Neues von der POD-Front

Marek Rouchal
Infineon Technologies AG

Vortrag Munich Perl Mongers
22. Januar 2003

Inhalt

- POD - was ist das?
- Darstellung/Konversion von POD
  - generelle Schwierigkeiten
- Existierende Konverter
- POD Basismodule
- Eigene Entwicklungen

POD - Was ist das?

- POD = "Plain Old Documentation"
- POD ist in den Perl-Code eingebettete Dokumentation
- POD ist nicht "literal programming"!
  - siehe Artikel von M.J.Dominus
- Vorteil: Code und Dokumentation sind eng beieinander und können leichter konsistent gehalten werden
- Der Perl-Interpreter ignoriert POD
- Es gibt eine Reihe von Perl-Skripten und Modulen um POD darzustellen und zu verarbeiten

POD - Darstellung

- Die komplette Perl-Core online-Dokumentation ist in POD
- Anzeige v.a. durch **`perldoc`**
  - reine ASCII-Text Ausgabe
  - unter UNIX erfolgt Highlighting durch **`nroff`** (wie manpages)
  - **`perldoc`** hat ein paar spezielle Optionen für **`perlfunc`**
- Darstellung auch durch Konversion in andere Formate und Verwendung der entsprechenden Viewer (z.B. **`HTML`**)

Neues von der POD-Front
There's more than one way to do it!

POD Syntax (1) - Beispiel

```
=head1 Hauptüberschrift

Dies ist ein einfacher Absatz

=head2 Nebenüberschrift

Ein weiterer Absatz

  CODE, weil eingerückt
  weiterer Code

=over 4

=item 1. Nummerierung

Text text text text...

=item 2. ...

Mehr Text ...

=back

=cut
```

```
=head1 Hauptüberschrift

=begin html

<table>
<tr>...</tr>
....
</table>

=end html

Text

=for html
<IMG SRC="/republic.png">

Text B<fett> I<kursiv>
C<courier> F<Filenamen>
S<bei Leerzeichen nicht umbrechen>
E<auml> Sonderzeichen
L<Hyperlink|Pod::Checker/pod_check>
X<Indexeintrag>
Z<> ein nicht-Zeichen

=cut
```

Pod Syntax (2)

- Die POD-Syntax wird in **`perlpod`** beschrieben
  - also "**`perldoc perlpod`**" ansehen!
- Konventionen und Stilfragen werden in **`perlpodspec`** beschrieben
- Vor und nach einem **`=direktive`** Befehl muss je eine Leerzeile stehen - *wirklich leer, nicht mal Whitespace!*
- Eingerückter Text wird 1:1 dargestellt, in einem fixed-width Font (insbesondere für Code geeignet)
- Es gibt keine Tabellen, Bilder oder sonstigen Gimmicks

Schwierigkeiten des Formats (1)

- Viele Konverter unterstützen "magische" Auto-Formatierung im Text - mal mehr, mal weniger
- Mittels **`=for`** oder **`=begin`**/**`=end`** kann man bestimmte Extras einbetten - es gibt aber kein "**`=else`**"
- Es gibt keine festgelegten Listentypen - alles hängt vom ersten **`=item`** ab.
  - **`=item *`**
  - **`=item 1.`**
  - **`=item text`**
- Die Freiheiten führen zu recht unterschiedlichen Stilen - entsprechend unterschiedlich sieht die Dokumentation der Perl-Module aus
  - selbst die Konverter gehen von verschiedenen Stilen aus!
- Internationalität: **`E<...>`** versus encoding (Latin-1)

Schwierigkeiten des Formats (2)

