Ruby/Tk-Kit

～ Tcl/Tk 環境を単一ファイルに結合する ～

永井　秀利

（nagai@ai.kyutech.ac.jp）

自己紹介

- ML 上での名乗りは永井@知能. 九工大
- 九州工業大学大学院 情報工学研究院
  知能情報工学研究系 知能情報メディア部門 助教
- Ruby/Tk メンテナ
- 本業は非ソフトウェア系
- Ruby は本業における道具

Ruby/Tk

- C 版 Ruby に標準添付の GUI ライブラリ
- 簡単に GUI 作成が可能なものとして知られる Tcl/Tk の wrapper ライブラリ
  - 稼働には Tcl/Tk のライブラリが必要
  - Tcl/Tk の拡張ライブラリのすべてを活用可能

Tcl/Tk

- Tcl はプログラム言語，Tk はその GUI ツールキット
- かなり古くから存在（最初のバージョンは 1988 年）
- 色々と批判されることも多いが，今でも広く使われ，開発・発展は継続中
- Perl や Python なども含め，標準的な地位で Tk をサポートする言語は多い

Ruby/Tk 導入時の問題点

- Ruby と Tcl/Tk とは独立した存在
  - 利点：必要に応じて種々のバージョンの Tcl/Tk との組合せが可能
  - 欠点： Ruby とは別に Tcl/Tk の導入が必要
- Ruby 自体の利用に必須の存在ではない
  - Tcl/Tk を含めて同時に強制導入とするのは難しい
  - ML 等で「 Tk が動かない」との質問も多い

→ Ruby のバイナリ配布において特に問題

Tcl/Tk の場合

- Tcl/Tk において，種々の拡張ライブラリを利用している場合にも類似した問題が存在
  - 便利な拡張が豊富に存在
  - でも未導入の Tcl/Tk 環境では使えない
  - 配布アプリの利用のために，導入を強いるべきか？
  - バージョンの違いによるトラブル発生のリスクも

Tclkit, Starkit, Starpack

- Tcl/Tk における配布問題への対応策
  - Tclkit
    - Tcl/Tk 実行環境の単一実行ファイル化
  - Starkit
    - アプリケーションの単一ファイル化
    - DLL を必要とする Tcl/Tk 拡張にも対応
  - Starpack
    - Tcl/Tk 実行環境とアプリケーションとを統合した単一実行ファイル化
    - Tclkit + Starkit と考えて良い

Tclkit の構造

(1) binary prefix

- 以下のライブラリを static link した実行ファイル
  - IncrTcl ： Tcl のオブジェクト指向拡張
  - Metakit ：データベースライブラリ
  - Zlib ：データ圧縮ライブラリ
  - TclVFS ： Tcl の Virtual File System 拡張

(2) Metakit dataset

- Tcl/Tk 本体と上記ライブラリの component をすべて含んだ database
- TclVFS を用いて mount され， Tcl からはファイルツリーの一部に見える

Tclkit の特徴

- 単一ファイルであるので， install/uninstall は単に cp/rm するだけ
- 実行時に component を展開したりしない
  - DLL の load 以外ではディスクを必要としない
  - DLL 必要時のみ tempfile にコピー， load 後に消去
- 元のファイルツリーを取り出したり，ファイルを追加・削除して再構築したりすることが可能
  - ツールプログラムで簡単に操作可能
  - 他のプラットホーム用のファイルも編集可能
  - 構築対象のファイルツリー構造をそのまま圧縮格納
    → VFS 上でもそのままの構造を維持

Tclkit から Ruby/Tk-Kit へ

- Ruby/Tk の core DLL である tcltklib.so に，Tclkit の構造を導入
  - Tcl/Tk 環境を丸々含んだ単一 DLL ファイル
  - core-tcltklib.so + MetaKit-dataset
    - core-tcltklib.so
      - Tcl/Tk, MetaKit, Zlib, TclVFS のライブラリを static link した以外は通常の tcltklib.so
    - MetaKit-dataset
      - 上記ライブラリに加え，同梱したい Tcl/Tk 拡張をまとめたもの

