Skriptsprachenorientierte Programmiertechnik

Perl Teil 4

Prof. Ilse Hartmann

Gliederung

Dateien lesen und schreiben

- Grundsätzliches Vorgehen:
  1. **Datei öffnen/erstellen, Dateihandle wird zugewiesen**
  2. **Daten lesen und/oder schreiben**
  3. **Datei(handle) schließen**

- Das Lesen aus Dateien erfolgt also analog zu Benutzereingaben aus der Standardeingabe (<STDIN>)!
- Das Handle STDOUT ist der Standardausgabekanal, d.h. normalerweise der Bildschirm

```
# Datei öffnen. „DATEI“ ist der Name des Handles
open DATEI, "irgendwas.txt";

# Lesen einer Zeile aus der Datei
my $zeile = <DATEI>;

# Datei(handle) schließen
close DATEI;
```

Dateihandles öffnen

- Das **open**-Kommando öffnet ein Dateihandle in verschiedenen Modi (lesend, schreibend, etc.)
- Der **Modus** wird durch entsprechende Zeichen vor dem Dateinamen gesteuert
- Im **schreibenden Modus** wird die Datei auf jeden Fall neu angelegt, eine etwaige bestehende Datei wird hierbei gelöscht bzw. überschrieben
- Im **anhängenden Modus** werden Daten an eine bestehende Datei angehängt (falls die Datei noch nicht besteht, wird sie neu angelegt)
- Statt Literale können Dateinamen auch aus beliebigen Ausdrücken (z.B. Skalaren) bestehen:

```
my $myfile = "irgendwas.txt";
open FILE, "> $myfile";
```

Dateihandles öffnen

Modus	Richtung	Beispiel
(ohne)	Lesend	open DATEI, "Dummy.txt";
<	Lesend	open DATEI, "<dummy.txt";
>	Schreibend	open AUSGABE, ">ergebnis";
>>	Anhängend	open LOG, ">>events.log";

Aus Handles lesen

- Der Zeileneingabeoperator eines Handles liefert beim Dateiende **undef** zurück

```
while ( defined  (my $zeile = <DATEI>))
{
    print $zeile;
}
```

- Über die Standardvariable **$_** ist eine verkürzte Schreibweise möglich

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

- Dies ist vor allem bei STDIN üblich:

```
while (<STDIN>) { ... }
```

- Eine komplette Datei kann auch über **foreach** in Verbindung mit **$_** eingelesen werden, indem das Handle im Listenkontext genutzt wird

- Die Datei wird hierbei jedoch komplett eingelesen, bevor die Schleife beginnt!

```
foreach (<DATEI>) {   print $_;   }
```

Fehler beim Öffnen abfangen

- Die **open-Funktion** liefert **undef**, falls das Öffnen der Datei nicht erfolgreich war

- Mit der **die-Funktion („sterben“)** kann ein Programm sofort mit einer Fehlermeldung beendet werden

- Die Perl-Variable **$!** enthält die System-Fehlermeldung im Klartext

Fehler beim Öffnen abfangen

Beispiel

```
my $erfolg = open (DUMMY, "<dummy.txt");
if (!$erfolg)
{
    print "Die Datei konnte nicht geöffnet werden!";
}
if (!open (LOG, ">>logs.txt"))
{
    die "Logdatei kann nicht angelegt werden! Fehler: $!";
}
```

In Handles schreiben

- Die **print-Funktion** kann auch zum Schreiben in Dateien genutzt werden

- Der Name des Handles steht direkt nach **print,** vor dem auszugebenden Text

- Das Handle für die Standardausgabe (normalerweise STDOUT) kann mittels **select** auf ein beliebiges Handle geändert werden

In Handles schreiben

Beispiel

```
if ( !open (AUS, ">ausgabe.txt") )
{
    die "Ausgabedatei kann nicht angelegt werden! Fehler: $!";
}
print  AUS  "Hier kommt eine Zeile.\n";

select  AUS;          # Ab hier Ausgabe in die Datei
print "Und hier eine zweite Zeile.\n";
print "Die dritte und letzte Zeile.\n";

select STDOUT;       # Wieder in die Standardausgabe schreiben
close AUS;
```