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
```

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`

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
```

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

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
```

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.

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'
```

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.

fehlerfreie Programme

fehlerfreie Programme

- . . . sind eine Illusion

fehlerfreie Programme

- . . . sind eine Illusion
- aber Möglichkeiten in Perl, typische Fehler zu finden
- führe Programm `bogus1.pl` aus
- Vergleiche die Ausgabe mit dem Quelltext

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)`

fehlerfreie Programme, cont.

- strikte Einhaltung von Regeln
- führe Programm `bogus2.pl` aus
- entferne das Kommentarzeichen vor `use strict;`

fehlerfreie Programme, cont.

- strikte Einhaltung von Regeln
- führe Programm `bogus2.pl` aus
- entferne das Kommentarzeichen vor `use strict;`
- Fehler führen zum Programmabbruch

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

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