# Programming Introduction

Lukas Keller

HS 2010 University Bern

## Contents

# 1 Terminal

## 1.1 Introduction

```
> uname -mns
  Darwin imac.local i386
  Report bugs to <bug-coreutils@gnu.org>.
> uname -mns
  Darwin mbkp.local i386
> ssh anker.unibe.ch
  user@bender.unibe.ch’s password:
> uname
  Linux
> uname -mon
  bender x86_64 GNU/Linux
> uname --help
  Usage: uname [OPTION]...
  Print certain system information. With no OPTION, same as -s.

    -a, --all print all information, in the following order,
                          except omit -p and -i if unknown:
    -s, --kernel-name print the kernel name
    -n, --nodename print the network node hostname
    -r, --kernel-release print the kernel release
    -v, --kernel-version print the kernel version
    -m, --machine print the machine hardware name
    -p, --processor print the processor type or "unknown"
    -i, --hardware-platform print the hardware platform or "unknown"
    -o, --operating-system print the operating system
        --help display this help and exit
        --version output version information and exit
```

## 1.2 Commands

`rm` removes a file or a directory

```
cami@bender:~/test$ ls
todelete.txt
cami@bender:~/test$ rm todelete.txt
cami@bender:~/test$ ls
```

`touch` updates the access and modification times of each FILE to the current time.

```
cami@bender:~/test$ ls -l
-rw-r--r-- 1 cami cami 0 2009-08-25 20:29 date.txt
cami@bender:~/test$ touch date.txt
```

```
cami@bender:~/test$ ls -l
-rw-r--r-- 1 cami cami 0 2009-08-25 20:30 date.txt
```

It can be very useful to create a new empty file on the fly:

```
~/test$ ls
~/test$ touch emptyfile.txt
~/test$ ls
emptyfile.txt
```

# 2 Documentation with Latex

## 2.1 Introduction

In this section we explain some LaTeX details and different formatting commands.

Whenever you need to lookup a certain symbol for LaTeX we suggest you to use the online recognition tool `detexify` at `http://detexify.kirelabs.org/`.

## 2.2 Common Commands

### 2.2.1 Sectioning

Depening on the documentclass given in the very beginning of this file there exist several sectioning levels:

1. `\section{NAME}`
2. `\subsection{NAME}`
3. `\subsubsection{NAME}`
4. `\paragraph{NAME}`

To enforce LaTeXto use a newline add a double slash `\\` at the end of a line.

### 2.2.2 Font size and style

| | |
|---|---|
| `\rm` | A normal text |
| `\sl` | *An italic text* |
| `\bf` | **A bold text** |
| `\tiny` | A tiny ext |
| `\scriptsize` | A very, very small text |
| `\footnotesize` | A very small text |
| `\small` | A small text |
| `\large` | A big text |
| `\Large` | A bigger text |
| `\LARGE` | An even bigger text |
| `\huge` | A huge text |
| `\Huge` | An enormous huge text |
| `\emph` | *An emphasized text* |
| `\underline` | An underlined text and here using the ulem-package |
| `\texttt` | `function goto(int a) ...` |
| `\uuline` | A double unterstrichener text using the ulem-package |
| `\uwave` | A wavy unterstrichener text using the ulem-package |
| `\sout` | ~~A crossed trough text using the ulem-package~~ |
| `\xout` | ~~A deleted text using the ulem-package~~ |

### 2.2.3 Notes

To create a footnote use the `\footnote{YOUR NOTE}` command[1].
If you want to put a remark at side of a page use `\marginpar`.

This is a note at the border of the page.

### 2.2.4 Lists

There exist several list types in LaTeX. You start a list by adding a `\being{LISTTYPE}` and end it with an `\end{LISTTYPE}`. A list item is added with a `\item` between the `begin` and `end`. `LISTTYPE` can be one of the following list:

- `enumerate`
- `itemize`
- `description` with `\item[topic]`

Note that you can nest lists if you want to.

1. e4
    a) e4 e5
    b) Lc4 d6
2. Lc4 d6

### 2.2.5 Math, LaTeX's real strengths

A much longer introduction, although still called a short math guide, is avaiable online at `ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf`.

**Inline Mode** Equations with numeration with `\begin{equation} FORMULA \end{equation}`:

$$E_{kin} = \frac{1}{2}mv^2 \tag{1}$$

Equations without numeration with `\begin{equation*} FORMULA \end{equation*}`:

$$E_{kin} = \frac{1}{2}mv^2$$

Shortcut using `\[ FORMULA \]`:

$$-\frac{\hbar}{2m}\Delta\Phi(\vec{r}) + V(\vec{r})\Phi(\vec{r}) = E\Phi(\vec{r})$$

Inline mode with `$ FORMULA $` displays as $\int_{\infty}^{\infty} |\psi(x)^2| \mathrm{dx} = 1$.

[1] . . . as you can see here.

