[Hack]
FlatODFとスクリプト処理で
お手軽帳票もどき
～Drawではがきの宛名書き～

2019-11-30

第17回 関西LibreOffice勉強会

岩橋 伴典（いさな）

2019-12-02 rev.1

自己紹介

- プロフィール
  - 氏名 = 岩橋 伴典 ; ハンドル = いさな ; コールサイン = JO3EMC ;
  - E-mail = jo3emc@jarl.com ; Twitter = @jo3emc ;
  - 職業 = 元会社員（無線技士）;
  - タグ = 無線, LibreOffice, Tcl/Tk（含 AndroWish）, Arduino（含 M5StickC）, にょロボてぃくす, アニメ, 日本SF大会, 新井素子, 谷山浩子, オカリナ ;
  - 詳しくはWebで♪(http://jo3emc.c.ooco.jp/)
- LibreOfficeとの関わり
  - OpenOffice.org 1.0リリースの頃からのおつきあい。
  - よく利用する機能 = Draw, Writer, Calc ; // プレゼンテーションしませんw。
  - コミュニティーへのコミット = 翻訳、バグ報告、Askで少しだけ…。 ;

何の話？

- 帳票システム≒ 自動文書生成 関連のお話です。
  - 簡易なものなら「FlatODF」形式を使うと手軽に実現できますよ。
    というお話…を過去に何度かさせてもらいました。
    - 2017-04-29　第14回 関西LibreOffice勉強会
      「Office文書を手打ちハックする ～FlatODF活用のすすめ～」
      - 説明文ばかりで絵もデモもない。けど語りたい内容は概ねこの中に。
    - 2018-05-13　LibreOffice Kaigi 2018
      「FlatODFとスクリプト処理でお手軽帳票もどき」
      - 2017-04-29を受けてデモするつもりが準備不足で惨敗。
  - 今回は惨敗したデモンストレーションの再挑戦です。
    - 身近で分かりやすい例として（？）「はがきの宛名書き」を取り上げます。
    - 帳票処理のデモに絞り、FlatODFの細かい話は割愛します。

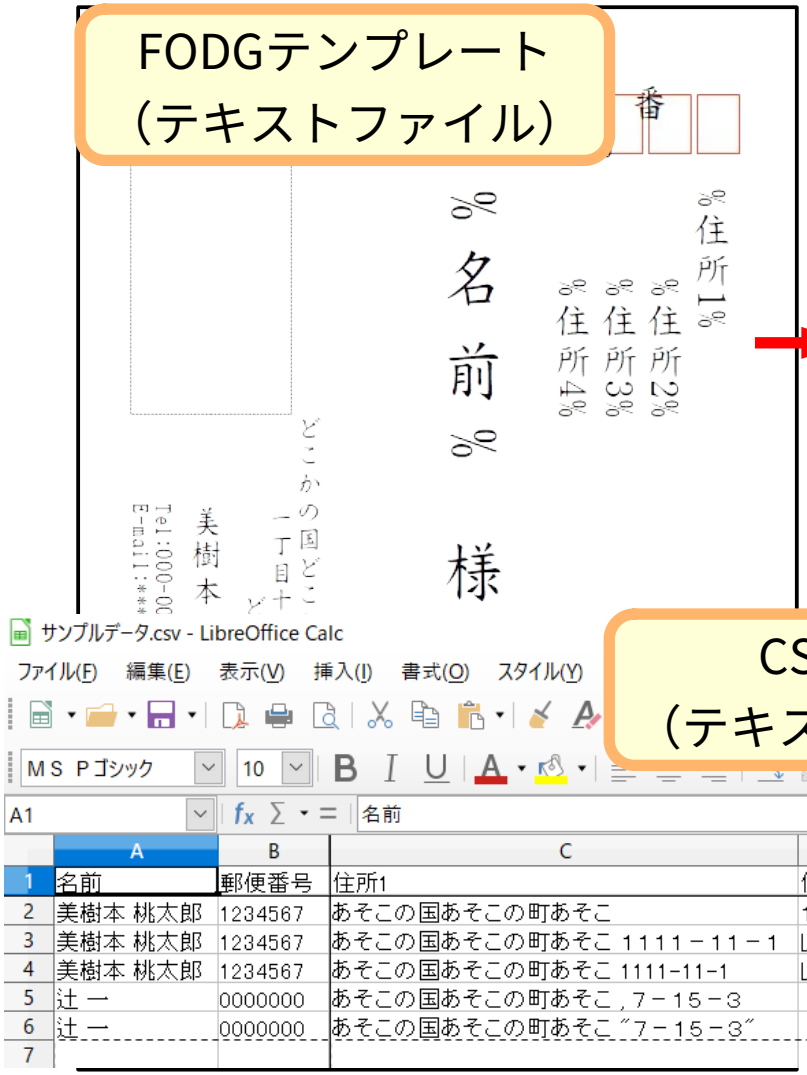
過去の帳票関連報告例（自身のもの除く）

- 2013-12-14　第5回 関西LibreOffice勉強会　　のがたじゅんさん
  「LibreOfficeのフラットODFとRedmine,
  　Git(Gitolite, SparkleShare)で文章共有システムを作ってみた」
  - 文書作成というよりは文書共有にFlatODFを活用する試み。diff取ったり。
- 2015-07-11　第10回 関西LibreOffice勉強会　　中村 倫さん
  「OpenDocumentを読み解こう」
  - 「Ruby」を使ったODFファイル操作。
- 2016-09-24　第13回 関西LibreOffice勉強会　　三浦 一仁さん
  「むしゃくしゃしたのでopen document で帳票テンプレート」
  - ODSをテンプレートにしたWEB帳票システム構築のデモ。
  - 「Java」+「JOpenDocument」ライブラリを使用。
  - ZIP解凍・圧縮＋エディタ編集によるODSファイルの直接編集も紹介。

