

Vorsemesterkurs Informatik

Sommersemester 2011

Einführung in die Bedienung von Unix-Systemen

SoSe 2011

Übersicht

- 1 Unix, Linux, Shells
- 2 Shell-Kommandos
- 3 Dateien und Verzeichnisse


Unix / Linux

- Mehrbenutzer-Betriebssystem
- ursprünglich 1969 in den Bell Labs entwickelt
- Viele moderne Betriebssysteme basieren auf Unix
- Bekanntes Beispiel: GNU Linux

Terminals / Shells

- Terminal = Schnittstelle zwischen Mensch und Maschine
- Textbasierte Terminals: Interaktion mittels Kommandos über die Tastatur
- Graphische Terminals, Graphische Benutzeroberflächen: Fenster, Interaktion über Tastatur, Maus, . . .
- Auf den Rechner der RBI u.a.: Linux Benutzeroberflächen: Gnome und KDE

Login

- Login = Anmelden des Benutzers am System
- Benutzername + Passwort
- danach hat man die Kontrolle in einer **Shell**
- oder kann eine solche starten
- Am sog. **prompt** kann man Kommandos eingeben
- Kommando eingeben, danach  („return“) betätigen

Shell-Kommandos

Kommandozeilenparameter

Beispiel: `ls -a dinge`

`ls` `-a` `dinge`

Kommando **Option** **Argument**

- **Kommando**: der eigentliche Befehl (ein Programm), im Bsp. `ls` für (list)
- **Optionen**: werden meist durch `-` oder `--` eingeleitet, verändern die Ausführung des Befehls
- **Argumente**: Dateien, Verzeichnisse, Texte auf die das Kommando angewendet wird.

Einige Kommandos

- `echo` *Text*
gibt *Text* aus
- `whoami` und `hostname`
gibt Benutzername bzw. Rechnernamen zurück
- `pwd` (print working directory)
gibt das aktuelle Arbeitsverzeichnis aus
- `mkdir` *Verzeichnis* (make directory)
erzeugt *Verzeichnis*
- `cd` *Verzeichnis* (change directory)
wechselt in *Verzeichnis*
- `cd ..`
wechselt ein Verzeichnis nach oben
- `ls` (list)
Anzeigen des Verzeichnisinhalts
- `man` *Kommando* (manual)
Man page zum *Kommando* anzeigen.

Beispiele

```
> echo "Hallo Welt!" ↵  
Hallo Welt!  
> pwd ↵  
/usr/usersb1/w97/sabel/vorkurs  
> whoami ↵  
sabel  
> hostname ↵  
diomedes
```


Beispiele

```
> echo "Hallo Welt!"  
Hallo Welt!  
> pwd  
/usr/usersb1/w97/sabel/vorkurs  
> whoami  
sabel  
> hostname  
diomedes
```

```
> mkdir dinge  
> ls  
dinge  
> cd dinge  
> ls  
> ls -a  
. .  
> cd ..  
> mkdir .versteckt  
> ls  
dinge  
> ls -a  
. .. dinge .versteckt
```

Dateien und Verzeichnisse

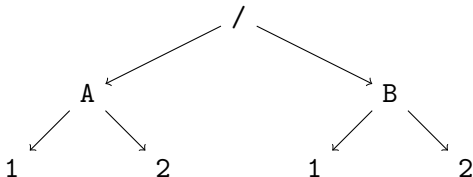
„everything is a file“

- Jedes Unix-System verwaltet einen **Dateibaum**
- Dateibaum: virtuelles Gebilde zur Datenverwaltung
- Bausteine sind dabei **Dateien** (file)
- **Datei** enthält Daten: Text, Bilder, Maschinenprogramme, . . .
- Spezielle Dateien: **Verzeichnisse** (*directories*), enthalten selbst wieder Dateien.
- Dateien haben **Namen**
- Jede Datei befindet sich in einem Verzeichnis, dem übergeordneten Verzeichnis
- **Wurzelverzeichnis** / (root directory) ist in sich selbst enthalten.

