Perl-Praxis

CPAN

Jörn Clausen
joern@TechFak.Uni-Bielefeld.DE

Übersicht

- Organisation des CPAN
- Module suchen und finden
- Perl-Module installieren
- Module für shared libraries installieren
- Fehler in Perl-Skripten finden und korrigieren

CPAN

- CPAN – Comprehensive Perl Archive Network

  http://www.cpan.org

- Module auf verschiedene Arten kategorisiert:

  http://www.cpan.org/modules/by-module/
  http://www.cpan.org/modules/by-category/
  http://www.cpan.org/modules/by-authors/id/

- zahlreiche Mirror, z.B.

  ftp://ftp.uni-bielefeld.de/pub/CPAN/

nach Modulen suchen

- verschiedene Suchmaschinen, z.B.

  http://search.cpan.org/

- Neuerungen auf CPAN:

  http://search.cpan.org/recent/

Aufgaben

- Suche nach einem Modul, um das Oster-Datum zu berechnen.

  Hinweis: englisches Wort fuer „Ostern“: easter

Date::Easter

- über Download-Funktion von search.cpan.org herunterladen
- oder aus passendem Verzeichnis auf CPAN:

  http://www.cpan.org/modules/by-module/Date/
- komprimiertes tar-Archiv: Date-Easter-1.14.tar.gz
- Readme-Datei direkt daneben: Date-Easter-1.14.readme
- Archiv entpacken:

```
$ tar zxvpf Date-Easter-1.14.tar.gz
```

- in Verzeichnis `Date-Easter-1.14` wechseln

CPAN-Module installieren

- Dokumentation lesen (`README`, `INSTALL`)
- Anweisungen in `Makefile.PL`
- typische Abfolge, um CPAN-Modul zu installieren:

```
$ perl Makefile.PL
$ make
$ make test
$ make install
```

- Problem: Modul wird in Perl-Verzeichnis installiert
- aufräumen:

```
$ make distclean
```

```
$ make install
Warning: You do not have permissions to install into
  /vol/perl-5.8/lib/site_perl/5.8.0/sparc-sun-solaris2.8 at
  /vol/perl-5.8/share/5.8.0/ExtUtils/Install.pm line 84.
Installing /vol/perl-5.8/share/site_perl/5.8.0/Date/Easter.pm
Installing /vol/perl-5.8/man/man3/Date::Easter.3
mkdir /vol/perl-5.8/lib/site_perl/5.8.0/sparc-sun-solaris2.8/auto/Date/Easter:
  Permission denied at /vol/perl-5.8/share/5.8.0/ExtUtils/Install.pm line 165
make: *** [pure_site_install] Error 255
```

CPAN-Modul konfigurieren

- eigenes Verzeichnis für Perl-Module: `$HOME/perl`

  ```
  $ perl Makefile.PL PREFIX='$HOME/perl LIB='$HOME/perl/lib
  ```

- erzeugt `Makefile`
- Modul kann von anderen Modulen abhängen
- suche nach „`PREREQ_PM`“ in `Makefile.PL`

```
$ perl Makefile.PL PREFIX='$HOME/perl LIB='$HOME/perl/lib
Checking if your kit is complete...
Looks good
Writing Makefile for Date::Easter
```

CPAN-Modul „übersetzen“

- erzeugtes `Makefile` verwenden:

```
$ make
```

- generiert Verzeichnisbaum unterhalb von `blib`:

```
arch/
      auto/
            Date/
                  Easter/
lib/
      Date/
            Easter.pm
      auto/
            Date/
                  Easter/
man3/
      Date::Easter.3
```

```
$ make
cp lib/Date/Easter.pm blib/lib/Date/Easter.pm
Manifying blib/man3/Date::Easter.3
```

CPAN-Modul testen

- vorgefertige Tests
- typischerweise in Unterverzeichnis `t`

```
$ make test
```

- falls Tests fehlschlagen, genauer untersuchen
- machmal 100%iger Erfolg nicht möglich
- manche Tests *sehr* einfach
- keine Garantie für Funktionsfähigkeit

```
$ make test
PERL_DL_NONLAZY=1 /vol/perl-5.8/bin/perl "-MExtUtils::Command::MM"
  "-e" "test_harness(0, 'blib/lib', 'blib/arch')" t/*.t
t/00basics.......ok
t/01gregorian....ok
t/02orthodox.....ok
All tests successful.
Files=3, Tests=27,  2 wallclock secs ( 0.90 cusr +  0.69 csys =  1.59 CPU)
```

CPAN-Modul installieren

- Modul und Dokumentation installieren:

```
$ make install
```

- erzeugte Dateien und Verzeichnisse unter PREFIX:

```
lib/
      Date/
            Easter.pm
      sparc-sun-solaris2.8/
            auto/
                  Date/
                        Easter/
            perllocal.pod
man/
      man3/
            Date::Easter.3
```

CPAN-Modul verwenden

- Modul in eigenes Programm einbinden:

```
use Date::Easter;
```

- Programm ausführen:

```
$ easter.pl
Can't locate Date/Easter.pm in @INC (@INC contains: /vol/
perl-5.8/lib/5.8.0/sparc-sun-solaris2.8 /vol/perl-5.8/sha
re/5.8.0 /vol/perl-5.8/lib/site_perl/5.8.0/sparc-sun-sola
ris2.8 /vol/perl-5.8/share/site_perl/5.8.0 /vol/perl-5.8/
share/site_perl .) at ./easter.pl line 3.
BEGIN failed--compilation aborted at ./easter.pl line 3.
```

CPAN-Modul verwenden, cont.