@arachan@githubさんによる関連記事がQiitaにも。

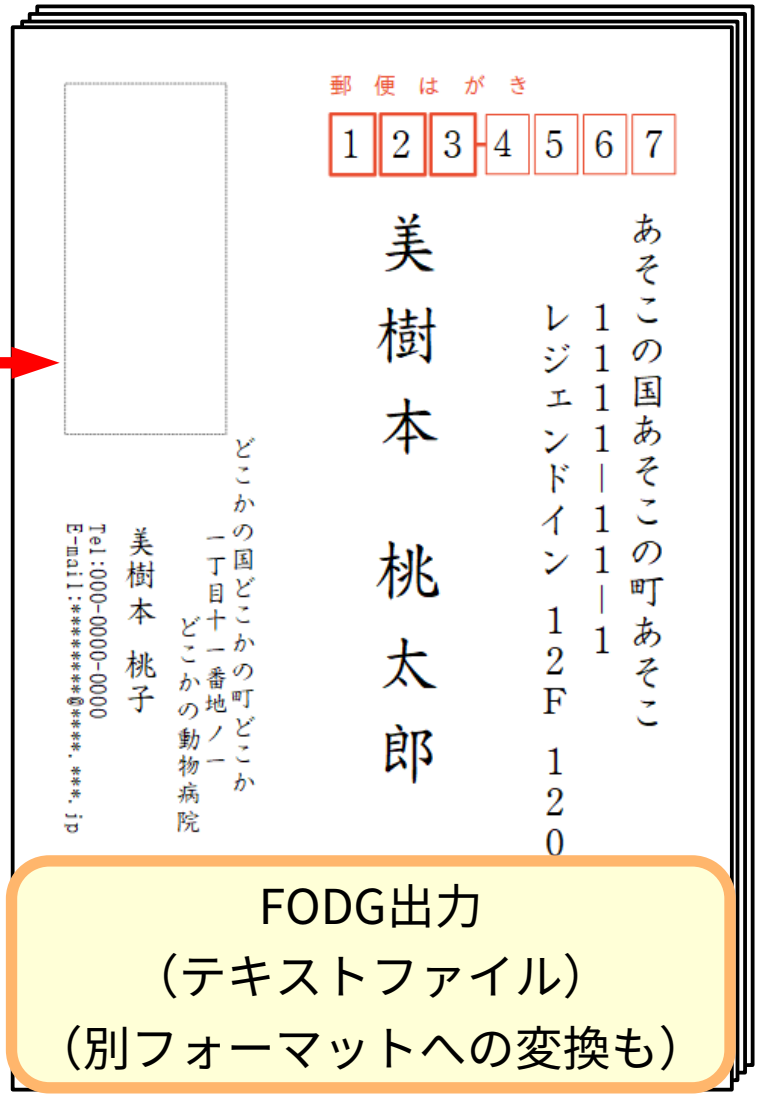
デモのイメージ

FODGテンプレート
（テキストファイル）



スクリプト
で差し込み

CSVデータ
（テキストファイル）



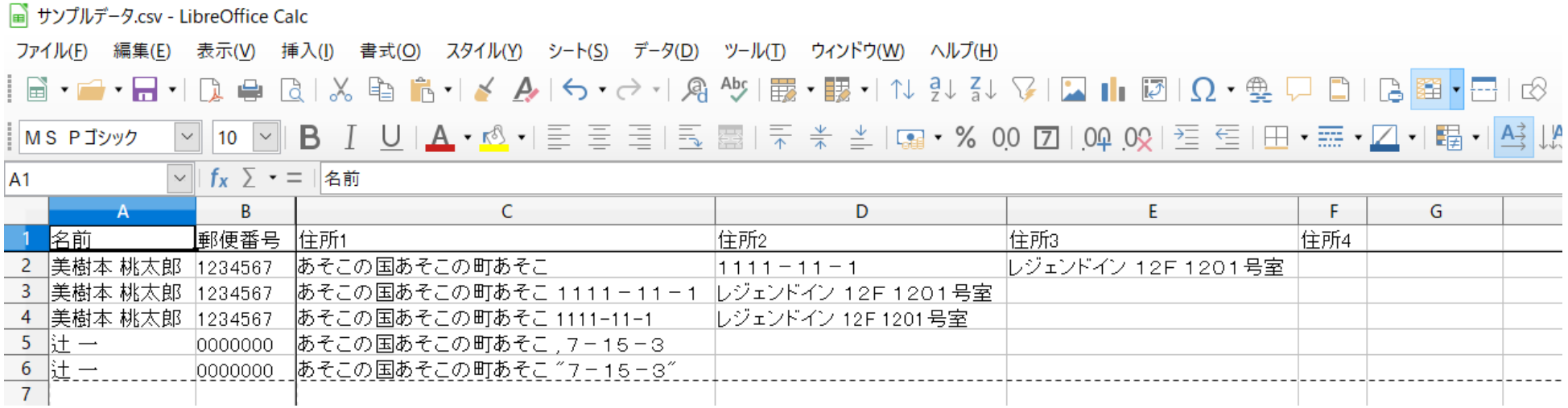
FODG出力
（テキストファイル）
（別フォーマットへの変換も）

今回のデモの特徴

- 単純な文字列置換だけで扱える簡易な内容に絞っています。
  - その気になればかなりの可能性を秘めていますが、今回は割愛。
- テンプレートにDrawファイル(FODG)を使います。
  - あまり例を目にしません（メリット・デメリット は末尾に）。
- 処理の基本部は独立したスクリプトプログラムだけで行います。
  - 特別な開発環境やライブラリ類は使いません。オープンです♪
    - ファイル入出力と文字列置換ができればお好みの開発環境が使えます。
  - Officeソフトがなくても文書生成できます。
    - 雛形作成や閲覧には要りますが…WEBシステムなら見れなくても、ね？
  - おまけでLibOと外部プログラム実行機能があれば形式変換や印刷も。
- 出力もFODGファイルなので、後からLibreOfficeで追加編集ができます。

サンプルデータの用意

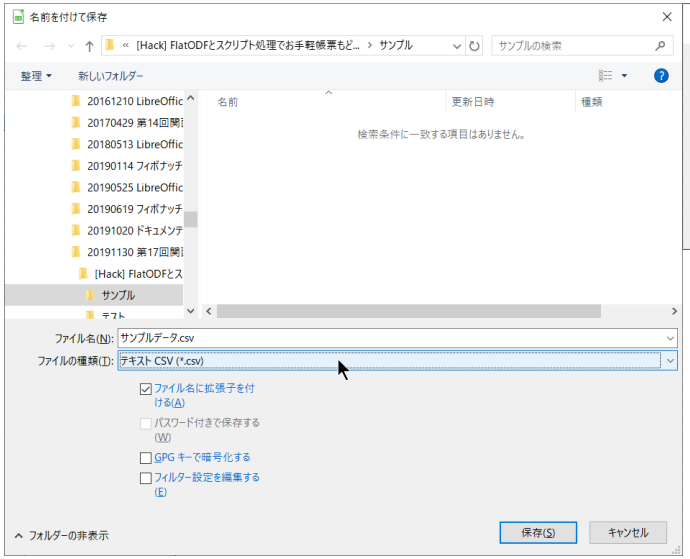
