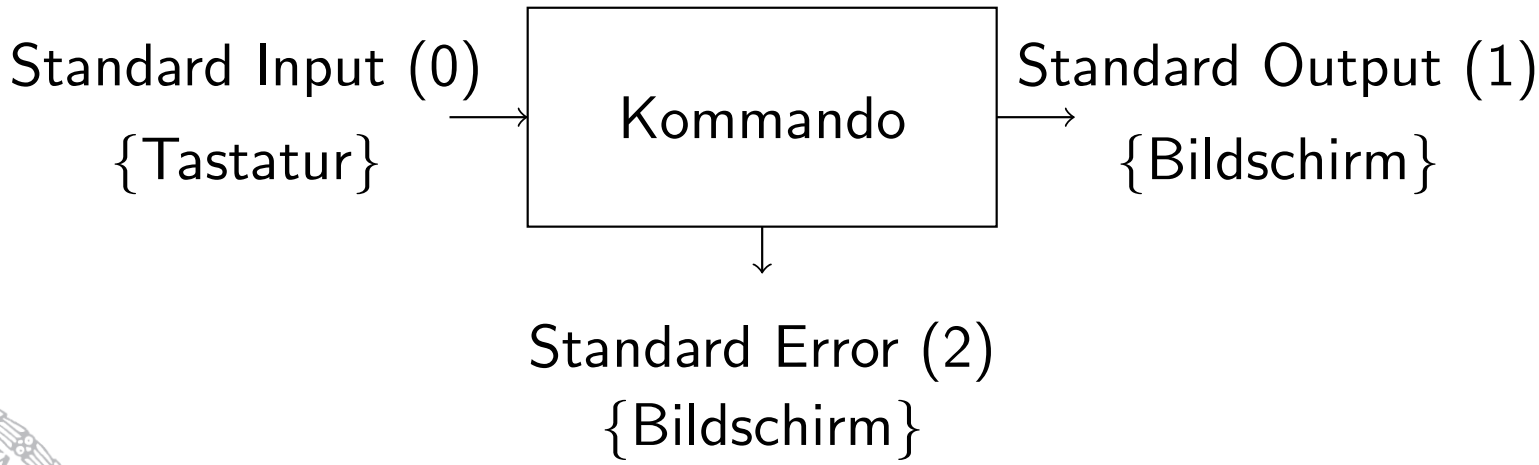
Optionen:

- -a all . . . zeige versteckte Dateien (.Dateiname)
- -d directory . . . listet Verzeichnisseinträge anstatt Inhalt
- -h human readable
- -l long . . . gibt zusätzlich Attribute aus
- -r reverse . . . umgekehrte Reihenfolge
- -R rekursiv . . . rekursive Ausgabe von Unterverzeichnissen
- -t Sortieren nach Änderungszeit (letzte Modifikation)
- -u Sortieren nach letzter Zugriffszeit

Elementares

Standardein- und Standardausgabe

Prinzipiell gibt es für die Kommandos drei elementare Files: die Standardeingabe (stdin), die Standardausgabe (stdout) und die Standardfehlermeldungen (stderr).



Ein- und Ausgabeumleitung

Ausgabeumleitung: > *und* >>

Mit “>” kann die Standardausgabe in eine Datei umgeleitet werden, wobei diese Datei in jedem Fall neu erstellt wird. Mit “>>” wird die Ausgabe von STDOUT an eine Datei angehängt. Beispiele:

```
ls >~/verzeichnis_inhalt.out
ls -lh Datei >>~/verzeichnis_inhalt.out
```

Die Standardfehlerausgabe (STDERR) kann mit 2 > umgeleitet werden.

```
ls >~/verzeichnis_inhalt.out 2 >~/fehler.log
```

Ein- und Ausgabeumleitung

Ausgabeumleitung: > *und* >>

Die unendlichen Weiten von /dev/null:

```
ls -lh Datei >>~/verzeichnis_inhalt.out 2 >/dev/null
```

eine Umleitung nach /dev/null bewirkt, dass die Ausgabe verschmissen wird.

Es ist auch möglich, die Standardfehlerausgabe in die Standardausgabe umzuleiten:

```
ls Datei >~/verzeichnis_inhalt.out 2 > &1
```

Ein- und Ausgabeumleitung

Eingabeumleitung: <

Ein Beispiel:

```
$ more < verzeichnis_inhalt.out
```

Beispiel

`$ cat a b > a` FALSCH!

der Inhalt der Datei *a* wird gelöscht.

Richtig: `$ cat b >> a`

hängt den Inhalt von *b* an den von *a* an.

Wichtige Kommandos

mv [DATEI] [DATEI] *(move)*

verschieben von Dateien

ln [OPTIONEN] [DATEI] [LINKNAME] *(link)*

verlinken von Dateien
Optionen:

-s symbolic . . . Symbolischer Link

Beim Hard-Link wird eine fiktive Kopie der Datei angelegt, d.h. wenn die originale Datei gelöscht wird existiert noch immer eine Datei mit vollem Inhalt.

Wichtige Kommandos

cp [OPTIONEN] [DATEI] *(copy)*

kopieren von Dateien

Optionen:

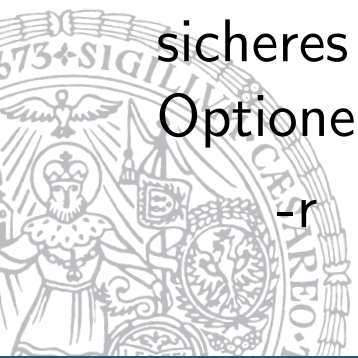
- -l link, Dateien werden nur verlinkt
- -p preserve, Owner und Timestamps werden beibehalten
- -r recursiv
- -v verbose, gibt nach Stdout den wirklich ausgeführten Befehl aus

scp [OPTIONEN] [DATEI] [USER]@[HOST:][DATEI] *(secure copy)*

sicheres kopieren von Dateien zu anderen Rechnern

Optionen:

- -r recursiv



Wichtige Kommandos

rm [OPTIONEN] [DATEI] *(remove)*

entfernen von Dateien Optionen:

- -f force
- -r recursiv
- -i interactive, (frägt bei jeder Datei nach)

Wichtige Kommandos

cat [OPTIONEN] [DATEI] *(concatenate)*

gibt Inhalt der Dateien an die Standardausgabe (stdout) aus.
Falls der Dateiname weggelassen wird, liest es aus der Standareingabe (stdin)
Optionen:

- -n number, füge Zeilenummer hinzu
- -s squeeze–blank, entferne mehrfache Leerzeilen

References I

Redinger, M. (2004). Linux einführung.
http://www.uibk.ac.at/zid/software/unix/linux/linux.pdf.