- Environment anpassen:

```
$ PERLLIB=$HOME/perl/lib:$HOME/perl/lib/sparc-sun-solaris2.8
$ MANPATH=$HOME/perl/man:$MANPATH
```

- in `.rcrc` (o.ä.) eintragen
- Dokumentation:

```
$ man Date::Easter
```

- Übersicht über installierte Module:

```
$ perldoc perllocal
```

- falls Perl-Skripte installiert werden:

```
$ PATH=$HOME/perl/bin:$PATH
```

Aufgaben

- Gib die Ostertermine der Jahre 2000 bis 2010 aus.

```
use Date::Easter;

foreach $year (2000..2010) {
    ($month, $day) = easter($year);
    print "$day.$month.$year\n";
}
```

shared libraries verwenden

- Perl-Modul als Interface zu Bibliothek
- typischerweise in „C“ geschrieben
- Beispiel: Compress-Zlib-1.22.tar.gz
- verwendet zlib
- an der TechFak in `/vol/local/` installiert
- Dokumentation lesen, `config.in` anpassen
- weitere Anpassungen für Solaris:

```
$ perl Makefile.PL LIB=... PREFIX=...
       LIBS'=-L/vol/local/lib -R/vol/local/lib -lz'
```

Die Datei `config.in` muß angepaßt werden:

```
-INCLUDE        = /usr/local/include
+INCLUDE        = /vol/local/include
```

Eigentlich müßte auch der Pfad unter `LIB` auf `/vol/local/` geändert werden. Leider reicht dies unter Solaris und einigen anderen Betriebssystemen nicht aus. Solaris benötigt weitere Informationen, um ein funktionierendes Perl-Modul zu erzeugen.

Ob das Perl-Modul korrekt gegen die in `/vol/local/` installierte Version der zlib gelinkt ist, kann man mit dem Programm `ldd` überprüfen:

```
$ ldd blib/arch/auto/Compress/Zlib/Zlib.so
        libz.so.1 =>     /vol/local/lib/libz.so.1
        libgcc_s.so.1 =>         /vol/gnu/lib/libgcc_s.so.1
        libc.so.1 =>     /usr/lib/libc.so.1
        libdl.so.1 =>    /usr/lib/libdl.so.1
        /usr/platform/SUNW,Ultra-80/lib/libc_psr.so.1
```

nach Installation

- *shared objects* von Architektur abhängig

```
lib/
      sparc-sun-solaris2.8/
            Compress/
                  Zlib.pm
            auto/
                  Compress/
                        Zlib/
                              Zlib.bs
                              Zlib.so
                              autosplit.ix
            perllocal.pod
```

- `Zlib.pm` eigentlich falsch eingeordnet

Aufgaben

- Mit Hilfe von `Compress::Zlib` kann man mit `gzip` komprimierte Dateien lesen. Komprimiere die Datei `romeo.txt` mit diesem Programm und lies sie anschließend mit Hilfe des Perl-Moduls wieder ein.

  Verwende die Funktion `gzopen`. Die in der man-page erwähnte zlib-Dokumentation ist in `/vol/local/include/zlib.h` zu finden.

```
use Compress::Zlib;

$gz = gzopen('romeo.txt.gz', 'rb');
while ($gz->gzreadline($line)) {
    print $line;
}
$gz->gzclose;
```

- Siehe man-page zu `Compress::Zlib`, Abschnitt "GZIP INTERFACE".

fehlerfreie Programme

- . . . sind eine Illusion
- aber Möglichkeiten in Perl, typische Fehler zu finden
- führe Programm `bogus1.pl` aus
- Vergleiche die Ausgabe mit dem Quelltext
- Aufruf: `perl -w`
- Pragma: `use warnings;`
- man-page `perllexwarn(1)`

- `bogus1.pl`:

```
$name = "Joe User";
print "Hello $nam\n";

$num = 12.345;
$val = 100 * num;
print "val is $val\n";

@a = (1..10);
$b = @a[0];
print "b is $b\n";
```

- Ausgabe:

```
Unquoted string "num" may clash with future reserved word at bogus1.pl line 7.
Scalar value @a[0] better written as $a[0] at bogus1.pl line 11.
Name "main::name" used only once: possible typo at bogus1.pl line 3.
Name "main::num" used only once: possible typo at bogus1.pl line 6.
Name "main::nam" used only once: possible typo at bogus1.pl line 4.
Use of uninitialized value in concatenation (.) or string at bogus1.pl line 4.
Hello
Argument "num" isn't numeric in multiplication (*) at bogus1.pl line 7.
val is 0
b is 1
```

fehlerfreie Programme, cont.

- strikte Einhaltung von Regeln
- führe Programm `bogus2.pl` aus
- entferne das Kommentarzeichen vor `use strict;`
- Fehler führen zum Programmabbruch
- Korrekturen:
  - Variablen mit `my` deklarieren
  - Subroutinen mit `&` oder `( )` aufrufen

- `bogus2.pl`

```
#use strict;

$name = "Joe User";
print "Hello $name\n";

$num = fortytwo;
print "num is $num\n";

sub fortytwo {
    return(42);
}
```

- Ausgabe:

```
Global symbol "$name" requires explicit package name at ./bogus2.pl line 5.
Global symbol "$name" requires explicit package name at ./bogus2.pl line 6.
Global symbol "$num" requires explicit package name at ./bogus2.pl line 8.
Global symbol "$num" requires explicit package name at ./bogus2.pl line 9.
Bareword "fortytwo" not allowed while "strict subs" in use at ./bogus2.pl line 8.
Execution of ./bogus2.pl aborted due to compilation errors.
```