Graphiken – Teil 2

LaTeX-Kurs der Unix-AG

Steffen Wolf

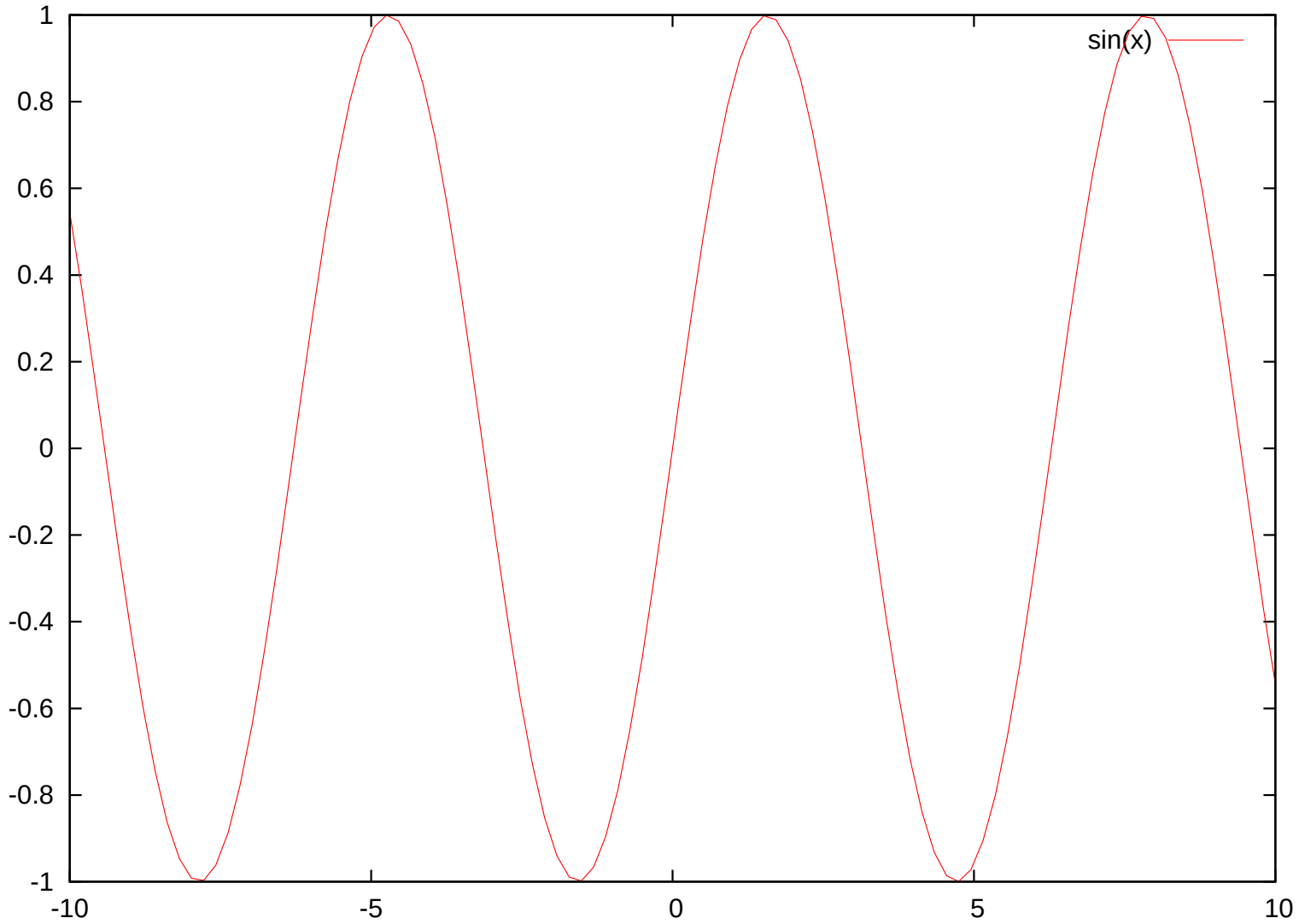
21. Juni 2006

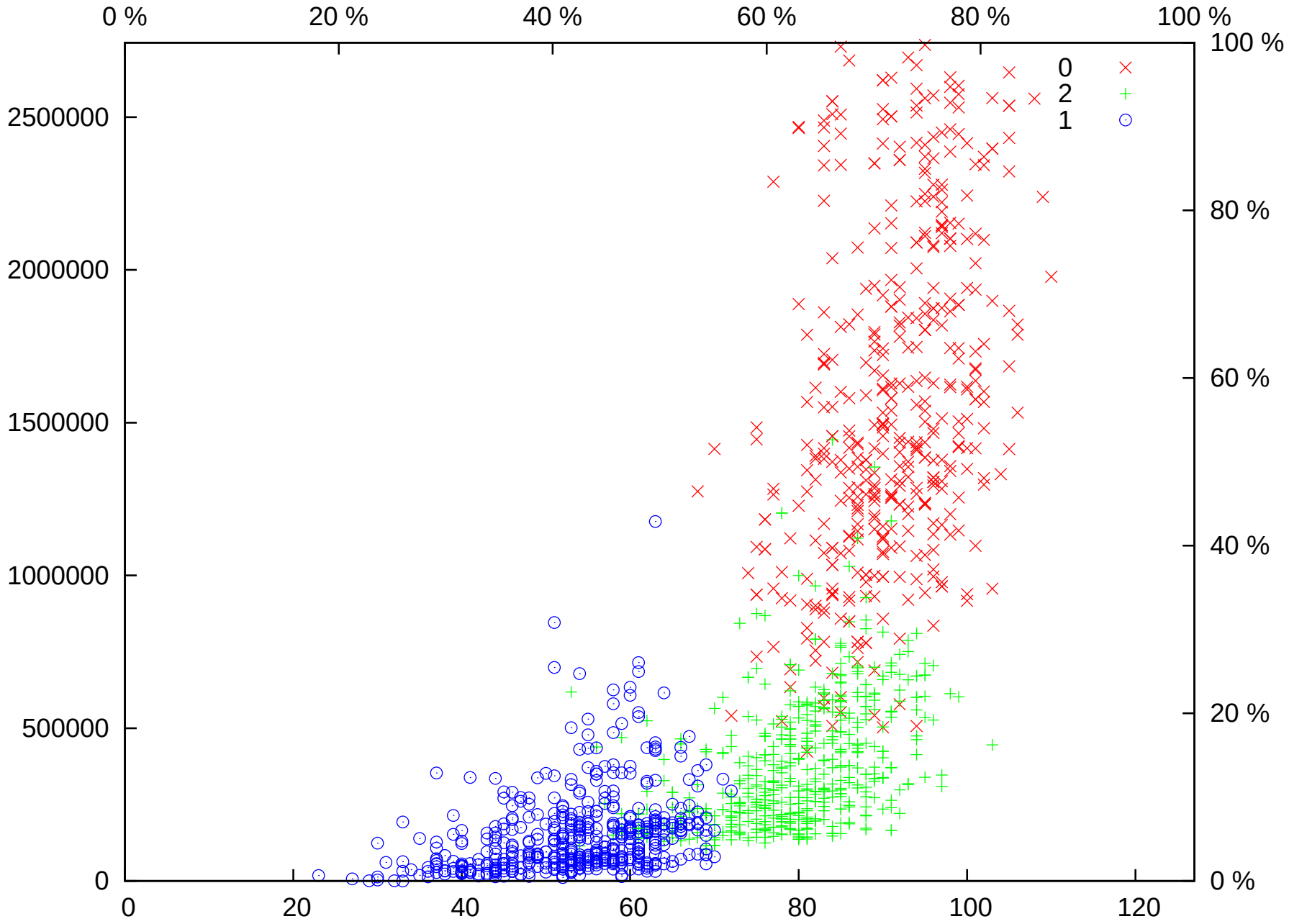
Mit freundlicher Unterstützung des AStAs der TU Kaiserslautern