- 今回はCSV形式で用意することにしました（馴染みがあって手軽）。
  - CSV形式は例外文字（データ内の「,」や「"」や改行）の扱いに要注意。
    - 今回のサンプルスクリプトでは不精していて、それら文字を使うとバグります。
    - 真面目にやるなら各開発環境で用意されているCSVライブラリを使いましょう。
- 作成にはLibreOffice Calcを使用しました
  - CSV保存は今回は手作業で…（コマンドラインによるODS→CSV変換は動作未確認）。

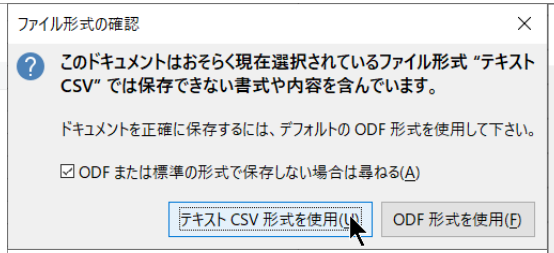


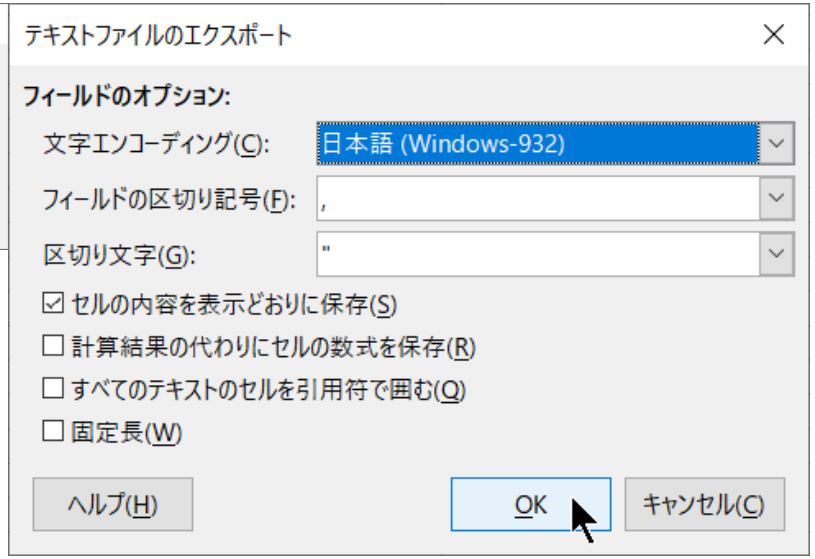
	A	B	C	D	E	F	G
1	名前	郵便番号	住所1	住所2	住所3	住所4	
2	美樹本 桃太郎	1234567	あそこの国あそこの町あそこ	１１１－１１－１	レジェンドイン １２Ｆ １２０１号室		
3	美樹本 桃太郎	1234567	あそこの国あそこの町あそこ １１１－１１－１	レジェンドイン １２Ｆ １２０１号室			
4	美樹本 桃太郎	1234567	あそこの国あそこの町あそこ 1111-11-1	レジェンドイン 12F 1201号室			
5	辻 一	0000000	あそこの国あそこの町あそこ，７－１５－３				
6	辻 一	0000000	あそこの国あそこの町あそこ〝7-15-3〟				
7							

