

Shells als Zugang zu Linux

Eine shell stellt den wichtigsten Zugang zu Linux dar. Über shell commands schicken wir Befehle an das Linuxsystem oder erhalten Antworten.

- Terminal Programme

`mate-terminal`

- Mate Anwendungen → Systemwerkzeuge → MATE-Terminal
- Mate Anwendungen → Systemwerkzeuge → LXTerminal

- Editieren von Dateien

`lxterminal`

- emacs <filename> <https://wiki.archlinux.de/title/Emacs> Erklärungen im Programm Eingabe in eine existierende Konsole

- vi <filename>

- i / a in den Edit Mode / hinter einem Wort editieren
- cw curser auf ein Wort positionieren und ein Wort ändern
- yy Kopieren der Zeile
- dd / ndd Löschen der Zeile / Löschen von n Zeilen
- p paste
- Esc Taste verlassen des edit modes
- :q! quit ohne zu sichern
- :wq! write and quit

vi summary: <http://www.linfo.org/vi/summary.html>

Nützliche Linux Kommandos

- Hilfe unter Linux

```
man <cmd>  
info
```

Beschreibung zu Kommandos anzeigen
Linux Hilfe System anzeigen

- Verzeichnisse und Files

```
drwxr-xr-x  
u g o
```

Rechte <dir> or <file> :
d = directory / r = read / w = write / x = execute
u = user / g = group / o = other

.

Aktuelles Directory

..

Directory oberhalb des aktuellen Directories

```
chmod u+x/u+r/u+w <file>
```

Beispiele: Ändern der Filerechte
(execute/read/write) für user, analog für other
oder group

- Verzeichnisbaum

```
ls -l -a -t <dir>
```

Listen aller Files im Directory <dir>
a für all auch . Files, t für zeitsortiert

```
drwxr-xr-x 2 marks users 4096 Oct 4 13:35 code  
-rw-r--r-- 1 marks users 3025217 Oct 3 13:20 einleitung.odp
```

```
cd <dir>
```

in das Directory <dir> wechseln

```
mkdir <dir>
```

Erzeugen des Directories <dir>

```
pwd
```

Verzeichnisbaum des aktuellen Directories

Nützliche Linux Kommandos

• Files

<code>cat <file></code>	Fileinhalt auf der shell ausgeben
<code>echo "bla bla" > <file></code>	String in <file> kopieren
<code>echo "more .." >> <file></code>	String an <file> anhängen
<code>mv <fileA> <fileB></code>	File oder Directory von <fileA> nach <fileB> umbenennen
<code>less <file></code>	Fileinhalt ansehen mit Blättern
<code>vi <file></code>	Editieren und erzeugen eines Files mit vi
<code>.myConfigFile</code>	Files und Directories mit Punkt im Namensbeginn sind nicht sichtbar
<code>ls -a <dir></code>	Zeigt auch Files mit Punkt im Namensbeginn
<code>chmod a+x <file></code>	Files für alle user ausführbar machen
<code>ls -l <dir></code>	Zeigt Files mit Rechten, Zeit, user and group

• Suchen im Verzeichnisbaum oder im File

<code>find <dir> -name <filename></code>	Suchen eines Files vom Directory <dir>
<code>grep pattern <file></code>	Ausdruck pattern in <file> finden
<code>locate pattern</code>	Files mit string pattern im Namen finden

`locate` benötigt eine Datenbank mit allen File Namen, die durchsucht wird. Das Programm `updatedb` erzeugt die DB und muss als root user gestartet werden.

Nützliche Linux Kommandos

- Löschen von Files und Directories

Vorsicht mit dem Löschen: Files und Directory werden instantan entfernt und sind weg oder nur aus dem backup zurück zu holen.

