

# Arbeiten mit Textdateien

- Öffnen von Dateihandles (zum Schreiben oder Lesen)
- Zeilenweises Bearbeiten des Dateiinhaltes
- Extrahieren von Informationen mittels Regex
- Schreiben in Dateien

# Öffnen von Dateihandles

```
open FILE, ">$dateiname" or die "Konnte  
Datei $dateiname nicht öffnen:$!\n";
```

## FILE

→ Bezeichner des Filehandle.

## ">\$dateiname"

→ Angabe des Dateinamen mit relativer oder absoluter Pfadangabe.

→ "<" - Zum Lesen Öffnen.

→ ">" - Zum Schreiben Öffnen.

open `FILE`, “>`$dateiname`” or die “Konnte  
Datei `$dateiname` nicht öffnen:`$!\n`”;

or die “Konnte Datei `$dateiname` nicht öffnen:`$!\n`”;

- Abfrage ob das Öffnen der Datei gelungen ist.  
Falls nicht, wird eine Fehlermeldung ausgegeben  
und das Programm beendet.
- `$!` ist eine spezielle Variable, die  
Fehlerinformationen enthält

- Dateinhandles müssen nach der Benutzung geschlossen werden.
- Durchführung mittels des `close` Operators

```
close FILE;
```

- Wenn die Datei zum Schreiben geöffnet war, werden dabei die Informationen in die Datei geschrieben

# Zeilenweises Bearbeiten des Dateiinhaltes

- sehr einfach über eine while Schleife zu realisieren.
- Einfache Ausgabe einzelner Zeilen der Datei mittels Zeilenoperator “<...>”
- Daten werden mittels automatisch bereitgestellter Variablen dem Benutzer zur Verfügung gestellt
- Variable “\$\_” bezieht sich immer auf den aktuellen Kontext, in diesem Fall auf die aktuelle Zeile der Datei

## Quelltext:

```
while(<FILE>){  
    print "$_";  
}
```

- Datei wird zeilenweise abgearbeitet
- Jeweils aktuelle Zeile wird ausgegeben
- Am Ende der Datei gibt der Zeilenoperator automatisch "false" aus und die Schleife wird beendet

# Extrahieren von Informationen mittels Regex

- Regexe (Regular Expressions) sind sehr mächtige Hilfsmittel zum Bearbeiten von Zeichenketten
- Perl ermöglicht sehr komplexe Regexe
- Syntax ähnlich dem in Unix Utilities

→ Textdatei enthält Daten folgenden Formates:

ID, Bezeichner, Datum

01, Klammer, Affe

02, Foo, Bar

→ Wir wollen die Daten ID und Datum herausfiltern

→ passender Regex:

`/(\d*),\w*,(\w*)/`

→ Ergebnisse werden in Variablen `$1` und `$2` dargestellt

## Quelltext unseres Programmes:

```
open FILE, "<$dateiname" or die "Konnte
  Datei $dateiname nicht öffnen:$!\n";
while(<FILE>){
    chomp($_);
    if(/(\d*),\w*,(\w*)/){
        print "ID: $1, Datum: $2\n";
    }
}
close FILE;
```

# Schreiben in Dateien

→ Öffnen eines neuen Dateihandles

```
open OUT, ">$dateiname" or die "Konnte  
Datei $dateiname nicht erstellen:$!\n";
```

→ **Achtung:** bereits vorhandene Dateien mit diesem Dateinamen werden ohne Nachfrage überschrieben!

→ Daher eventuell testen ob Datei schon vorhanden ist:

```
die "Datei schon vorhanden!\n" if (-e  
$dateiname);
```

## Neuer Quelltext:

```
die "Datei schon vorhanden!\n" if (-e  
"ergebnis.txt");
```

```
open IN, "<textdatei.txt" or die "Konnte  
textdatei.txt nicht öffnen:$!\n";  
open OUT, ">ergebnis.txt" or die "Konnte  
ergebnis.txt nicht erstellen:$!\n";
```

```
while(<IN>){  
    chomp($_);  
    if(/(\d*),\w*,(\w*)/){  
        print OUT "ID: $1, Datum: $2\n";  
    }  
}
```

```
close IN;  
close OUT;
```