CalcでのCSV保存

- 「名前を付けて保存」ダイアログの[ファイルの種類]で「テキスト CSV (*.csv)」を選択、[保存]。
- [ファイル形式の確認]ダイアログで[テキストCSV形式を使用]を選択。
- [テキストファイルのエクスポート]ダイアログでパラメーター設定して[OK]。
  - 今回は「日本語 (Window-932)」（スクリプト内で指定してます）「,」「"」とし、[すべてのテキストのセルを引用符で囲む]はチェックしない！（重要：前述の例外文字バグで動きません。）
    - 「,」の代わりに「{タブ}」を選べばTSV形式になります（今回の用途には本来こちらがベター）。

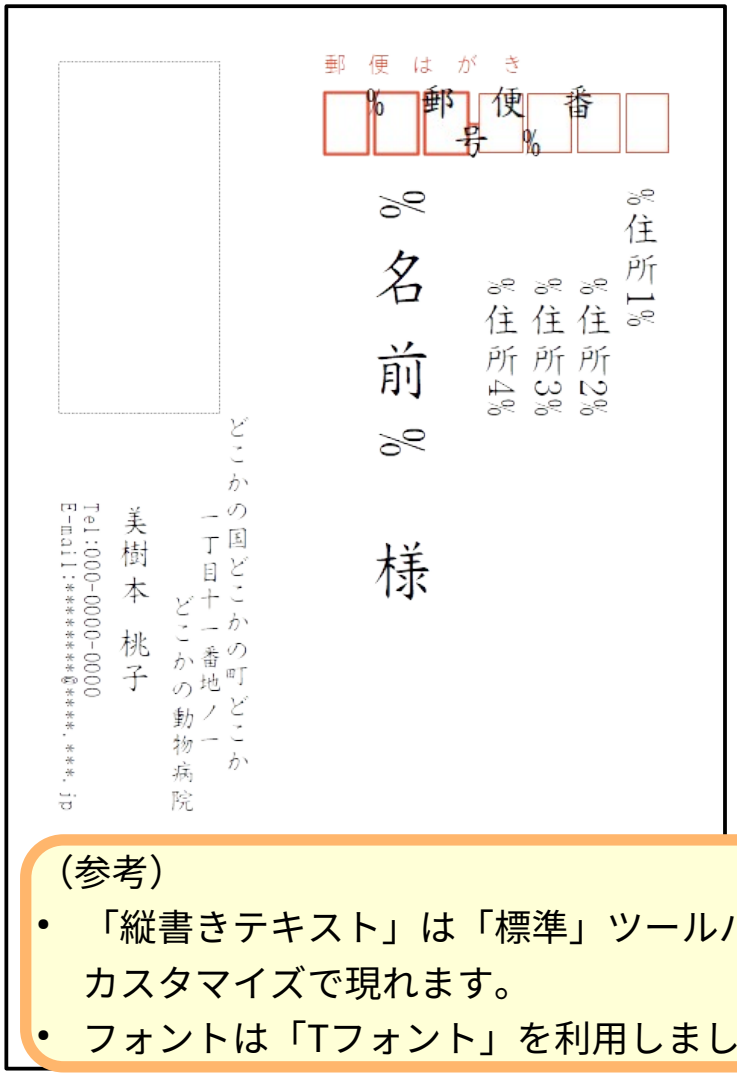






サンプルテンプレートの用意

- Drawでデザインする。
- データを差し込みたい箇所に項目指定のマーカー文字列を書く。
  - 今回は「項目名」を「%」で挟んだものにしてみました。
  - 作り方次第で工夫の余地が。ただし…
    - XMLの実体参照に置き換えられる文字（「"&'<>」など）は使えません。
    - FODFのXML構文の中に出て来るパターンも使えません。
- 「Flat XML ODF 図形描画 (*.fodg)」形式で保存（ODG→FODG変換でも可）。
- 実は今回はテキストエディタで直接FODG書いてるんですけどね…。