<code>rm <file></code>	Löschen des Files <file> im aktuellen directory
<code>rm -rf <dir></code>	Rekursives löschen aller Files und der darunterliegenden Directory Struktur von <dir> (vorsicht)
<code>rm -rf *</code>	Rekursives löschen aller Files und der darunterliegenden Directory Struktur vom aktuellen directory.
<code>rmdir <dir></code>	Löscht das leere Directory <dir>

- Verwendung von Pipes

Das Pipe Zeichen | unter Linux bedeutet, das der Output vom links stehenden Command als Input für das rechts stehende Command benutzt wird.

<code>commandA commandB</code>	Pipe
<code>cat File.txt grep -i "Pet"</code>	Sucht nach pattern Pet in File.txt

Nützliche Linux Komandos

• Weitere Commands

- `~` ist ein Synonym für das home directory des eingelogten Benutzers (`cd ~/myDir`).
- `*` wirkt als Wildcard und kann in File und Directory Namen eingesetzt werden.
- `;` Mehrere commands können in einer Zeile hintereinander ausgeführt werden, wenn Sie durch `;` getrennt sind.
- `top` zeigt eine Liste mit allen Prozessen, die im System laufen.
- `history` erzeugt eine Liste der eingegebenen letzten 1000 Befehle.
- Mit den **Pfeil Up/Down Tasten** lässt sich in den bereits eingegebenen Komandos blättern.
- Mit **/pattern** kann in `vi` und `less` nach dem string pattern gesucht werden.
- **!g** führt das letzte mit **g** beginnende Komando aus.
- **Tabulator Taste** ergänzt eine begonnene Eingabe
- **Strg + c** bricht ein Komando ab, das gerade ausgeführt wird.
- **Komando &** das Komando wird im Hintergrund ausgeführt.
- **Strg + z bg <return>** ein Komando, das gerade ausgeführt wird, wird in den Hintergrund schickt.
- **ps** zeigt Prozesse, die in der shell ausgeführt werden
- **ps -aux** zeigt Prozessliste
- **wc -l <file>** zählt die Anzahl der lines in einem File <file>
- **kill -9 <process id>** entfernen des Prozesses mit der id <process id>

Arbeitsvorschläge:

- Erzeugen Sie ein Directory "c++Einfuehrung". Wechseln Sie in das Directory und erzeugen Sie "folien". Laden Sie `LinuxCommands.pdf` und `einleitung.pdf` von unserer Kurswebpage und speichern die Files in "folien".

Wo befinden sich die heruntergeladenen Files zunächst?

Gibt es eine Option für `grep` mit der Groß- und Kleinschreibung ignoriert wird?

- Probieren Sie beide Editoren `emacs` und `vi`. Schicken Sie `emacs myFile.txt` in den Hintergrund.

- `lsmod` listet die geladenen Kernelmodule. Bestimmen Sie die Anzahl der aktiven Kernel module.

- Öffnen Sie das oben editierte File mit `less`. Verwenden Sie `ps` und `top` um die process number zu finden. Suchen Sie nach der process number und beenden Sie den Prozess.

- Das Kommando `uname -r` zeigt Ihnen die Kernel Version, die Sie verwenden. In dem directory `/lib/modules/..kernel../` sind alle Kernelmodule in weiteren Directories zu finden. Die Module haben die Fileendung `*.ko`. Wie können wir die Anzahl aller Module herausfinden?

Linux Shell Scripts - Einleitung

- **Einleitung**

Beim Arbeiten am und mit einem Linux System treten wiederholt gleiche und komplizierte Befehlsketten auf. Shell Skripte sind ein Weg Administrations- und Analyseprozesse zu automatisieren.

