Perl und OpenVMS

Beispielprogramme

Ressourcen

Eigenschaften

- Perl ist eine stabile, portable Programmiersprache.
- Sie wird für kritische produktive Anwendungen im öffentlichen und privaten Sektor eingesetzt.
- Perl ist Open Source Software, lizensiert unter der "Artistic License" oder der "GNU General Public License".
- Sie ist verfügbar für alle modernen UNIX Derivate sowie für Win32, VMS, Netware, OS390 und viele mehr.
- "Perl is a language for getting your job done" (Larry Wall in "Programming Perl")
- TMTOWTDI (There's More Than One Way To Do It)

Basics

- Hello World

```
print "Hello World!";
```

- Implementierung als zweistufiger Interpreter (compile, run)

```
> perl –c hello.pl
hello.pl syntax OK

> perl hello.pl
Hello World!

>
```

Datentypen

- Scalare

```
$name = 'Barney';                       # string literal
$name = "$name Rubble";                 # interpoliert

$nbr  = 123;                            # Zahl
$big  = 123_432_000                     # besser lesbar
$nbr  = 0x1e3f;                         # Hex

$flt  = 1.23E-6                         # Float
```

- Automatische Typconvertierung

```
$nbr_str = '123';
$sum     = $nbr_str + 321;            # --> 444
print $nbr_str + 5;
$str     = 'humbug123';
print $str++;                         # --> humbug124
```

- Arrays

```
@names = ('Fred', 'Barney', 'Wilma');

@names = qw/
   Fred Barney Wilma
   Betty Pebbles Bambam
/;
```

- Hashes

```
%Flintstones = (
   Fred   => 'Wilma',
   Barney => 'Betty',
);

%Flintstones = qw/Fred Wilma Barney Betty/;
```

- Referenzen (ebenfalls Scalare)

```
$name = 'Pebbles';
$sref = $name;

print $$sref;                           # --> Pebbles

@names = qw/Pebbles Bambam/;
$aref  = \@names;
$aref  = ['Pebbles', 'Bambam'];         # anonymes array

print $aref->[1];                       # --> Bambam
```

```
%them = qw/Fred Wilma Barney Betty/;
$href = \%them

print $href->{Barney}                  # --> Betty
```

komplexe Strukturen

- werden mittels Referenzen erzeugt

```
%row0 = qw/Name Rubble     FirstName Barney Wife Betty/;
%row1 = qw/Name Flintstone FirstName Fred   Wife Wilma/;

@data = (\%row0, \%row1);

print $data[0]->{Wife};
print $data[0]{Wife};
```

- oder in einem Schritt mit anonymen Hashes

```
@data = (
   { qw/Name Rubble     FirstName Barney Wife Betty/ },
   { qw/Name Flintstone FirstName Fred   Wife Wilma/ },
);  # zusätzliches Komma wird ignoriert
```

- ein zweistufiger Hash (Hash of Hashes)

```
%data = (
   Rubble => {
      FirstName => 'Barney',
      Wife      => 'Betty',
   },
   Flintstone => {
      FirstName => 'Fred',
      Wife      => 'Wilma',
   },
);
```

Kontrollstrukturen

- Bedingungen

```
if ( $name eq 'Barney' ) {
   $force += 10;
   print "Bam! Bam!\n";
}

print "Huhu!\n" unless $idx == 20;

$child = $father eq 'Fred' ? 'Pebbles' : 'Bambam';
```

- Schleifen

```
while ( my $line = <$input> ) {
   print if $line =~ /error/i;
} continue {
   $count++;
}
```

```
for (my $i=0; $i<@names; $i++) {
   print $names[$i], "\n";
}

foreach my $n ( @names );
   print uc($n), "\n";
}

print uc(), "\n" for @names;
```

Reguläre Ausdrücke

- Matching

```
for my $line ( @text ) {
   push(@found, $line) if $line =~ /wichtig/;
}

@found = grep { /wichtig/ } @text;
```

- Ersetzen

```
for my $line ( @text ) {
   $line =~ s/wichtig/sehr wichtig/g;
}

s/wichtig/sehr wichtig/g for @text;
```

- Extrahieren

```
$txt = '%TCPIP-E-NOCONN';

if ( $txt =~ /%(\w+)-(\w+)-(\w+)/ )
   $sub  = $1;
   $type = $2;
   $msg  = $3;
}

my($sub, $type, $msg) = $txt =~ /%(\w+)-(\w+)-(\w+)/;
```

Perl unter OpenVMS

- voll unterstützt (statt speziell portiert)
- zahlreiche Module für VMS Spezifika
- leicht mittels Wrapper in .COM−Datei enzubetten
- Binärdistribution auf der Freeware CD
- zu wenig genutzt
- nicht alle Module portiert (Tk!)
- module unter VAX oft nicht getestet

Module für VMS

```
VMS::Misc        --> Typkonvertierungen
VMS::System      --> Systeminfo (+)
VMS::Device      --> Deviceinfo und −Manipulation
VMS::Stdio       --> IO mit RMS−Erweiterungen (+)
VMS::FileUtils   --> Utilities für Dateinamen
VMS::FindFile    --> f$search() Analogon
VMS::IndexedFile --> Zugriff auf RMS Indexed files
VMS::Monitor     --> System Performance Info (−)
VMS::Mail        --> Zugriff auf Mail (+)
```

Beispielprogramme

- Auslesen von VMS Systeminformationen

```
use strict;
use warnings;

use VMS::System;
use Data::Dumper;

my  @NodeList = VMS::System::node_list();
print Dumper(@NodeList);

foreach my $n ( VMS::System::sys_info_names() ) {
   my $info = VMS::System::get_one_sys_info_item($n);
   my $out = ref($info) ? Dumper($info) : $info;
   $out ||= '';
   print $n, " −−> ", $out, "n";
}
```

- Filtern des Operator Logfiles unter OpenVMS nach Subsystem Fehlermeldungen

```
use strict;
use warnings;

## used modules
use VMS::Stdio qw( :CONSTANTS :FUNCTIONS );

## INIT
$| = 1;

my $oplog        = 'OPC$LOGFILE_NAME';
my $line_buffer  = '';
my $max_buffer   = 16384;
my $sleep_int    = 5;
```

```
sub ParseData {
   my @records = split(/^s*$/m, $line_buffer);
   while ( @records ) {
      if ( @records == 1 ) {
         last unless $records[0] =~ /ns*n$/;
      }
      my $rec = shift(@records);
      my($subsys, $type, $errval) =  $rec =~ /%(w+)−(w)−(w+)/;
      next unless uc($type) eq 'E';
      ReportEvent($subsys, $type, $errval, $rec);
   } # END while
   $line_buffer = $records[0];
} # END ParseData
```

```
sub ReportEvent {
   my($sub, $typ, $err, $rec) = @_;

   print $rec;
}
```

```
my $last_pos;
while ( 1 ) {
   my $fh = vmsopen($oplog, 'shr=get,put,upd')
      or die "couldn't open $oplog, $!";
   $fh->setpos($last_pos) if $last_pos;
   while ( my $line = <$fh> ) {
      $last_pos     = $fh->getpos();
      $line_buffer .= $line;
      ParseData() if length($line_buffer) > $max_buffer;
   }
   ParseData();
   print "no more data, sleeping $sleep_int seconds...n";
   sleep($sleep_int);
}
```