（参考）
- 「縦書きテキスト」は「標準」ツールバーのカスタマイズで現れます。
- フォントは「Tフォント」を利用しました。

サンプルスクリプトの用意

- 今回は報告者の得手の都合で「Tcl」言語を使います。
  - PythonやPerlと似た時代に生まれたスクリプト言語です。
  - 組み込みマクロ言語用途向けを意識して開発された超ズボラ仕様。
  - 結構パワフルで、意外とあちこちでひっそり使われてます。
  - 開発は続いていて、今大きなバージョンアップに向けた動きが。
  - 今回GUIは無用なので「Tcl/Tk」の「Tk」は使いません。
- 他言語に精通してないので断言はできませんが…
  - PythonやPerlでもたぶん似た書き方ができます。
  - Cだと文字列置換が大変ですがたぶんなんとか…。
  - 他もまぁ、頑張ればなんとかなるのでは（無責任）。

サンプルスクリプト

- 約60行。
  - コメントや補助処理も含めて。
  - 入れ子まとめはほぼしてません。
- ライブラリ不使用。
  - そのためCSV処理はザル。
- 利用しているややチートな機能。
  - 正規表現による文字列置換。
  - splitコマンド。
  - 外部プログラム実行。

  けどスクリプト言語なら、ね…。

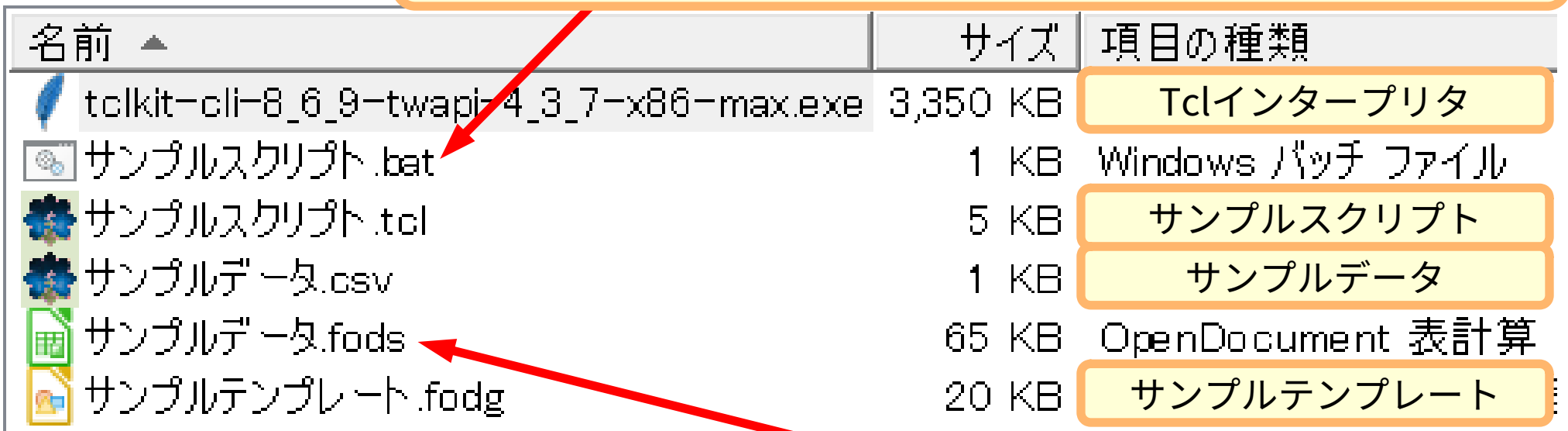
```
# 2019-11-30 第17回 関西LibreOffice勉強会
# [Hack] FlatODFとスクリプト処理でお手軽帳票もどき ～Drawデータではがきの宛名印刷～
# 岩橋 伴典（いさな）
# サンプルスクリプト

set V(list) [glob -nocomplain {出力*.*}] ;# 古い出力ファイルをリストアップして…
file delete -force {*}$V(list)           ;# まとめて削除（デモで手を抜く用）

set V(soffice) {C:¥Program Files¥LibreOffice¥program¥soffice} ;# LibreOffice実行ファイル名の設定（インストールされてると後でご利益が）

# FODGテンプレート読込
#  exec $V(soffice) --convert-to fodg {サンプルテンプレート.odg} ;# ODGファイルをLibreOfficeでコマンドライン変換してもいい
set V(channel) [open {サンプルテンプレート.fodg}]              ;# ファイルオープン
fconfigure $V(channel) -encoding utf-8                         ;# 読込の文字エンコーディングをFODF仕様にセット
set V(string.template) [read $V(channel)]                      ;# ファイル内の文字列を全て読込
close $V(channel)                                              ;# ファイルクローズ

# CSVデータ読込
#  exec $V(soffice) --convert-to csv {サンプルデータ.ods} ;# ODSをLibreOfficeでコマンドライン変換してもいい…はずなんだけどこれではダメです…
set V(channel) [open {サンプルデータ.csv}]              ;# ファイルオープン
fconfigure $V(channel) -encoding cp932                  ;# 読込の文字エンコーディングをファイルに合わせてセット
set V(string) [read $V(channel)]                        ;# ファイル内の文字列を全て読込
close $V(channel)                                       ;# ファイルクローズ

set V(list.line) [split $V(string) ¥n] ;# 読み込んだCSVデータを改行で区切ってリスト化

set V(line) [lindex $V(list.line) 0]   ;# CSVデータリストの最初のレコードを選択
set V(list.item) [split $V(line) {,}]  ;# 選択したレコードをカンマで区切って項目リスト化（CSVデータの処理としては不充分です）

