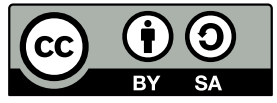
Arbeiten mit der Shell Teil 1

Linux-Kurs der Unix-AG

Benjamin Eberle

13. November 2014



UNIX AG
TU Kaiserslautern

RH Regionales Hochschul-
Rechenzentrum Kaiserslautern RK

Shell: Standard-Features

- Prompt (häufig: `benutzer@rechner:~$`) zeigt an, dass die Shell auf Befehle wartet
- Befehl eingeben, mit Enter ausführen
- Befehle bestehen aus einem Programmnamen (z. B. `ls`: Dateien auflisten) und (manchmal optionalen) Parametern
- Parameter sind entweder Optionen (z. B. `-a`: auch versteckte Dateien anzeigen) oder Argumente (z. B. `/home/linux-kurs`: Dateien in `/home/linux-kurs` anzeigen)
- $\rightarrow$ `ls -a /home/linux-kurs` zeigt alle Dateien in `/home/linux-kurs` an

Absolute und relative Pfade

- Datei- und Verzeichnisnamen können auf zwei Arten angegeben werden:
- absolut: ganzer Pfad vom Wurzelverzeichnis aus
- Beispiel: `/home/linux-kurs/meine_bilder/tux.png`
- relativ: Pfad relativ zum aktuellen Verzeichnis
- Beispiel: `meine_bilder/tux.png`

Spezielle Verzeichnisse

- `.` (aktuelles Verzeichnis)
- `..` (Elternverzeichnis des aktuellen Verzeichnisses)
- `~` (Homeverzeichnis)
- Kombination mit relativen Pfaden:
- `~/meine_bilder/tux.png`
- `../anderer_ordner/notizen.txt`

Die Unix-Philosophie

- Unix-Philosophie: viele kleine Programme, die jeweils eine Aufgabe gut lösen
- wichtiges Unix-Paradigma: Alles ist eine Datei
- keine Registry, MMC, Systemsteuerung, Laufwerksbuchstaben

Allgemeines zu Befehlen

- „Alles ist eine Datei“: die wichtigsten Befehle dienen der Dateiverwaltung
- Argumente sind häufig Dateien oder Verzeichnisse
- Verhalten von Befehlen hängt häufig vom aktuellen Verzeichnis ab (relative Pfade, `ls` ohne Argumente)
- `pwd` gibt das aktuelle Verzeichnis (Arbeitsverzeichnis) aus, wird standardmäßig auch im Prompt angezeigt
- Groß- und Kleinschreibung ist wichtig, sowohl bei Befehlen als auch bei Dateinamen

ls

- `ls` („list“): Verzeichnisinhalt auflisten
- wichtige Optionen:
  - `-a`: auch versteckte Dateien anzeigen (Dateiname beginnt mit einem `.`)
  - `-R`: Inhalt von Unterverzeichnissen rekursiv auflisten
  - `-l`: weitere Informationen ausgeben: Dateirechte, Besitzer, Größe, Änderungsdatum
  - `-h`: Dateigröße in möglichst große Einheiten umrechnen (sonst immer Byte)
  - weitere, für die Prüfung wichtige, Optionen: Buch, Kapitel 6.2.2
- Argumente:
  - optional ein oder mehrere Verzeichnisse, sonst wird der Inhalt des aktuellen Verzeichnisses angezeigt

cd

- `cd` („change directory“): in ein anderes Verzeichnis wechseln
- keine wichtigen Optionen
- Argument: Verzeichnis, in das gewechselt werden soll
  - relativer oder absoluter Pfad
  - `~`, `.` und `..` können benutzt werden
  - `cd -` wechselt in das vorherige Verzeichnis
  - wird kein Argument angegeben, wechselt `cd` ins Home (entspricht `cd ~`)

mkdir/rmdir

- `mkdir` („make directory“): legt Verzeichnisse an
- `rmdir` („remove directory“): löscht leere Verzeichnisse
- wichtige Option: `-p` legt Elternverzeichnisse automatisch an/löscht diese, falls leer
- Argumente: Verzeichnisse, die angelegt/gelöscht werden sollen

rm

- `rm` („remove“): löscht Dateien und Verzeichnisse
- wichtige Optionen:
  - `-r` oder `-R`: löscht Verzeichnisse mit Inhalt rekursiv
  - `-f`: überhaupt keine Fragen stellen (z. B. bei schreibgeschützten Dateien)
  - `-i`: vor dem Löschen jeder einzelnen Datei nachfragen
- Argumente: beliebig viele Dateien und/oder Verzeichnisse
- Achtung: `rm` löscht Dateien unwiederbringlich ohne nachzufragen!
- `rm -rf` ist besonders gefährlich

Dateien anlegen

- normalerweise werden Dateien angelegt, wenn man etwas darin speichert
- zu Testzwecken kann auch `touch` verwendet werden
- `touch` setzt eigentlich das Änderungsdatum der Datei auf das aktuelle Datum
- nicht existente Dateien werden leer angelegt

mv

- `mv` („move“) verschiebt Dateien und Verzeichnisse
- wichtige Optionen:
  - `-b`: legt Sicherungskopien an, wenn Dateien überschrieben werden
  - `-i`: fragt vor dem Überschreiben nach
- Argumente:
  - zwei Dateiennamen: Datei wird umbenannt
  - beliebig viele Dateien/Verzeichnisse und Verzeichnis als letztes Argument: Dateien werden in das Verzeichnis verschoben

cp

- `cp` („copy“) kopiert Dateien und Verzeichnisse
- wichtige Optionen:
  - `-b`: legt Sicherungskopien an, wenn Dateien überschrieben werden
  - `-i`: fragt vor dem Überschreiben nach
  - `-r` oder `-R`: kopiert Verzeichnisse rekursiv
  - `-u`: kopiert nur, wenn die Ausgangsdatei neuer als das Ziel ist bzw. dieses noch nicht existiert
  - `-a`: behält Rechte und Änderungszeiten beim Kopieren bei, als `root` ausgeführt auch Dateieigentümer und Gruppe
- Argumente:
  - zwei Dateinamen: Kopie der Datei wird angelegt
  - beliebig viele Dateien/Verzeichnisse und Verzeichnis als letztes Argument: Dateien werden in das Verzeichnis kopiert

nano

- `nano` ist ein primitiver und vergleichsweise einfach zu bedienender Texteditor für die Kommandozeile
- optionales Argument: ein Dateiname (sonst muss dieser beim Speichern ausgewählt werden)
- wenn die Datei nicht existiert, wird sie angelegt, sonst geladen
- Navigieren im Text mit den Pfeiltasten
- Speichern mit `Strg-O`, Beenden mit `Strg-X` (`^` entspricht `Strg`)

Spickzettel

Alle Befehle

Befehl	Optionen
`pwd`	
`ls`	`-a`, `-R`, `-l`, `-h`
`cd`	
`mkdir`	`-p`
`rmdir`	`-p`
`rm`	`-r`/`-R`, `-f`
`touch`	
`mv`	`-b`, `-i`
`cp`	`-b`, `-i`, `-r`/`-R`
`nano`	