**Parenthesis**

$$\Biggl(\biggl(\Bigl(\bigl((\,)\bigr)\Bigr)\biggr)\Biggr)$$

**Spaces**

| | | | |
|---|---|---|---|
| **Spaces** | Small spaces | `\_` | $y = x^2\_y' = 2x\_y'' = 2$ |
| | Middle sized spaces | `\quad` | $y = x^2 \quad y' = 2x \quad y'' = 2$ |
| | Big spaces | `\qquad` | $y = x^2 \qquad y' = 2x \qquad y'' = 2$ |

**Indices and Powers**

$$a_i, x^{n+1} \qquad a_{ij} + b_{ij} = p_{ij} \qquad \ldots\text{and nested} \qquad a_{x_ij} = n_{x^{2^b_n}}$$

**Fractions**

$$\frac{Zaehler}{Nenner} \qquad \frac{a}{b} + \frac{c}{b} = \frac{a+c}{b} \qquad \frac{\frac{a}{b}}{\frac{c}{d}} \qquad \frac{\binom{n+1}{k/2}}{5!}$$

In the simple math environment two FORMULAdifferent sized fractions can be used; the small fractions $\frac{1}{2}$ or the normal sized $\dfrac{1}{x}$.

**Roots**

$$\sqrt[\text{root depth}]{\text{root term}} \qquad \sqrt{x+y-z}, \sqrt[5]{4+x}$$

**Functions**

$$f : \mathbb{N} \to \mathbb{R} \qquad f : x \mapsto x^2$$

Mathematical functions are writtein explicitely written in normal text not math mode text:

$$\sin(x) = \sin(x) \textbf{ and not } sin(x)$$

**Varia**

$$\left(\sqrt{\frac{A^C}{B_y}} + \sum_{i=1}^{N} a_i\right)$$

$$A \xrightarrow{\lambda_a} B$$

$$\iint z\,dx\,dy \quad \textbf{not} \quad \int\int zdxdy$$

$$\iint z\,dx\,dy \quad \textbf{not} \quad \int\int zdxdy$$

$$\Leftarrow \Leftrightarrow \Longleftrightarrow \Rightarrow \_ \Uparrow \Updownarrow \Downarrow$$

$$\bigcap \cap \sum \int_0^{2\pi} \vec{a}\dot{a}\ddot{a}a''$$

## Matrices

$$\det A = \|a_{ik}\| = \begin{vmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{vmatrix} .$$

# 3 Ruby Programming

## 3.1 Befehle

```
ruby Dateiname #Ausfuehren von einer in Ruby geschriebenen Datei

irb #Starten von Ruby

puts "Hallo" #Ausgeben vom String "Hallo"

%w{"String"} #Zerlegt den String in einzelne Woerter

ar=["a","b","c"] #Array mit den Elementen a, b und c erstellen und als ar abspeichern

ar.size  #Laenge des Arrays/Anzahl Elemente des Arrays ar

ar.sort  #Sortiert die einzelnen Elemente des Arrays ar alphab./numm.

ar.collect { |word| word.size} #gibt die Laenge der einzelnen Elementen in eines Arrays aus

ar.sort_by { |word| word.size} #gibt die Elemente endspechend ihrer Laenge in eines Arrays s

ar.collect!{|x| x + "!"} #Fuegt jedem Element des Arrays ein ! hinzu; Output ist wieder ein

ar.reject!{|x| Bedingung} #Loescht alle Elemente des Arrays, welche die Bedingung erfuellen;

ar.join    #Fuegt die einzelnen Elemente des Arrays zu einem einzigen String zusammen

ar.join("-")   #Fuegt die einzelnen Elemente des Arrays zu einem einzigen Sting zusammen und

ar.reverse   #Gibt den Array in der umgekehrten Reihenfolge aus

"Hallo #{ar}"   #Variable in einem String

ar.each{|x| print x, "--"} #Gibt einen String(?) aus, in welchem jedes Element des Arrays mi

ar + ar    #"Dupliziert" die Elemente im Array

ar*2    #Verdoppelt die Elemente im Array

"String".to_a   #Wandelt den String String in ein Array um mit einem Element, welches dem ga

"Hallo".chomp("llo")  #Schneiden dem String "llo" ab und gibt dann in diesem Fall den String

help String   #Aufrufen der Hilfe Klasse String
```

```
puts ar[1]    #gibt 2. Element des Arrays aus
```

## 3.2 Bash commands in Ruby

```
`ls`     #Gibt die Dateien im aktuellen Ordner aus
```

## 3.3 Beispielcodes

Array ar loeschen und ausgeben:

```
#Array ar loeschen
ar.collect{|x| x.chomp(x)}
```

Create a function which does that(Chop all elements). The function should be called with ‘superSort(‘ls‘)

```
def superSort(st,ar)
       ar.collect{|x| x.chomp(st)} #NOCH
end
```

## 3.4 Ausgabe einer Liste

```
Code:

list = ["a","b","c","d"]

#Funktion Listenelemente einzeln ausgeben
def li_aus(l)
       l.each do |l|
              puts "#{l}"
       end
end

#Liste ausgeben
li_ausgeben(list)

Output:
a
b
c
d
```