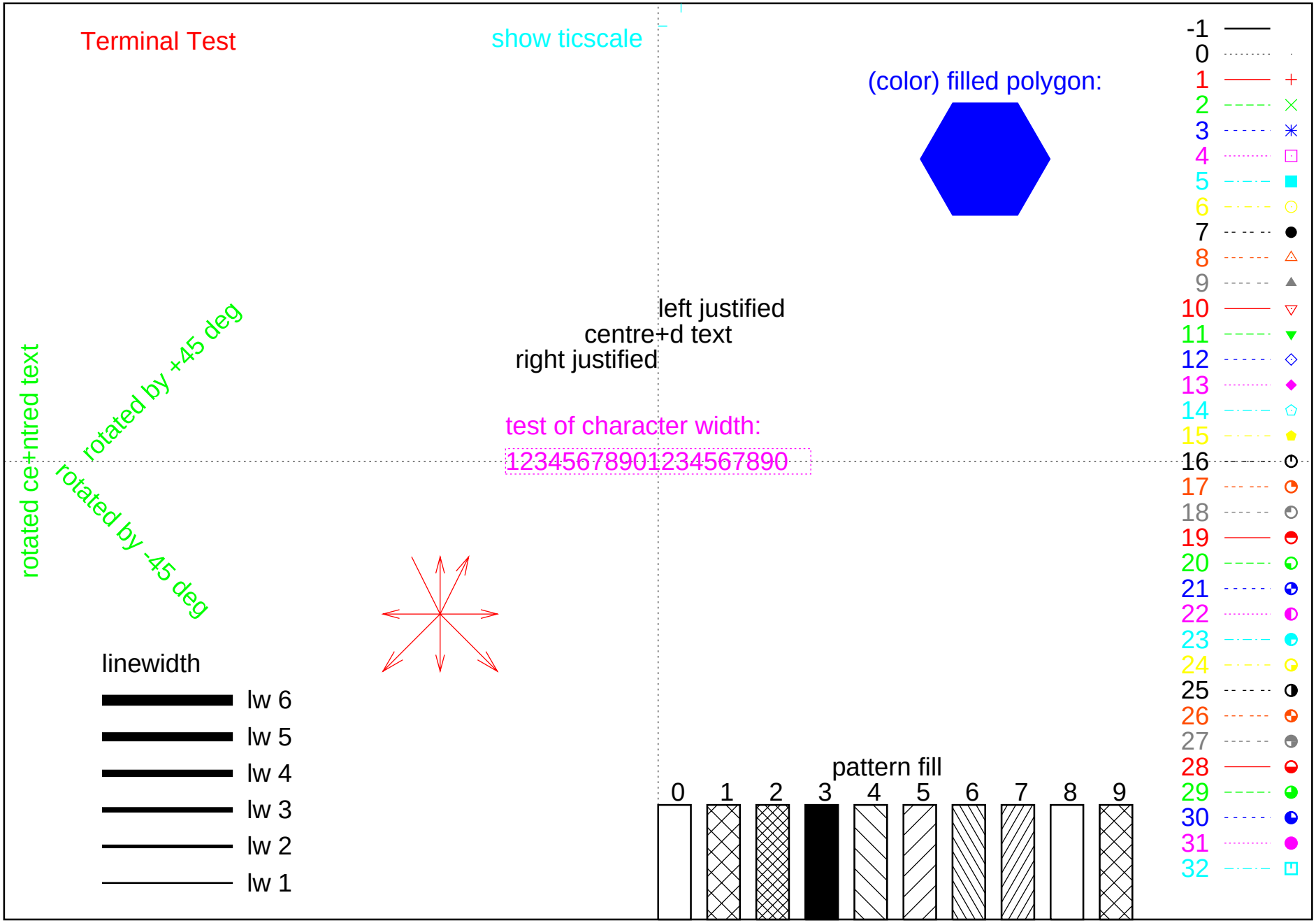
Graphiken

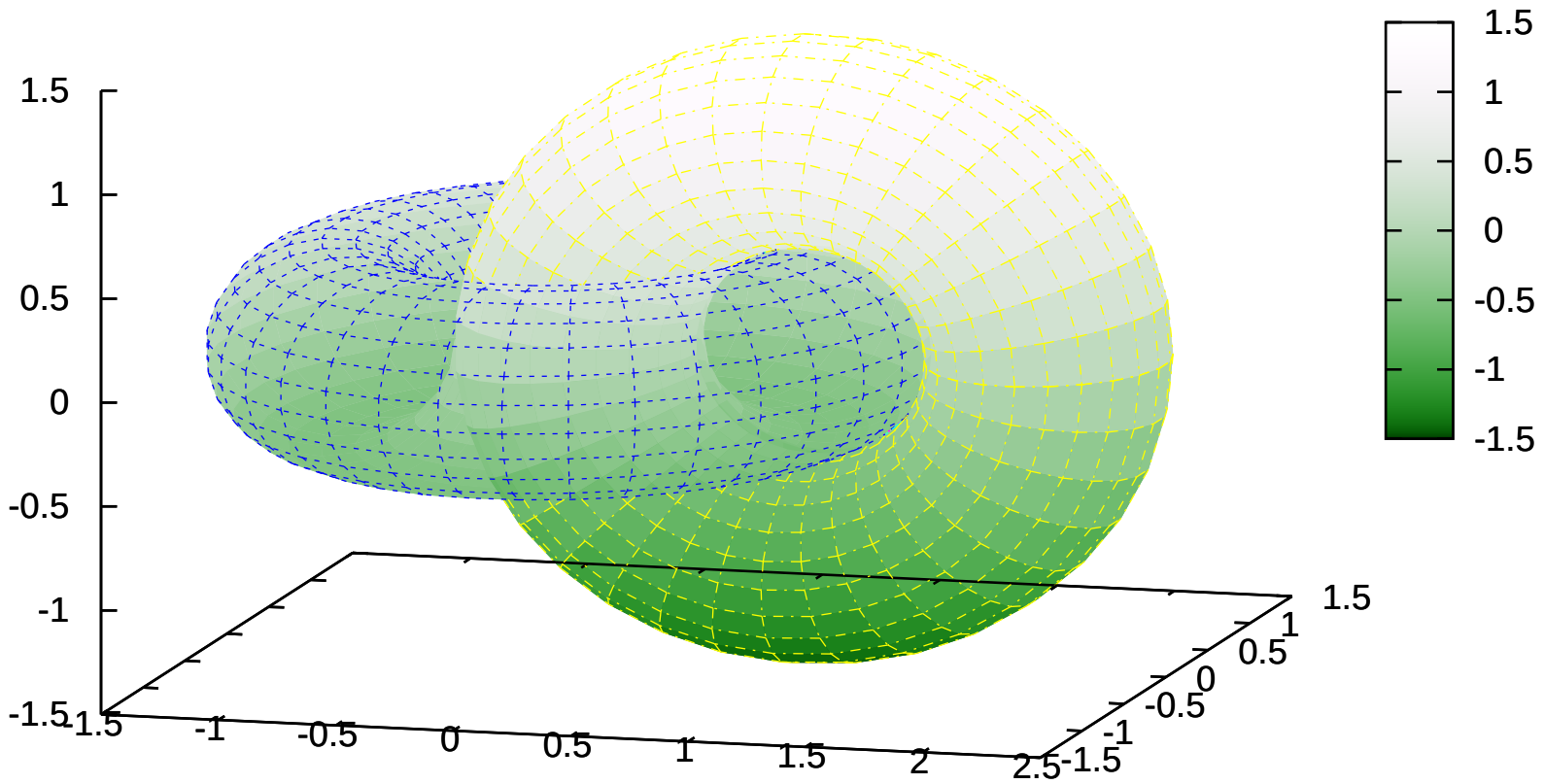
- Teil 1:
  - Einbindung von Graphikdateien in LaTeX
  - Generierung von Graphiken mit LaTeX-Befehlen
- Teil 2:
  - Generierung von Graphiken mit anderen Tools:
    - `gnuplot`
    - `dot`
    - `xfig`
    - `dia`
  - Konvertierung von Graphiken:
    - `convert` von Image-Magick
    - `fig2dev`
  - Automatische Konvertierung mit `make`