# 読み込んだCSVデータリストの2つ目（インデックス 1）以降のレコードを順次処理
for {set V(i.line) 1} {$V(i.line) < [llength $V(list.line)]} {incr V(i.line)} {

  set V(line) [lindex $V(list.line) $V(i.line)] ;# レコード選択
  puts "$V(i.line)¥t$V(line)"                   ;# 無反応は寂しいので標準出力へ賑やかしの文字列出力
  set V(list.data) [split $V(line) {,}]         ;# 選択したレコードをカンマで区切ってデータリスト化（CSVデータの処理としては不充分です）
  set V(string.tmp) [join $V(list.data) {}]     ;# データリストを全部くっつけた文字列を作る
  if {$V(string.tmp) eq {}} then {continue}     ;# くっつけた文字列が空っぽなら以降の処理を飛ばして次のレコードへ
  set V(string.output) $V(string.template)      ;# 出力文字列にテンプレートをコピー

  # データの各項目について順次処理
  for {set V(i.item) 0} {$V(i.item) < [llength $V(list.item)]} {incr V(i.item)} {
    set V(item) [lindex $V(list.item) $V(i.item)]                              ;# 項目名をセット
    if {$V(item) eq {}} then {continue}                                        ;# 項目名が空っぽなら以降の処理を飛ばして次の項目へ
    set V(data) [lindex $V(list.data) $V(i.item)]                              ;# 項目名に対応するデータをセット
    set V(string.output) [regsub -all "%$V(item)%" $V(string.output) $V(data)] ;# 項目名を%で挟んだ文字列を全て検索して対応するデータに置換
  }

  # ファイル出力
  set V(filename) "出力 [format {%04d} $V(i.line)].fodg" ;# ファイル名を設定
  set V(channel) [open $V(filename) w]                   ;# ファイルオープン（書込モード）
  fconfigure $V(channel) -encoding utf-8 -translation lf ;# 書込の文字エンコーディングと改行コードをFODF仕様にセット
  puts -nonewline $V(channel) $V(string.output)          ;# 書込（最後に改行を追加しない）
  close $V(channel)                                      ;# ファイルクローズ

  # ファイル形式変換&印刷（LibreOfficeでコマンドライン変換）
  exec $V(soffice) --convert-to pdf $V(filename)                    ;# PDF変換
#  exec $V(soffice) --convert-to png $V(filename)                    ;# PNG変換
#  exec $V(soffice) --convert-to odg $V(filename)                    ;# ODG変換
#  exec $V(soffice) -p $V(filename)                                  ;# 印刷（デフォルトプリンターにて）
#  exec $V(soffice) --pt {Brother DCP-J952N Printer} $V(filename)    ;# 印刷（プリンター指定）

}
```

