Linux (allgemein): Der date-Befehl

Der Befehl date hat mehr Funktionen als er selber von sich preis gibt. Auch die Manpage und die Infopage verschweigen die Macht von `-d`/`--date`. Hier ein paar Beispiele die praktische Daten für Scripte liefern:

Das Datum von heute, gestern und morgen:

```
# echo "Heute: `date -I`"
Heute: 2010-08-06
# echo "Gestern: `date -d yesterday -I`"
Gestern: 2010-08-05
# echo "Morgen: `date -d tomorrow -I`"
Morgen: 2010-08-07
```

Umrechnungen in den Unix-Timestamp (auch UNIX-Zeit genannt): (Sekunden seit dem 01.01.1970)

```
# echo "Datum->Timestamp: `date +%s`"
Datum->Timestamp: 1281089450
# echo "Timestamp->Datum: `date -d "1970-01-01 1281089450 sec"`"
Timestamp->Datum: Fr 6. Aug 11:10:50 CEST 2010
```

Hier noch ein paar Berechnungsmöglichkeiten mit `-d`/`--date`:

```
date -d tomorrow
date -d yesterday
date -d "10 days ago"
date -d "1 month"
date -d "1 year"
date -d "1 month ago"
date -d "1 year ago"
date -d "4 weeks ago"
```

Auch Kombinationen sind möglich:

```
# echo "Heute: `date -I`"
Heute: 2010-08-06
# echo "Kombi1: `date -I -d "1 year 2 month"`"
Kombi1: 2011-10-06
# echo "Kombi2: `date -I -d "1 year 2 month ago"`"
Kombi2: 2011-06-06
# echo "Kombi3: `date -I -d "1 year ago 2 month ago"`"
Kombi3: 2009-06-06
```

Gerne wird auch Ausgabeformat verändert:

```
# date +"%Y-%m-%d %H:%M"
2010-08-06 11:10
```

Typische Daten:
%Y= Jahreszahl (4stellig)
%y= Jahreszahl (2stellig)
%m=Monat (2stellig)
%d= Tag (2stellig)
%H= Stunde (2stellig)
%M= Minute (2stellig)
%S= Sekunde (2stellig)

Den vollen Umfang an Möglichkeiten erhält man auf der Info-Page:

```
info coreutils 'date invocation'
```

Eindeutige ID: #1398
huschi
2010-08-06 12:48