# 4 Terminal basic commands

## 4.1 man

- Name: format and diplay on-line manual pages
- Befehl (Beispiel):

  ```
  man man
  ```

## 4.2 cd

- Get the manpage for cd:

  ```
  man cd
  ```

- Help for cd:

  ```
  cd -help

  > Usage: cd [-plvn][-|<dir>]
  ```

- Go one dir up then come back here:

  ```
  cd ..
  cd "Pfad"
  ```

Frage: wie wieder zurueck? Ausser mit dem vorherigen Pfad eingeben

- Cd to your home directory:

  ```
  cd
  ```

## 4.3 ls

- List all files in this directory:

  ```
  ls

  > README.md
  ```

- Directly list all the files in the parent directory

  ```
  ls ..

  >01_man 06_touch 11_ssh 16_wc README.md
  >02_cd 07_mv 12_version_control_system 17_du
  >03_ls 08_cp 13_grep 18_find
  >04_pwd 09_rm 14_less 19_wget
  >05_mkdir 10_cat 15_sort 20_good_to_now
  ```

- List all files in this directory with additional information:

```
ls -l

> total 4
> -rw-r--r-- 1 ei16 studi 210 2010-09-05 19:37 README.md
```

- List all files recursively from the parent directory

Frage: Stimmt der Befehl?

```
ls -R

> .:
> README.md
```

### 4.4 pwd

- Check your current working dir:

```
pwd

> /home/ei16/Documents/pi/03_terminal_basic_commands/04_pwd
```

- Cd to the parent dir and check it there:

```
cd ..
pwd

> /home/ei16/Documents/pi/03_terminal_basic_commands
```

- Cd to the folder 03 ls symlink, a symbolic link, and check it again:

```
pwd

> /home/ei16/Documents/pi/03_terminal_basic_commands/03_ls
```

- Do an ‘ls‘, do you know where you are?

```
ls

>README.md
```

Jetzt bin ich im Ordner 03 ls mittels Link in diesen Ordner.

- Go one dir up cd .., and check again where your are.

```
cd ..
pwd

>/home/ei16/Documents/pi/03_terminal_basic_command
```

## 4.5 mkdir

- Create a new directory named ‘dir‘:

  ```
  mkdir dir
  ```

- Create in one command the directories ‘.foo/bar/‘

  ```
  mkdir foo; cd foo; mkdir bar
  ```

## 4.6 touch

- Create a new file name ‘bla‘ using ‘touch‘:

  ```
  touch bla
  ```

- Change the access and modification time of ‘bla‘ to yesterday:

  ```
  touch bla -d 2010-09-14

  ls -l
  >-rw-r--r-- 1 ei16 studi 0 2010-09-14 00:00 bla
  >-rw-r--r-- 1 ei16 studi 111 2010-09-05 19:37 README.md
  ```

## 4.7 mv

- Create two directories named ‘a‘ and ‘b‘:

  ```
  mkdir a b
  ```

- Move ‘a‘ into ‘b‘:

  ```
  mv a b
  ```

- Move ‘a‘ back:

  ```
  mv b/a .
  ```

- Rename ‘a‘ to ‘c‘:

  ```
  mv a c
  ```

- Remame ‘c‘ to ‘b‘ and directly overwrite it:

  ```
  mv c b -f
  ```

## 4.8 cp

- Create a file named ‘a‘ and write your favourite singer’s name in it :

  ```
  touch a
  vim a
  ```

- Copy the ‘a‘ to a file named ‘b‘ and check that both files have the same contents:

  ```
  cp a b
  ```

- Copy the directory ‘source‘ to ‘source copy‘ including all subfolders:

  Wie?

  ```
  ---
  ```

## 4.9 rm

- Delete ‘a‘:

  ```
  rm a
  ```

- Delete the folder ‘source‘ recursively with a single command:

  ```
  rm -r -f source
  ```

- Create two files ‘a‘, ‘b‘ and remove both in one command:

  ```
  touch a b
  rm -f a b
  ```

## 4.10 cat

- ‘cat‘ the ‘blink‘ file without arguments then use ‘vim‘ or ‘less‘ to look at the same file. Any difference?

  ‘Blink‘ mit ‘less‘ oder ‘vim‘ oeffnen, dann gibs keinen gruenen Hintergrund, also wird nicht ausgegeben.

- ‘cat‘ this ‘README.md‘ file showing the line numbers:

  ```
  cat -n README.md #Nummeriert jede Zeile
  ```

- ‘cat‘ this ‘README.md‘ file showing the line numbers even on empty lines

  ```
  cat -s README.md #Nummeriert keine leeren Zeilen
  ```

- `Just run ‘cat‘ directly without any argument. Can you imagine what happens by using the man page or the internet?`

  Man page: With no FILE, or when FILE is -, read standard input

- `Create two files ‘a‘ and ‘b‘ with a small text, then run ‘cat a b‘, what happens?`

  Es werden beide Dateien ausgegeben.

### 4.11 ssh

NOCH

### 4.12 version control system

http://github.com

username: wprog
URL: http://github.com/wprog
Anleitung:

```
git add NeueDatei
git commit -a
git push
```