サンプルスクリプトの構成の概略

- 古い出力ファイルを削除（デモで手を抜く用）
- サンプルテンプレート読込
- サンプルデータ読込
- 読み込んだサンプルデータを改行で区切ってリスト化（splitコマンド）
- 最初のレコード（1行目）を「,」で区切って項目名リスト化（splitコマンド）
- 2つめ（2行目）以降のレコードを順次処理
  - レコードを「,」で区切ってデータリスト化（splitコマンド）
  - 出力文字列にサンプルテンプレートをコピー
  - 項目名リストのそれぞれについて出力文字列内のマーカー文字列を全て対応するデータに置換（正規表現による文字列置換）
  - ファイル出力。
  - LibreOfficeでコマンドライン変換や印刷（外部プログラム実行）

実行前のフォルダ状況

サンプルスクリプト起動用バッチファイル
（スクリプトをインタープリタに渡すだけ。デモを楽にする用。）



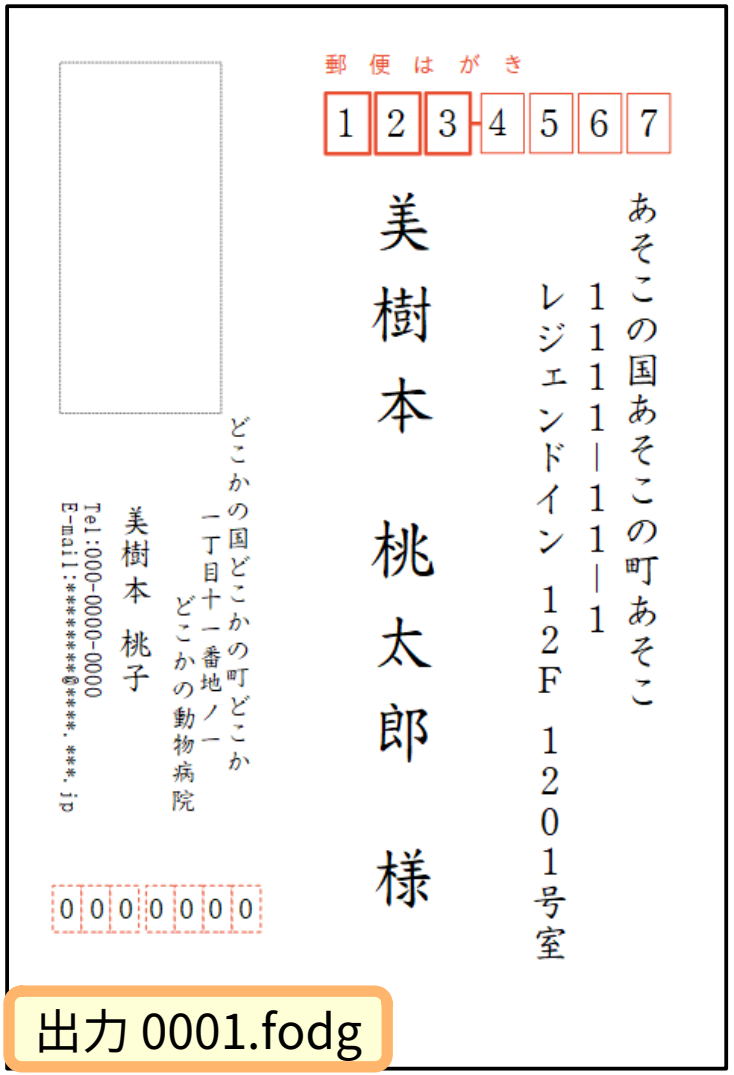
サンプルデータの
元ファイル（便宜上）

事後配布版ではファイル名など
若干構成変更しています。

デモ1 基本機能

名前	サイズ
tclkit-cli-8_6_9-twapi-4_3_7-x86-max.exe	3,350 KB
サンプルスクリプト.bat	1 KB
サンプルスクリプト.tcl	5 KB
サンプルデータ.csv	1 KB
サンプルデータ.fods	65 KB
サンプルテンプレート.fodg	20 KB
出力 0001.fodg	20 KB
出力 0002.fodg	20 KB
出力 0003.fodg	20 KB
出力 0004.fodg	20 KB
出力 0005.fodg	20 KB