- Hyperlinks (**`L<...>`**) verweisen auf Text im Zieldokument (**`=headX <text>`** oder **`=item <text>`**)
  - **`<text>`** kann Highlighting enthalten, oder **`*`** oder **`1.`**
  - **`<text>`** kann oft nicht 1:1 als reference-Marker verwendet werden (was passiert mit Blanks, Sonderzeichen usw.?)
  - dementsprechend sind bei der Konvertierung mehrerer POD-files zwei Durchgänge erforderlich
  - inkrementelle Konversionen einzelner Files praktisch unmöglich
- Modulnamen (z.B. **`File::Spec::Unix`**) und POD-files (z.B. **`/usr/local/lib/perl5/File/Spec/Unix.pm`**) sind schwer zuzuordnen
  - "**`podkonv /usr/local/lib/perl5/File/Spec/Unix.pm`**" woher soll es wissen, dass das Modul "**`File::Spec::Unix`**" heisst?
  - Eigentlich nur über Durchsuchen von **`@INC`** sauber zu lösen, wenn man noch OS-spezifische Unterverzeichnisse berücksichtigt

POD Konverter - **`Pod::Text`**

- **`Pod::Text`** ist der denkbar einfachste Konverter: konvertiert POD nach ASCII Text
- \+ Wird aktiv gepflegt (Russ Alberry) und ist im Perl Core
- \+ Wird von **`perldoc`** verwendet, wenn keine andere Option verfügbar ist
- \+ Unterstützt "intelligente" Terminals (Highlighting)
- – kann naturgemäß keine Hyperlinks darstellen

POD Konverter - **`Pod::Man`**

- **`Pod::Man`** ist der wohl älteste Konverter: konvertiert POD nach UNIX manpage format (**`*roff`**)

+ Wird aktiv gepflegt (Russ Alberry) und ist im Perl Core
+ Wird von **`perldoc`** in Verbindung mit **`nroff`** verwendet
  - je nach Fähigkeiten des Terminals schöne Resultate
+ Unterstützt alle Direktiven und Highlighting, verhält sich weitgehend robust bei "schlechtem" POD

– Merkwürdige Ausgabe langer **`=item`** Zeilen
– Hat eine Menge "guesswork" eingebaut, um bestimmte Teile im Text automatisch hervorzuheben
  - liegt aber manchmal daneben

POD Konverter - **`Pod::Html`**

- **`Pod::Html`** ist im Perl Core enthalten und stammt noch von Tom Christiansen
  - wird so gut wie nicht gepflegt
  - schlimmes Highlighting, z.T. unkorrektes HTML
  - Hyperlinks passen so gut wie nie
  - Verschachteltes Highlighting oft daneben
  - Probleme mit **< & >**
  - "guesswork" grottenschlecht

POD→HTML, etc.

- **`PodToHtml`** (Nick Ing-Simmons)
- **`Pod::HtmlPsPdf`** (Stas Bekman)
- **`Pod::HtmlTree`** (Michael Schilli)
- **`Pod::Tree::Html`** (Steven McDougall)
- **`Pod::POM::Html`** (Andy Wardley)
- **`Pod::Simple::Html`** (Sean Burke)
- **`Pod::HTML`** (Kenneth Albanowski)
- **`Marek::Pod::HTML`** (siehe hinten)

POD Konverter - **Tk::Pod**

- **Tk::Pod** zeigt POD in einem Tk-Fenster an

+ Schöne Darstellung, **L<...>** als Hyperlinks, die in der Regel einwandfrei funktionieren

– Wird nicht mehr aktiv gepflegt (Achim Bohnet)

– Dem Browser fehlt der "Back"-Knopf, oder eine History

POD Konverter - **`Pod::Latex`**

- **`Pod::Latex`** konvertiert POD in LaTeX source code, der dann durch LaTeX nach DVI, PS oder PDF formatiert werden kann
- \+ Wird gepflegt (Tim Jenness)
- \+ Endresultat qualitativ hochwertig
- \+ Eigene Nachbearbeitung der LaTeX-source möglich
  - z.B. Seitenstil
- – Konversionsprozess insgesamt recht aufwändig
- ☺ Esoterik: Aus LaTeX könnte man mit LaTeX2HTML auch nach HTML...