Ruby/Tk-Kit の起動ステップ

1. tcltklib.so を読み込む
2. load 時の __FILE__ 情報から， tcltklib.so の置き場所を確認（仮に /x/ruby/y/tcltklib.so とする）
3. MetaKit dataset を開く
4. database 操作で VFS のトップレベルから boot.tcl を検索して読み込み，実行
   1. VFS を mount （/x/ruby/y/tcltklib.so/... となる）
   2. Tcl/Tk のライブラリ検索パスを VFS 上の lib ディレクトリ (/x/ruby/y/tcltklib.so/lib) に設定
5. tcltklib の初期化完了

Ruby/Tk-Kit の長所 / 短所

- 単一ファイルである
- バイナリ配布時に Tk 関連のトラブルを回避可能
- 構造が同じなので，Tclkit 用のツールが使える
- 環境にインストールされている Tcl/Tk と干渉しない
- 圧縮により，展開した場合と比較してコンパクト
- 拡張の追加に特殊な make は必要なし．展開した VFS ディレクトリに追加して再構築 (pack) するだけ（どうしても必要なら自己改編も可能）
- Tcl/Tk の更新には tcltklib.so の作り直しが必要
- Tcl/Tk と Ruby/Tk とで同じ拡張を使いたい場合は，再構築だけとはいえ手間が増える

Ruby/Tk-Kit 開発の現状

- Linux 用と Windows (RubyInstaller) 用の試作品公開
- 基本での構築後，ActiveTcl 上の Tcl/Tk 拡張を単純にコピーして再構築するだけで使えることを確認
- trunk と ruby_1_8 との tcltklib.c には，作成に必要なコードを commit 済み
  - 通常のコンパイルオプションでは非活性
  - VFS 上に必要な Ruby/Tk 用 boot.tcl の内容まで組み込むべきか等，もう少し改善すべき点も残る
- 必要なライブラリが多い（しかも条件がうるさい）ので，extconf.rb での対応をどうすべきか思案中
- Ruby と Tclkit との両方が使えるすべての環境のサポートが目標
  （参考：http://www.equi4.com/pub/tk/downloads.html）

Ruby/Tk-Kit から Rubykit へ

- 今はまだ将来構想の段階
- Ruby/Tk-Kit (Tclkit) の構造を Ruby 本体に持ち込む
  - Ruby 自体の単一実行ファイル化
  - install/uninstall は cp/rm だけで OK
  - Tcl/Tk の wish コマンドのように，引数の有無でインタープリタとなったり irb になったり…
  - Tcl/Tk と共通の VFS 構造なので， Ruby/Tk まで問題なく一体化が可能
  - gem 対応は？
    - gem の MetaKit database を VFS として mount
    - 別ファイルとすることで動的に更新可能

Rubykit 体系の基本構想

- Tclkit, Starkit, Starpack の体系を模倣
- Rubykit が Tclkit に相当
  - 実行対象の指定なしなら irb を起動
- Starkit 相当が Rubykit で起動するアプリケーション単一ファイル
  - Starkit 同様, boot 用の短い Ruby コード + 必要ライブラリまで含めた MetaKit dataset の VFS
  - このタイプの場合のみ, 実行時の自己更新が可能
  - 複数 OS 用 DLL を同梱した汎用化も可能
- Starpack 相当がアプリケーション単一実行ファイル
  - Rubykit + Starkit 相当
  - 別ファイルを作って良いなら, gem も OK ?

他の実行ファイル化手法との比較

- http://route477.net/d/?date=20090606 にある比較表

	exe 作成	Linux/Mac 用バイナリ作成	テンポラリディレクトリを使わない	必要なファイルのみ格納
Ocra	○	×	×	○
Exerb	○	×	○	○
Crate	×	○	○	○
RubyScript2Exe	○	○	×	×
Rubykit（計画）	○	○	○	○

- 開発およびテストをした時の環境（ファイルツリー）をそのまま pack して実行ファイル化できる
- 実行中の追加がないなら gem 利用も多分問題ないはず
- Ruby がサポートする環境に汎用なものとするのが目標

まとめ

- 開発中の Ruby/Tk-Kit と，その発展としての Rubykit の構想について述べました
- Ruby が動くすべての環境で汎用的に広く使われるものにできればいいなと考えています
  - 配布が容易で，必要ならユーザによる編集も可能
  - 望むなら，ファイルを展開した状態での利用も可能

    → そのまま実行可能なバイナリ配布，
    兼，インストール用パッケージとできる
  - コード隠蔽に関する特別な機能はないので，必要なら pack するアプリケーション上で対処
- まだまだ道程は長いので，協力者募集中です