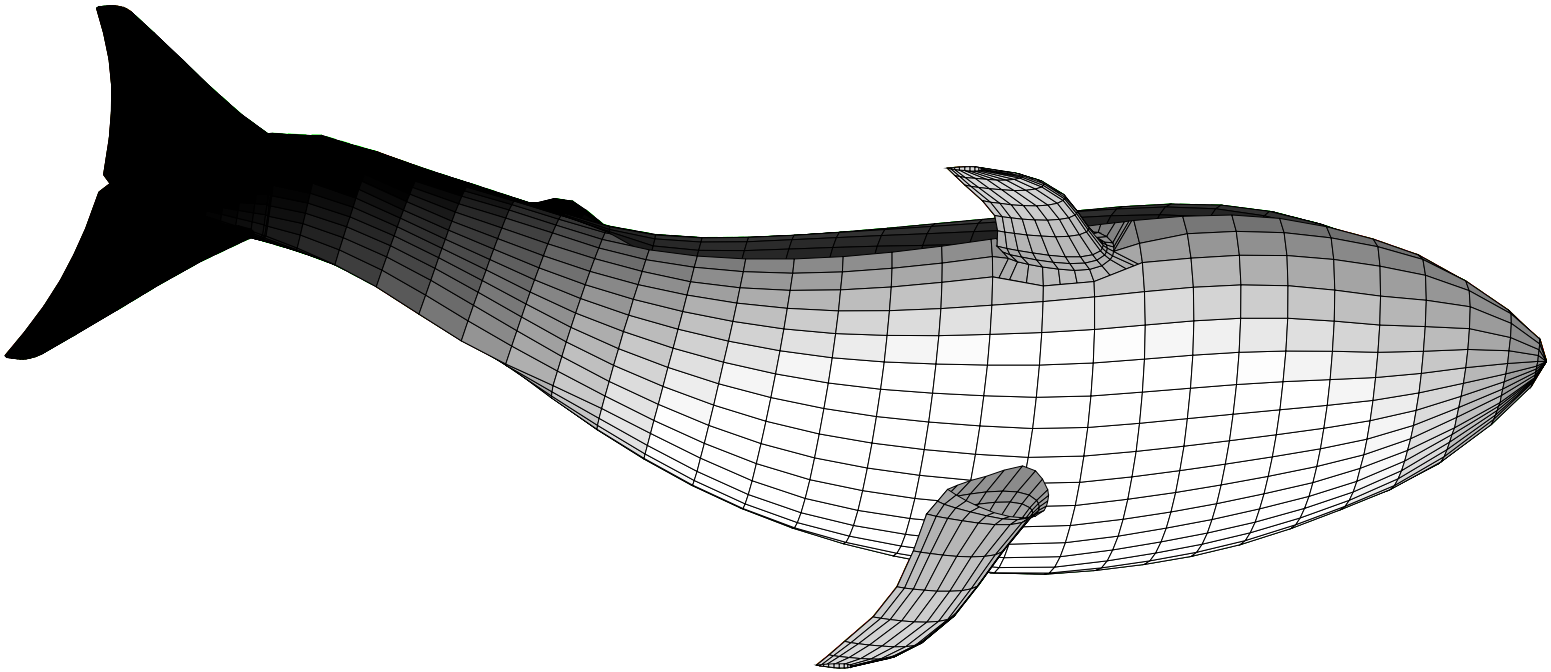
gnuplot-Beispiele

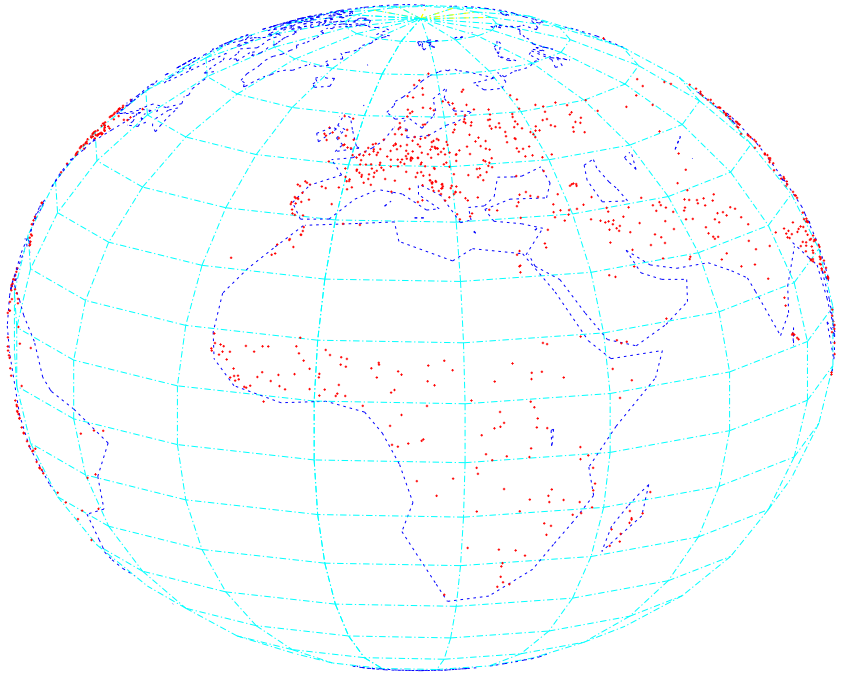



0 %
20 %
40 %
60 %
80 %
100 %
2500000
2000000
1500000
1000000
500000
0
0
20
40
60
80
100
120
0
2
1


Terminal Test
show ticscale
(color) filled polygon:
left justified
centre+d text
right justified
test of character width:
12345678901234567890
rotated by +45 deg
rotated by -45 deg
rotated ce+ntred text
linewidth
lw 6
lw 5
lw 4
lw 3
lw 2
lw 1
pattern fill
0 1 2 3 4 5 6 7 8 9
-1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32


1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
1.5
1
0.5
0
-0.5
-1
-1.5
1.5
1
0.5
0
-0.5
-1
-1.5





`gnuplot` allgemein

- `gnuplot` Version 4.0: April 2004
- Copyright (C) 1986 - 1993, 1998, 2004 Thomas Williams, Colin Kelley and many others
- mächtiges Werkzeug zur Darstellung von Graphen und Plots
- in den meisten Distributionen vorhanden
- `http://www.gnuplot.info/`

`gnuplot`-Befehle

Die wichtigsten Befehle sind:

- **plot** zum Zeichnen einer 2D-Funktion
- **splot** für 3D
- **help** zum Aufruf der Online-Hilfe
- **set** zum Setzen von Parametern

- Variablen: a=3
- Funktionen: f(x)=**sin**(x)
- Kommentare: # *so*

gnuplot-Befehle

```
1  plot x
2  # drei Funktionen auf einmal:
3  plot x, sin(x), 2**x
4  # ** = Potenz

6  # Legende nach links unten und umrahmt
7  set key left bottom box

9  # Variablen-Deklaration
10 a=0.2
11 # und -Benutzung
12 splot sin(sqrt(x*x+y*y))*a
```

gnuplot-Befehle

```
1 # 3D ohne Durchsicht
2 set hidden3d
3 set isosamples 20
4
5 splot sin(sqrt(x*x+y*y)), -0.8
6
7 # Farbmarkierungen der Funktionswerte
8 set pm3d
