# GUI ade

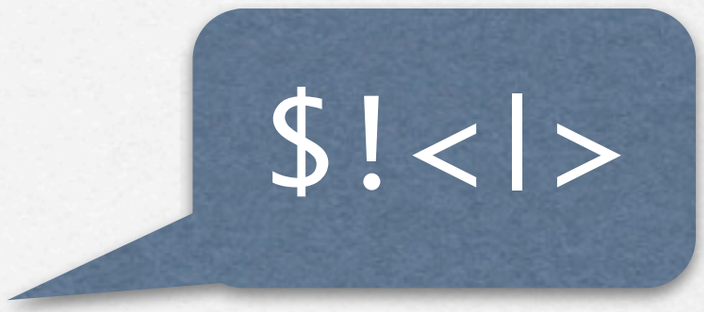
Wolfgang Kinkeldei

# Problemkind Shell

- [ ] Kommando selbst eingeben
- [ ] Syntax beachten
- [ ] Sonderzeichen
- [ ] Lernaufwand

# Problemkind Shell

- Kommando selbst eingeben
- Syntax beachten
- Sonderzeichen
- Lernaufwand



# Vorteil GUI

- Ansprechende Optik
- Befehle im Menu wählbar
- zahlreiche Optionen
- einfach
- kaum Lernaufwand

# menschliche Probleme

- [ ] große Mengen
- [ ] Unterschiede erkennen
- [ ] exakte Wiederholung
- [ ] komplexe Abläufe

# Was kann die Shell?

- Einzelne Befehle ausführen
  - z.B. Suchen aller JPEG Dateien

```
find . -name "*.jpg"
```

  - weitere Optionen möglich,
    z.B. Alter maximal 2 Tage

```
find . -name "*.jpg" -mtime -2
```

# Suchen in vielen Dateien

```
$ grep -r "Routing Funktion" *

Binary file script/linux-schulung/..schulung.tar
matches

script/linux-schulung/backup/linux.tex:  Routing
Funktionalität, Firewall-Möglichkeiten

script/linux-schulung/linux.tex:  Routing
Funktionalität, Firewall-Möglichkeiten
```

# Unterschiede sehen

- Vergleich zwischen 2 Versionen eines Modules bestehend aus vielen Dateien

```
$ diff -r  MIME-Lite-3.028 MIME-Lite-3.029
1157c1157
<         $attrs->{'MIME-Version'} = '1.0';
---
>         $attrs->{'mime-version'} = '1.0';

... einige 100 Zeilen mehr...
```

# Befehle verketten (Pipe)

- Muster: `Befehl-1 | Befehl-2`
- Bedeutung:
  Ausgabe von Befehl-1
  wird durch Befehl-2 weiterverarbeitet
- Anwendung:
  Transformation, Filterung, Sortierung

# Pipe Beispiel

- z.B. Bildgröße von Dateien filtern

```
$ identify*.jpg
mgp00001.idx.jpg JPEG 256x192 ...
mgp00001.jpg[1] JPEG 1024x768 ...

$ identify video/mgp*.jpg | grep 1024x768
mgp00001.jpg[1] JPEG 1024x768 ...
mgp00002.jpg[1] JPEG 1024x768 ...

$ identify video/mgp*.jpg | grep -v 1024x768
mgp00001.idx.jpg JPEG 256x192 ...
mgp00002.idx.jpg JPEG 256x192 ...
```

# Mehrfache Pipes

- Anzahl von Bildern nach Größe

```
$ identify mgp*.jpg | grep  1024x768 | wc -l
  24

$ identify mgp*.jpg | grep  256x192 | wc -l
  24

$ identify mgp*.jpg | cut -d " " -f 3 | sort | uniq -c
  24 1024x768
  24 256x192
```

# Textmanipulation

- variablen definieren

```
$ x="shell"; echo $x;
shell

$ y="hallo, $x"; echo $y
hallo, shell

$ stat -f%z datei.txt              # -f%z = Dateigröße
647

$ z="Hallo `stat -f%z datei.txt`"; echo $z
Hallo 647
```

# Einfache „Programme“

- Schleifen: Wiederholung von Befehlen

```
$ for f in *.jpg; do <<irgendwas>>; done
```

- Dateiname und Dateigröße ausgeben

```
$ for f in *.txt; do echo "$f: `stat -f%z $f`"; done;

ci_idea.txt: 699
clean_code_essence.txt: 647
textmate_licence.txt: 211
```

# Schleifen Beispiel

- Dateigröße aller PDF-Dateien zusammen

```
$ echo "1+2+3" | bc
6

$ s=0;
  for f in *.pdf; do s=$s+`stat -f%z $f`; done;
  echo $s | bc
4181133              # 0+188780+228599+42147+3721607
```

# Das wars

```
$ cat danke.txt
```

Danke für die Aufmerksamkeit

- Spickzettel:
  http://www.catonmat.net/blog/gnu-coreutils-cheat-sheet