POD Konverter - Misc

- **`Pod::DocBook`** - DocBook SGML
- **`Pod::Dsr`**, Pod::Hlp - hat was mit VMS zu tun
- **`Pod::GroveBuilder`** - SGML::Grove
- **`Pod::HtmlHelp`**, **`Pod::WinHtml`** - Interface zu M$ HtmlHelp
- **`Pod::Lyx`** - Frontend für LaTeX
- **`Pod::PXML`** - spezielles XML
- **`Pod::PalmDoc`** - POD für den Palm
- **`Pod::Pdf`** - erzeugt direkt PDF
- **`Pod::PerlPoint`** - für die Slideshow
- **`Pod::RTF`**, **`Pod::Rtf`** - der Weg in die Windows-Welt
- **`Pod::XML`**
- **`SDF`** - simple document format (eine Art Erweiterung von POD)

POD Basismodule - **`Pod::Parser`**

- **`Pod::Parser`** ist das altehrwürdigste Basismodul
- stammt von Brad Appleton, letzte Version von Marek
- basiert auf Callbacks - das zu konvertierende File wird gelesen, geparst und für jede Direktive ein Callback aufgerufen
- der Textanteil wird als Baumstruktur übergeben
- in der PodParser-Distribution ist auch **`Pod::Checker`** und **`Pod::Find`** enthalten, sowie einige Hilfsutilities für Hyperlinks, das Cachen von Dokumenten und Link-Knoten
- Außer Bugfixes kaum noch Weiterentwicklung

POD Basismodule - **`Pod::Compiler`**

- **`Pod::Compiler`** basiert auf **`Pod::Parser`** und stellt den POD-Inhalt eines Files als Baumstruktur dar, die auf jeden Fall in sich konsistent ist
- Meldet bei der Erstellung der Struktur Fehler und Inkonsistenzen
- Methoden für die Behandlung von Listen, Knoten, usw.
  - komplexe Konvertierungen denkbar - das ganze Dokument liegt vor, und kann vorwärts/rückwärts/seitwärts durchgegangen werden
- War als Basisklasse für alle möglichen Konverter gedacht, ist aber langsam und schlecht programmiert
  - ich habe es noch nicht mal geschafft, es für mein **`Marek::Pod::HTML`** zu verwenden

POD Basismodule - **`Pod::Tree`**

- **`Pod::Tree`** liest ein POD-file und legt es als Baumstruktur ab
- Im Ergebnis mit **`Pod::Compiler`** vergleichbar
- Programmierinterface ähnlich umständlich wie **`Pod::Compiler`**
- Das im Modul enthaltene **`Pod::Tree::HTML`** macht auch keine korrekten Links
- Fazit: Hat sich nicht durchgesetzt

POD Basismodule - **Pod::POM**

- **Pod::POM** - "POD Object Model": liest ein POD-file und macht daraus eine objektorientierte Baumstruktur
- macht einen sehr durchdachten und sauber implementierten Eindruck
- sehr ordentliches Interface zu Konvertern ("double dispatch")
- tauchte aus dem Nichts auf - und scheint nicht mehr weiterentwickelt zu werden
  - ein paar Verbesserungsvorschläge wurden nicht beantwortet
- würde ich sofort für meine Entwicklungen benutzen, wenn ein paar Kleinigkeiten gefixt würden
  - OO-Methode zum Zusammenfügen von konvertiertem Text fehlt
  - beim Konvertieren von Text ist nicht bekannt, in welchem Kontext dieser steht

POD Basismodule - **`Pod::Simple`**

- **`Pod::Simple`** ist das neueste auf CPAN - eine vollständige from-scratch Überarbeitung von **`Pod::Parser`**
- Vom selben Autor wie **`perlpodspec`** - der sich auch um Konzepte kümmert
- Das erste Modul, das Ansätze bzgl. Unicode in POD enthält
- Stellt einen guten Kompromiss zwischen Geschwindigkeit und Komfort dar
- Basiert in erster Linie auf Callbacks
- Die Distribution enthält eine Menge Beispiele und Dokumentation
- Das ist meines Erachtens die Zukunft!