9 splot sin(sqrt(x*x+y*y))
```

gnuplot-Befehle

```
1 # Wahl des zu zeichnenden Ausschnitts
2 set xrange [1:*]
3 set yrange [-10:10]
4
5 # Logarithmische Skala
6 set logscale x
7
8 # letzten Plot noch einmal zeichnen
9 replot
```

gnuplot-Befehle

```
1  # Daten aus Datei
2  plot 'datafile' using 1:($4+$3) every 20\
3     title "data"

5  # verschiedene Stile
6  plot x with lines linetype 2 linewidth 1

8  # Funktionsdeklaration
9  f(x)=sin(x)

11 # Verschoenerungen
12 set label "kleine Graphik"
13 unset key
```

`gnuplot`: Plot von Daten

- **plot** ’ file ’ oder **splot** ’ file ’
- Daten liegen zeilenweise in Datei
- jede Zeile enthält zwei Werte x und y
- Leerzeilen unterbrechen durchgezogene Linien
- doppelte Leerzeilen strukturieren (für **index** n)
- **index** n wählt den n-ten Block aus (Zählung ab 0)
- **using** n:m wählt die n-te Spalte für x und die m-te für y aus
- **using** n wählt die n-te Spalte für y (Zählung ab 1), x wird die Zeilennummer (Zählung ab 0)
- **using** n:m:l für 3D
- **using** n:($3+$4) für weitere Rechnungen mit den Daten
- **using** "Zeile: %lf Spalte: %lf" für `scanf`-Format

gnuplot-Stile

- **with** lines: durchgezogene Linie (default)
- **with** points: einzelne Punkte
- **with** linespoints: durchgezogene Linie + einzelne Markierungen
- **with** dots: nur Pixel
- **with** impulses: Linie vom Punkt zur x-Achse
- **with** steps: Treppenstufen (z. B. mit **set samples** 30)
- **with** boxes: Box vom Punkt zur x-Achse
- **with** pm3d: Höhenfarben (nur 3D)
- **with** yerrorbars: Bereichsmarkierungen (für Daten mit Fehlerinformationen)
- u. v. m.