- Mögliche Skript Sprachen: bash, csh, perl, python,
 - (auch in Kombination mit kompilierten C/C++ Prozessen)
 - **Shell Script** ist ein ausführbares Textfile in dem in der ersten Zeile mit der Sequenz `#!/Pfad/zum/Interpreter: #!/bin/bash` (shebang) die Sprache festgelegt wird.
 - Alle Zeichen im Skript, die nach einem `#` stehen werden ignoriert.
 - Ausführen von einem Skript
- ```
chmod u+x myScript.sh
./myScript.sh
```

- **Ein Beispiel**

```
#!/bin/bash
ein kleiner test
echo Hello world!
echo "so stellen wir" \<" Sonderzeichen dar "*" \<$Maehh
Ausdruck hinter \< wird woertlich genommen
exit 0
```

# Linux Umgebungsvariable

- Environment Variable

Das Systemverhalten wird unter anderem durch **environment variables** bestimmt. Diese werden systemweit gesetzt und werden im allgemeinen mit grossen Buchstaben geschrieben. Sie sind aber auch für jeden Benutzer anpassbar. Im File `.bashrc` in Ihrem home directory wird diese Anpassung vorgenommen.

|                               |                                                                                  |
|-------------------------------|----------------------------------------------------------------------------------|
| <code>printenv</code>         | ansehen der augenblicklich gesetzten Variablen                                   |
| <code>HOST</code>             | Name Ihres Rechners                                                              |
| <code>USER</code>             | Name des aktuellen users                                                         |
| <code>PATH</code>             | Pfade die nach ausführbaren Programmen durchsucht werden, jeweils mit : getrennt |
| <code>LD_LIBRARY_PATH</code>  | Pfade, in denen nach shared libraries gesucht wird                               |
| <code>PRINTER</code>          | Name des default Printers                                                        |
| <code>MATHEMATICA_HOME</code> | Anwendungsspezifische Pfade, mathematica Pfad                                    |

Nach Änderungen in `.bashrc` File muß das environment neu gesetzt werden:

```
>$ source $HOME/.bashrc
```



# Linux Umgebungsvariable

- Weitere nützliche Variable

Die Befehlseingaben werden im File `.bashr_history` im home directory gespeichert. Um einen Zeitstempel im history command hinzuzufügen verwenden wir folgenden Eintrag in der `.bashrc` oder `.profile`

```
export HISTTIMEFORMAT="%d/%m/%y %T "
```

- Setzen von Befehlssynonymen

Das Setzen von Abkürzungen für Befehle wird im File `.alias` in Ihrem home directory vorgenommen. Das `.alias` File wird von der Datei `.bashrc` ausgeführt. Einige Beispiele:

```
alias ll='ls -lat' Ausführliche zeitsortierte Liste des aktuellen <dir>
alias ..='cd ..' Mit .. ein directory aufsteigen
alias cd..='cd ..' Fix Tippfehler für das Wechseln ins höhere <dir>
alias .='echo $PWD' Zeigt den Pfad des aktuellen <dir>
```

Nach Änderungen im `.alias` File muß das environment neu gesetzt werden:

```
>$ source $HOME/.bashrc oder source $HOME/.alias
```

## Anpassung der Files `.bashrc` und `.alias`

### Arbeitsvorschlag:

- Nehmen Sie die gerade diskutierten Modifikationen im `.alias` File vor.
- `history` soll von nun an das Datum mit ausgeben. Implementieren Sie die Modifikationen.
- Schauen Sie sich die `PATH` Variable an und fügen Sie einen `“.`“ in den Pfad ein. Was ist die Wirkung?

# Linux Installation

- Linux kann mehr oder weniger einfach auf Ihrem PC/Laptop installiert werden.

Kursmaterial eines Kurses zur Installation und der Anwendung von Linux finden Sie unter [https://www.physi.uni-heidelberg.de/~marks/linux\\_einfuehrung/](https://www.physi.uni-heidelberg.de/~marks/linux_einfuehrung/)

- Installation von einem Linux System in einer virtuellen Maschine, die unter Windows arbeitet.

Präsentation von Elias Grimm und Yannik Hess

<https://www.physi.uni-heidelberg.de/~egrimm/virtualbox.php>