POD Utilities (1)

- **Pod::Checker** (**podchecker**) [**PodParser-1.21**]
  - überprüft POD Syntax und Struktur
- **Pod::Constants** [**Pod-Constants-0.15**]
  - kann Konstanten aus POD extrahieren
- **Pod::Coverage** [**Pod-Coverage-0.11**]
  - überprüft, ob alle sub's dokumentiert sind
- **Pod::Escapes** [**Pod-Escapes-1.03**]
  - zur Auflösung **E<...>** bzw. Latin-1
- **Pod::Find** [**PodParser-1.21**]
  - sucht bzw. lokalisiert POD files / module
- **Pod::PP** [**Pod-PP-0.1.2**]
  - POD Präprozessor

POD Utilities (2)

- **Pod::Select** [**PodParser-1.21**]
  - Extraktion von Teilen aus POD
- **Pod::Simplify** [**PodSimplify-0.0.4**]
  - Versuch einer ganzen Bibliothek (text, html, texi, ...)
- **Pod::Spell** [**Pod-Spell-1.01**]
  - ermöglicht Rechtschreibprüfung im POD
- **Pod::Stripper** [**Pod-Stripper-0.22**]
  - entfernt alles POD aus einer Datei
- **Pod::Tests** [**Test-Inline-0.15**]
  - extrahiert Code-Beispiele und Tests aus POD
- **Pod::Usage** [**PodParser-1.21**]
  - zeigt Syntax und/oder Manpage aus POD an
  - besonders gut mit **Getopt::Long** (-help, -verbose)

POD Utilities (3)

- **`Pod::Functions`** [CORE]
  - gibt die eingebauten Perl-Funktionen gruppiert a la **`perlfunc.pod`** aus
- **`Pod::HTML2POD`**
  - konvertiert HTML nach POD (sic!)

Aktuelles

- Es gibt eine Menge Diskussionen über POD als solches - ist das Format überhaupt noch "zeitgemäß"?
- In Perl 5.8 und 6.0 spielt Unicode eine wichtige Rolle - POD hinkt hinterher...
- Es gibt eine überarbeitete, modularisierte Version von **`perldoc`**
  - + macht einen soliden Eindruck
  - + Einfügen von zusätzlichen Output-Formaten möglich
  - + soll Konfiguration erlauben - z.B. Verwendung von RTF und WordPad unter Windows zum Anzeigen von "schönen" Manpages

Eigenes - **`Marek::Pod::HTML`**

- War ursprünglich als Ersatz für das Core-**`Pod::HTML`** gedacht
- Andreas König riet mir zu dem Prefix "**`Marek::`**"
- Ich habe alles, was CPAN zu dem Thema hergab zusammengetragen - das Modul geriet zur eierlegenden Wollmilchsau
- Hauptziel war Qualität des Outputs - korrektes HTML und funktionierende Hyperlinks
- verwendet **`Pod::Parser`**, **`HTML::Element`**: korrektes HTML "by construction"
- bei Konversion mehrerer Pod-Files ist die Konsistenz der Hyperlinks garantiert
- wie **`PodToHtml`** Ausgabe des **`HTML::Element`** mittels **`HTML::FormatPS`** als PostScript möglich
- Code ist leider schrecklich - nicht mehr schön zu warten

Eigenes - **Pod::MIF**

- Infineon verwendet FrameMaker zur technischen Dokumentation des IC-CAD-Designsystems, welches eine Menge Perl-Skripte/Module enthält
- Zur Zeit verwende ich **pod2fm** zur Konversion POD→ FrameMaker (und weiter → PDF) - ist ok, hat aber Schwächen in der Qualität und verwendet MML
- MML (historisch) unterstützt nicht alle FrameMaker-Features; MIF ist das eigentliche ASCII-Standardformat ("Maker Interchange Format")
- Neuerdings gibt es auch XML als Interface (FrameMaker 6.0)
- Ich wollte schon immer auf MIF umsteigen, erst unter Verwendung von **Pod::Parser**, dann **Pod::Compiler** und zuletzt mit **Pod::POM**
- Ich werde wohl einen neuen Versuch mit **Pod::Simple** starten, evtl. unter Verwendung von XML-in nach FrameMaker

The End

Danke für Eure
Aufmerksamkeit und Geduld!