gnuplot-Stile: weitere Einstellungen

- **linetype** 1: Farbe der Linie
- **linewidth** 0.5: Linienbreite
- **pointtype** 5 Typ der Punkte
- **pointsize** 2: Größe der Punkte
- fast alles abkürzbar:

```
1 plot x w l lt 2 lw 2
2 replot -x w p pt 2 lt 1 ps 3
3 plot 'dat' u 1:2 i 0 t "data"
```

`gnuplot`-Ausgabe

- **set terminal** x11: Ausgabe im X-Fenster (default)
- **set terminal** postscript colour eps: Sinnvoll für LaTeX
- **set terminal** png: PNG-Format
- **set terminal** latex, **set terminal** eepic,
  set terminal texdraw, ...: Verschiedene Formate für TeX
- **set terminal** fig: für `xfig`
- **set terminal** table: ASCII-Dump der Werte
- u. v. m., Liste mit: **set terminal**

- **set output** 'dings.eps': Ausgabe in Datei umlenken

gnuplot-Skripte

- Damit gnuplot-Skripte möglich:

```
1  set terminal postscript eps
2  set output 'sin.eps'
3  plot sin(x)

5  set output 'cos.eps'
6  plot cos(x)

8  set output 'data.eps'
9  plot '-' with lines
10 1
11 2
12 e
```

`gnuplot`-Aufruf

- `gnuplot` ohne Parameter für interaktive Sitzung
- `gnuplot skript.gpi` für Skriptabarbeitung
- `gnuplot skript.gpi -` für interaktive Sitzung nach Skriptabarbeitung
- `gnuplot prepare.gpi skript.gpi` für Nacheinanderabarbeitung zweier Skripts, etwa:

Makefile:

```
1 %.eps: %.gpi postscript.gpi
2     gnuplot postscript.gpi $< >$@
```

postscript.gpi:

```
1 set terminal postscript eps
```

gnuplot-Verschönerungen

- **set format** y "%0.0f" setzt das Format für die Beschriftung an der y-Achse (default: "%g")
- unset **ytics** schaltet Beschriftung der y-Achse aus
- Beispiel für absolute Werte links, Prozentwerte rechts:

```
1 set y2range [0:100]
2 set format y2 "%0.0f %%"
3 set format y "%0.0f"
4 set ytics nomirror
5 set y2tics
6 opt=2743556
7 set yrange [0:opt]
8 plot 'data'
```

gnuplot-Verschönerungen

- unset **key** schaltet die Legende ab
- **set xzeroaxis** schaltet eine Null-Achse für x ein
- **set xlabel** "Zeit" beschriftet die x-Achse
- **set label** "Text" at 0,0 schreibt Text an eine bestimmte Position
- Alles aus:

```
1 unset key
2 unset border
3 unset xtics
4 unset ytics
5 #unset ztics # fuer 3D
6 plot sin(x)
```

gnuplot: weitere Spielereien

- Polarkoordinaten:

```
1 set polar
2 set trange [0:2*pi]
3 plot t
```

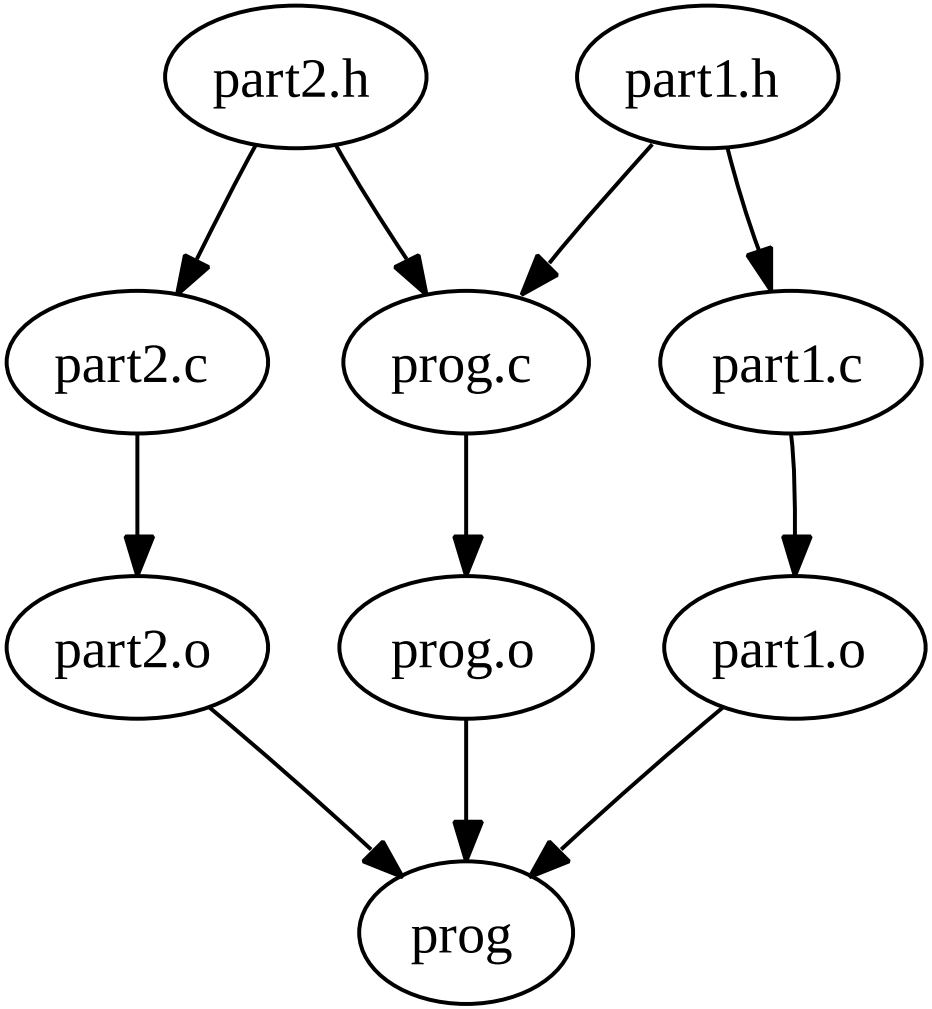
- Gitternetz: **set grid**
- Parameter:

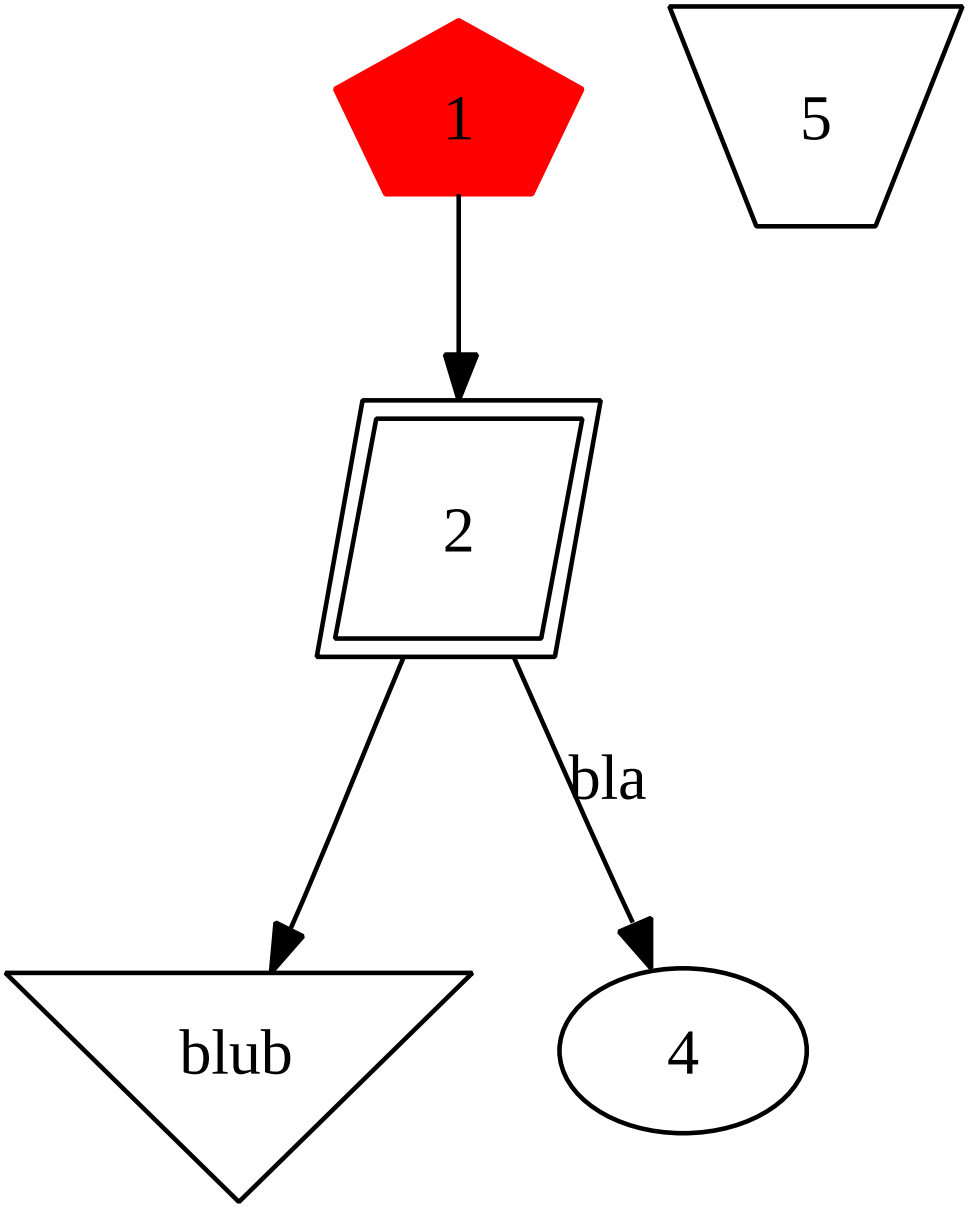
```
1 set parametric
2 set trange [0:2*pi]
3 set samples 1000
4 plot sin(t) + sin(t*100)/10,\
5      cos(t) + cos(t*100)/10
```