Ein Verzeichnisbaum

Beispiel:

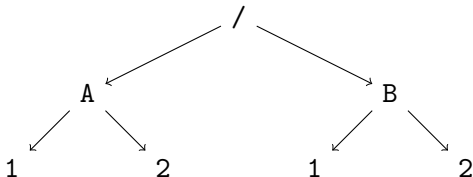
- Wurzelverzeichnis / enthält zwei Verzeichnisse A und B.
- A und B enthalten je ein Verzeichnis mit dem Namen 1 und 2.




Ein Verzeichnisbaum

Beispiel:

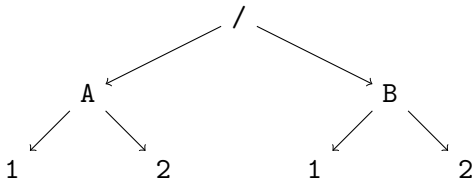
- Wurzelverzeichnis / enthält zwei Verzeichnisse A und B.
- A und B enthalten je ein Verzeichnis mit dem Namen 1 und 2.



```
> tree   
/  
+-- A  
|   +-- 1  
|   +-- 2  
+-- B  
    +-- 1  
    +-- 2
```

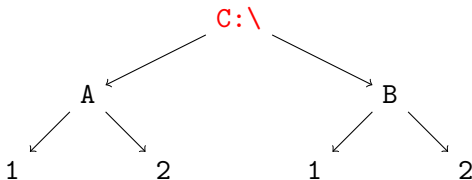
Relative und absolute Pfade (1)

- **Absoluter Pfad** einer Datei oder eines Verzeichnisses:
Pfad von der Wurzel beginnend, Verzeichnisse getrennt mit / (slash)
- z.B. /A/1 und /B/1.



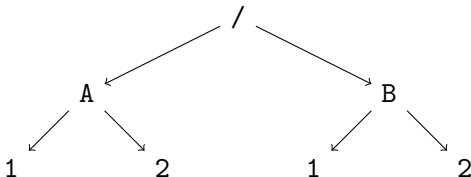
Relative und absolute Pfade (1)

- **Absoluter Pfad** einer Datei oder eines Verzeichnisses:
Pfad von der Wurzel beginnend, Verzeichnisse getrennt mit / (slash)
- z.B. /A/1 und /B/1.
- Unter **Windows**: Wurzelverzeichnis ist Laufwerk, und Backslash \ statt / z.B. C:\A\1.



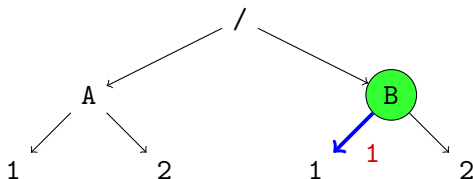
Relative und absolute Pfade (2)

- **Relative Pfade:** Pfad vom aktuellen Verzeichnis aus, beginnen **nicht** mit /.



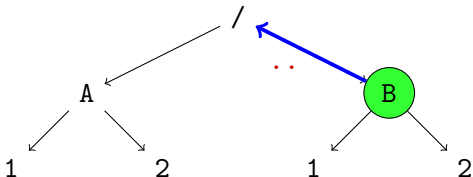
Relative und absolute Pfade (2)

- **Relative Pfade:** Pfad vom aktuellen Verzeichnis aus, beginnen **nicht** mit /.
- z.B. man ist in /B: Dann bezeichnet 1 das Verzeichnis /B/1.