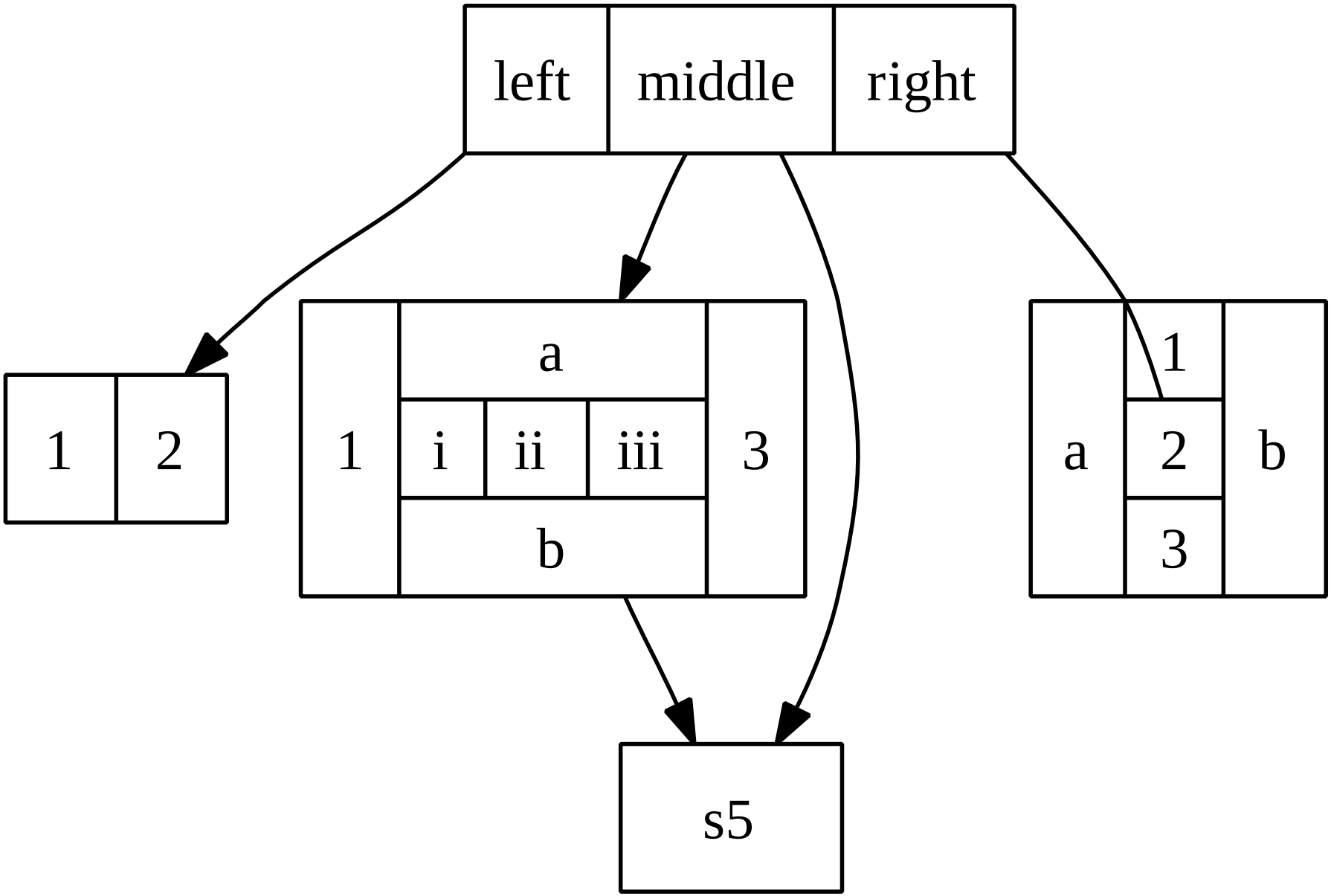
dot

- `dot` ist Teil der GraphViz-Sammlung
- mächtiges Werkzeug zum Darstellen von Netzwerken
- also gerichtete und ungerichtete Graphen, Zustandsmaschinen und derartiges
- Common Public Licence 1.0
- `http://graphviz.org/`

`dot`-Beispiel




1
5
2
bla
blub
4


left
middle
right
1
2
1
a
i
ii
iii
b
3
a
1
2
3
b
s5

dot: Quelltexte

```
1  digraph G {
2   "part1.c" -> "part1.o";
3   "part1.h" -> "part1.c";
4
5   "part2.c" -> "part2.o";
6   "part2.h" -> "part2.c";
7
8   "prog.c" -> "prog.o";
9   "part1.h" -> "prog.c";
10  "part2.h" -> "prog.c";
11
12  "prog.o" -> "prog";
13  "part1.o" -> "prog";
14  "part2.o" -> "prog";
15 }
```

dot: Quelltexte

```
1 digraph G {
2   1 -> 2 -> 3;
3   2 -> 4 [label="bla"];
4   1 [shape=polygon,sides=5,color=red,style=
      filled];
5   2 [shape=polygon,sides=4,skew=.4,peripheries
      =2];
6   3 [shape=invtriangle,label="blub"];
7   5 [shape=polygon,sides=4,distortion=.7];
8 }
```

dot: Quelltexte

```
1  digraph structs {
2   node [shape=record];
3   s1 [label="<l>left|<m>middle|<r>right"];
4   s2 [label="1|2"];
5   s3 [label="a|{1|<h>2|3}|b"];
6   s1:l -> s2;
7   s1:r -> s3:h [arrowhead=none];
8   s4 [label="1|{a|{i|ii|iii}|b}|3"];
9   s1:m -> s4;
10  s4 -> s5;
11  s1 -> s5;
12 }
```

`dot`: Aufruf

- `dot` ohne Parameter startet interaktive Sitzung
- `dot` ordnet dann die Knoten nur neu an
- Ausgabedateien mit: `dot -Tps -o output.ps input.dot`
- Ausgabeformate:
  - `-Tps`: Postscript
  - `-Tfig`: `xfig`-Graphiken
  - `-Tpng`, `-Tjpg`, `-Tgif`: Pixelformate

Makefile:

```
1 %.eps: %.dot
2     dot -Tps -o $@ $<
```

`dot`: Syntax

- Graph wird eingeleitet mit: **digraph** oder **graph** Name { ... }
- in den geschweiften Klammern folgen die Daten
- Bögen mittels Knoten1 −> Knoten2; oder
  Knoten1 −− Knoten2;
- einzelne Knoten durch ihren Namen
- zusätzliche Eigenschaften in eckigen Klammern:

```
1  n1 [label="Knoten"];
2  a -> b [label="Bogen"];
```

`dot`: strukturierte Knoten

- Knoten können eine interne Struktur erhalten
- einfach durch | im Bezeichner
- durch { } schachtelbar
- innere „Knoten“ durch Anker ansprechbar
- Anker setzen mit <anker>
- Anker aufrufen mit knoten:anker

Beispiel:

```
1 n [label="a|{1|<h>2|3}|b"];
2 a -> n:h;
```

`dot`: weitere Eigenschaften

- **label**: Beschriftung der Knoten und Bögen
- **shape**: Form der Knoten
  - ellipse : default
  - circle , box, diamond, triangle, hexagon u. v. m.
- **arrowhead**: Form der Pfeile der Bögen
  - normal: einfacher Pfeil, default
  - dot, odot, inv, invdot, none u. e. p. m.
- **color**: Farbe
- **width**: Breite der Knoten
- **height**: Höhe der Knoten

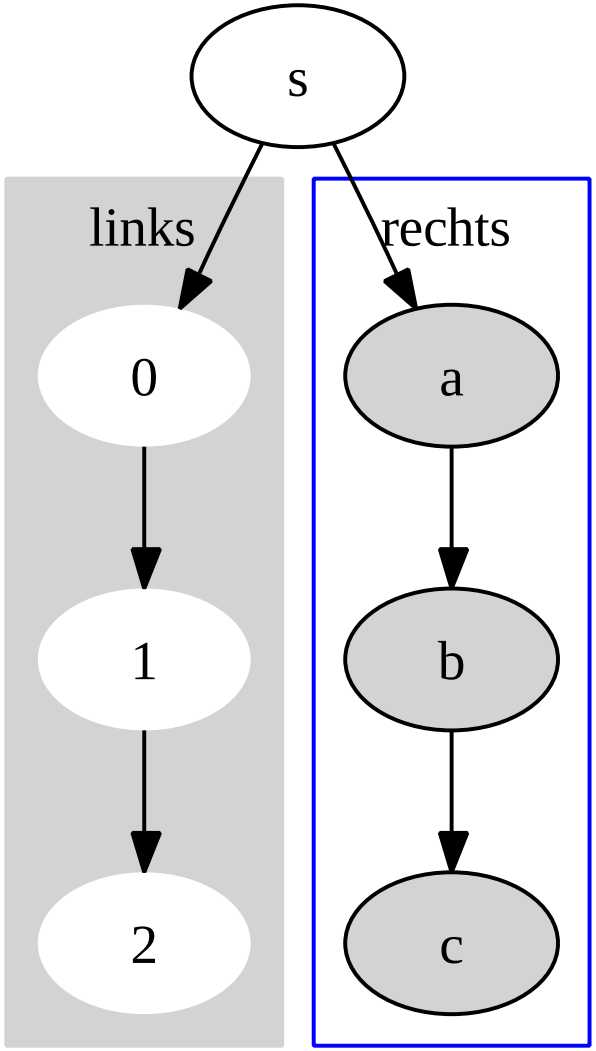
`dot`: Eigenschaften global setzen

- Knoteneigenschaften: **node** [...]
- Bogeneigenschaften: **edge** [...]

Beispiel:

```
1 digraph G{
2  node [shape=circle];
3  n1 [label="Knoten 1"];
4  node [width=2,shape=ellipse];
5  n2 [label="breiter"];
6  edge [arrowhead=dot];
7  n1 -> n2;
8 }
```

`dot`: Strukturierte Graphen



```
digraph G {
 subgraph cluster0 {
  label="links";
  style=filled; color=lightgray;
  node [style=filled,color=white];
  0->1->2;
 }
 subgraph cluster1 {
  label = "rechts"; color = blue;
  node [style=filled];
  a->b->c;
 }
 s->{0;a};
}
```

`dot`: noch mehr

- Mehr Informationen:
  `http://graphviz.org/Documentation/dotguide.pdf`

- weitere GraphViz-Programme:
  - `neato`: für ungerichtete Graphen, ohne Hierarchien
  - `lefty` und `dotty`: graphische Editoren

xfig

- xfig ist ein leicht gealtertes aber dennoch mächtiges vektororientiertes Zeichenprogram
- ganz gut geeignet für UML-ähnliches
- http://epb.lbl.gov/xfig/
- neueste Version 3.2.4: Dez 2002
  Version 3.2.5 seit Feb 2004 im alpha-Status
- Original Copyright (c) 1985 by Supoj Sutanthavibul
  Parts Copyright (c) 1994-1999 by Brian V. Smith
  Parts Copyright (c) 1991 by Paul King
  Other Copyrights may be found in various files
- sehr intuitiv

`xfig` in LaTeX einbinden

- `xfig` erzeugt und bearbeitet `.fig`-Dateien
- Umwandlung in von LaTeX lesbares Format nötig
- Umwandler: `fig2dev`
- Aufruf: `fig2dev -L eps input.fig output.eps`
- oder per Makefile:

```
1 %.eps: %.fig
2     @echo Generating figure $@...
3     @fig2dev -L eps $< $@
```

dia

- dia ist ein Zeichenprogramm für UML-ähnliches
- http://gnome.org/projects/dia/
- GPL
- noch intuitiver

`dia` in LaTeX einbinden

- `dia` erzeugt `.dia`-Dateien
- XML-Format, wahlweise gzipped

- `dia` kann in viele Formate exportieren
- Befehl zum Umwandeln: `dia -e output.eps input.dia`
- oder per Makefile:

```
1 %.eps: %.dia
2     dia -e $@ $<
```

Makefile

- viele Dateien müssen vor dem LaTeX-Lauf übersetzt werden
- automatische Behandlung wünschenswert

⇝ `make`

- Makefile enthält Regeln zum Erzeugen von Dateien aus anderen
- Syntax:

```
1 Zieldatei: Quelldatei
2     Befehl
```

```
1  plots=$(wildcard *.gpi)
2  ploteps=$(plots:.gpi=.eps)
3
4  dots=$(wildcard *.dot)
5  doteps=$(dots:.dot=.eps)
6
7  figures=$(ploteps) $(doteps)
8
9  %.eps: %.dot
10     dot -Tps -o $@ $<
11
12 %.eps: %.gpi
13     gnuplot $< >$@
14
15 %.aux:: $(figures)
16
17 include ~/Latexmakefile
```

`convert`: ImageMagick-Konvertierungsprogramm

- `convert` konvertiert Graphikdateien bequem in andere Formate
- Anwendung von Filtern möglich
- `convert` ist Teil des ImageMagick-Pakets
- `http://www.imagemagick.org/`
- Beispiele:

```
1 convert input.png output.eps
2 convert input.jpg -size 640x480 thumb.jpg
3 convert input.jpg -resize 50% small.jpg
```

Graphikformate für LaTeX

- je nach verwendetem LaTeX-Compiler unterschiedliche Graphikformate nötig
- `latex` mit Paket `graphicx`
  - Postscript (`.ps`, `.eps`)
- `pdflatex`
  - Bitmapformate (`.jpg`, `.png`)
  - ganze PDFs (Paket `pdfpages`, Befehl `includepdf`)

Ausblick auf Graphiken Teil 1

- Einbindung der Graphikdateien in LaTeX-Dokumente
- Generierung von Graphiken innerhalb von LaTeX
  - `gastex`
  - `picture`
  - Chemische Formeln
  - evtl. `pstricks`