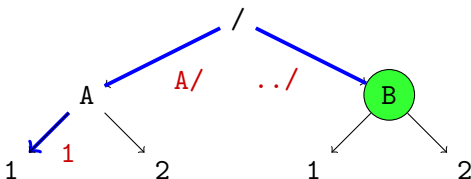
Relative und absolute Pfade (2)

- **Relative Pfade:** Pfad vom aktuellen Verzeichnis aus, beginnen **nicht** mit /.
- z.B. man ist in /B: Dann bezeichnet 1 das Verzeichnis /B/1.
- .. ist das **übergeordnete** Verzeichnis
- z.B. man ist in /B: Dann bezeichnet .. das Wurzelverzeichnis und ../A/1 bezeichnet /A/1



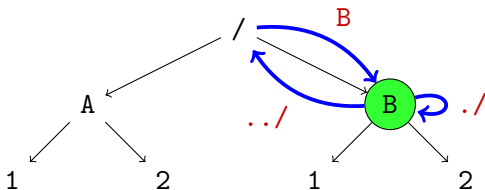
Relative und absolute Pfade (2)

- **Relative Pfade:** Pfad vom aktuellen Verzeichnis aus, beginnen **nicht** mit /.
- z.B. man ist in /B: Dann bezeichnet 1 das Verzeichnis /B/1.
- .. ist das **übergeordnete** Verzeichnis
- z.B. man ist in /B: Dann bezeichnet .. das Wurzelverzeichnis und ../A/1 bezeichnet /A/1



Relative und absolute Pfade (2)

- **Relative Pfade:** Pfad vom aktuellen Verzeichnis aus, beginnen **nicht** mit /.
- z.B. man ist in /B: Dann bezeichnet 1 das Verzeichnis /B/1.
- .. ist das **übergeordnete** Verzeichnis
- z.B. man ist in /B: Dann bezeichnet .. das Wurzelverzeichnis und ../A/1 bezeichnet /A/1
- . bezeichnet das **aktuelle** Verzeichnis, z.B. ./../B gleich zu ../B

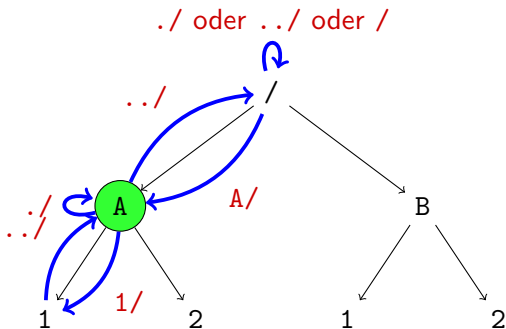


Relative und absolute Pfade (3)

In /A bezeichnen

- 1,
- ../../A/1,
- /../../../../A/1//,
- /A/../../A/1

alle das Verzeichnis /A/1.



Das Homeverzeichnis

- **Homeverzeichnis** = persönliches Verzeichnis des Benutzers
- Das eigene Homeverzeichnis findet man z.B. durch `echo ~`
- denn: `~`: Kürzel für das Homeverzeichnis
- und z.B. `~musterfrau` = Kürzel für das Homeverzeichnis von Musterfrau

Dateien editieren

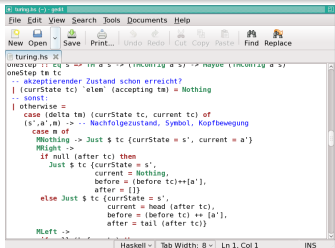
Texteditor: Programm zum Erstellen und Verändern von Textdateien (insbesondere Programmen)

Graphische Editoren, z.B.

- **kate** (kate-editor.org/) KDE
- **gedit** (projects.gnome.org/gedit/) Gnome
- **Notepad++** (notepad-plus-plus.org/) für Windows
- **TextWrangler** (barebones.com/products/textwrangler/) für Mac OS X
- **Emacs** (www.gnu.org/software/emacs/)
- **XEmacs** (<http://www.xemacs.org/>)

Textmodus z.B. vi

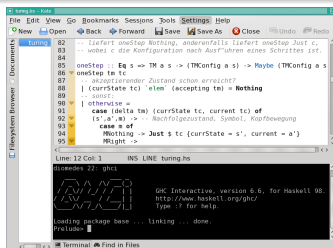
Einige Texteditoren



```

1  turing.hs |<
2  oneStep :: Eq s => (TM a s) -> (TMConfig a s) -> Maybe (TMConfig a s)
3  oneStep tm tc
4  -- akzeptierender Zustand schon erreicht?
5  | (currState tc) `elem` (accepting tm) = Nothing
6  -- sonst:
7  | otherwise =
8  | case (delta tm) (currState tc, current tc) of
9  (s',a',m) -> -- Nachfolgezustand, Symbol, Kopfbewegung
10 case m of
11 Nothing -> Just $ tc (currState = s', current = a')
12 MRight ->
13 | if null (after tc) then
14 Just $ tc (currState = s',
15           current = Nothing,
16           before = (before tc)++[a'],
17           after = [])
18 else Just $ tc (currState = s',
19               current = head (after tc),
20               before = (before tc) ++ [a'],
21               after = tail (after tc))
22 MLeft ->
  
```

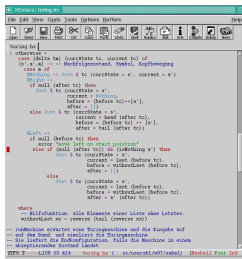
gedit (Linux, Gnome)



```

82 -- liefert oneStep Nothing, anderenfalls liefert oneStep Just c
83 -- wobei c die Konfiguration nach Ausfuehren eines Schrittes ist.
84
85 oneStep :: Eq s => (TM a s) -> Maybe (TMConfig a s)
86 oneStep tm tc
87 -- akzeptierender Zustand schon erreicht?
88 | (currState tc) `elem` (accepting tm) = Nothing
89 -- sonst:
90 | otherwise =
91 | case (delta tm) (currState tc, current tc) of
92 (s',a',m) -> -- Nachfolgezustand, Symbol, Kopfbewegung
93 case m of
94 Nothing -> Just $ tc (currState = s', current = a')
95 MRight ->
  
```

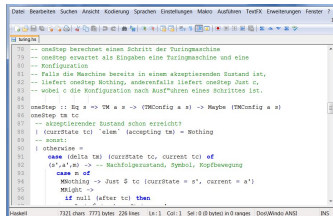
kate (Linux, KDE)



```

1  turing.hs |<
2  case (delta tm) (currState tc, current tc) of
3  (s',a',m) -> -- Nachfolgezustand, Symbol, Kopfbewegung
4  case m of
5  Nothing -> Just $ tc (currState = s', current = a')
6  MRight ->
7  | if null (after tc) then
8  Just $ tc (currState = s',
9            current = Nothing,
10           before = (before tc)++[a'],
11           after = [])
12 else Just $ tc (currState = s',
13               current = head (after tc),
14               before = (before tc) ++ [a'],
15               after = tail (after tc))
16 MLeft ->
  
```

xemacs (Linux)



```

78 -- oneStep berechnet einen Schritt der Turingmaschine
79 -- oneStep erwartet als Eingaben eine Turingmaschine und eine
80 -- Konfiguration
81 -- Falls die Maschine bereits in einem akzeptierenden Zustand ist,
82 -- liefert oneStep Nothing, anderenfalls liefert oneStep Just c,
83 -- wobei c die Konfiguration nach Ausfuehren eines Schrittes ist.
84
85 oneStep :: Eq a => TM a s -> (TMConfig a s) -> Maybe (TMConfig a s)
86 oneStep tm tc
87 -- akzeptierender Zustand schon erreicht?
88 | (currState tc) `elem` (accepting tm) = Nothing
89 -- sonst:
90 | otherwise =
91 | case (delta tm) (currState tc, current tc) of
92 (s',a',m) -> -- Nachfolgezustand, Symbol, Kopfbewegung
93 case m of
94 Nothing -> Just $ tc (currState = s', current = a')
95 MRight ->
96 | if null (after tc) then
  
```

Notepad++ (MS Windows)