出力ファイル
FODG

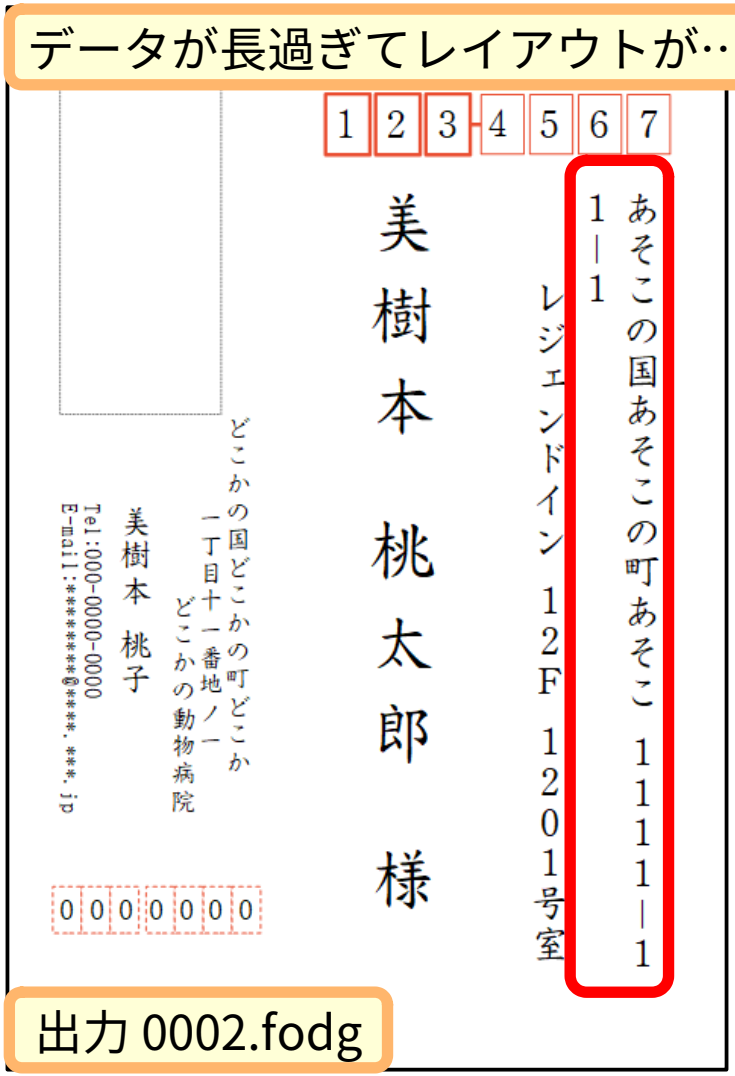


出力 0001.fodg

デモ1 基本機能 - 失敗例1

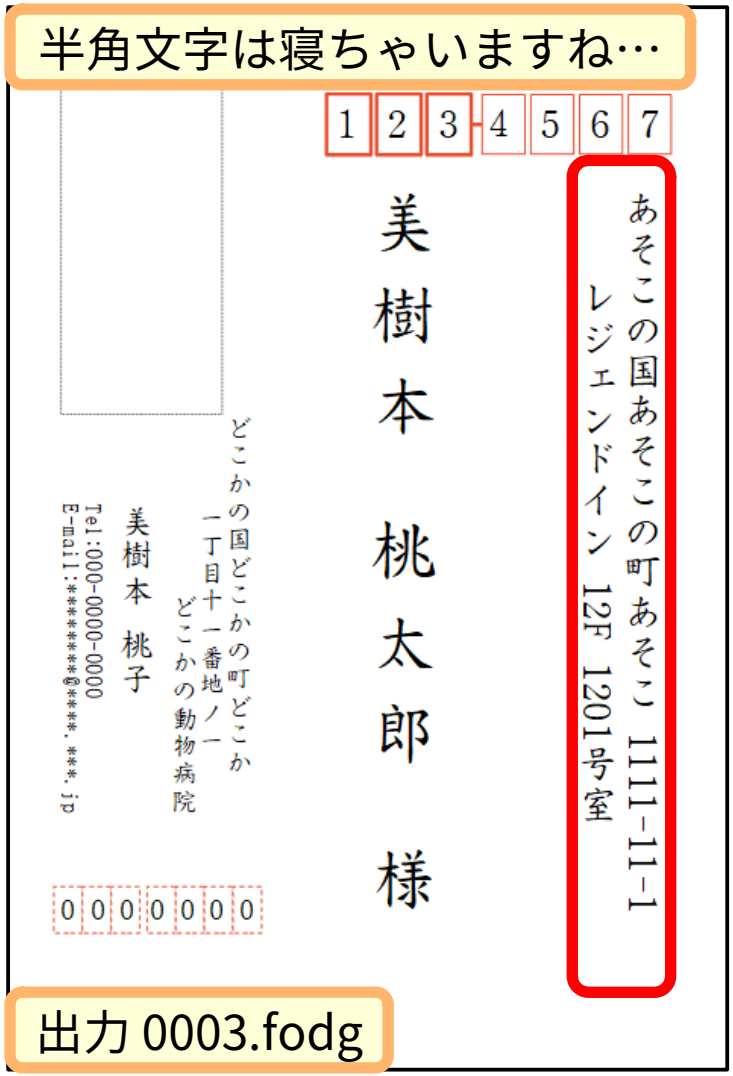
ま。今回は単純な作りなので…。

データが長過ぎてレイアウトが…



出力 0002.fodg

半角文字は寝ちゃいますね…

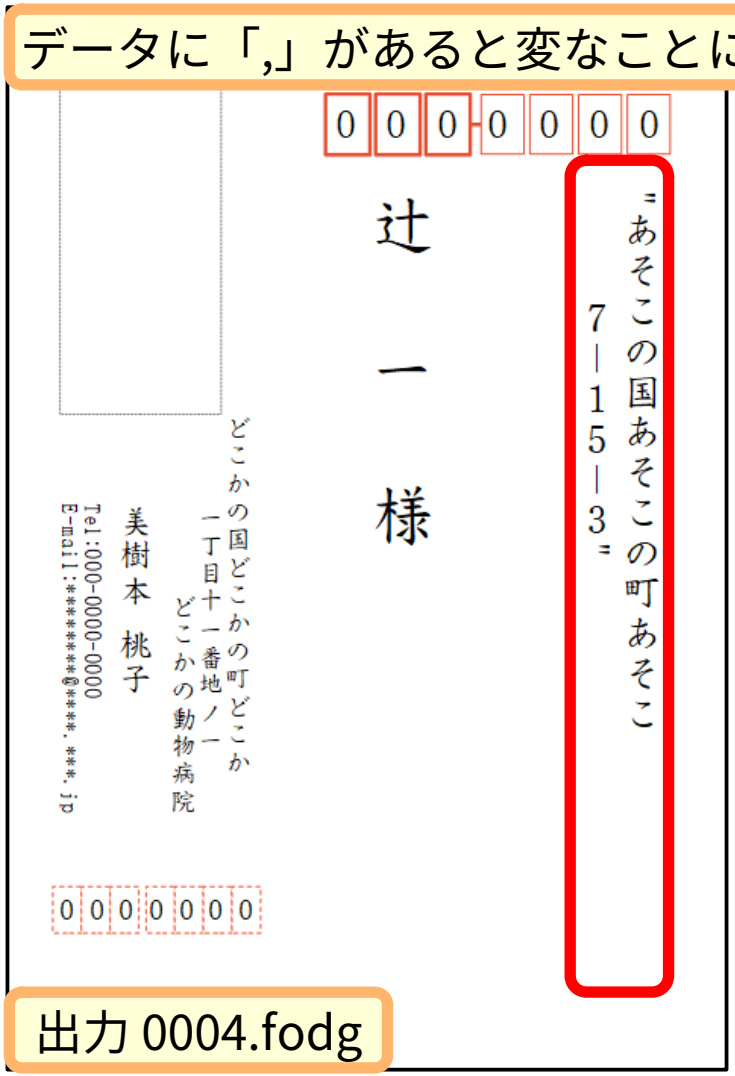


出力 0003.fodg

デモ1 基本機能 - 失敗例2

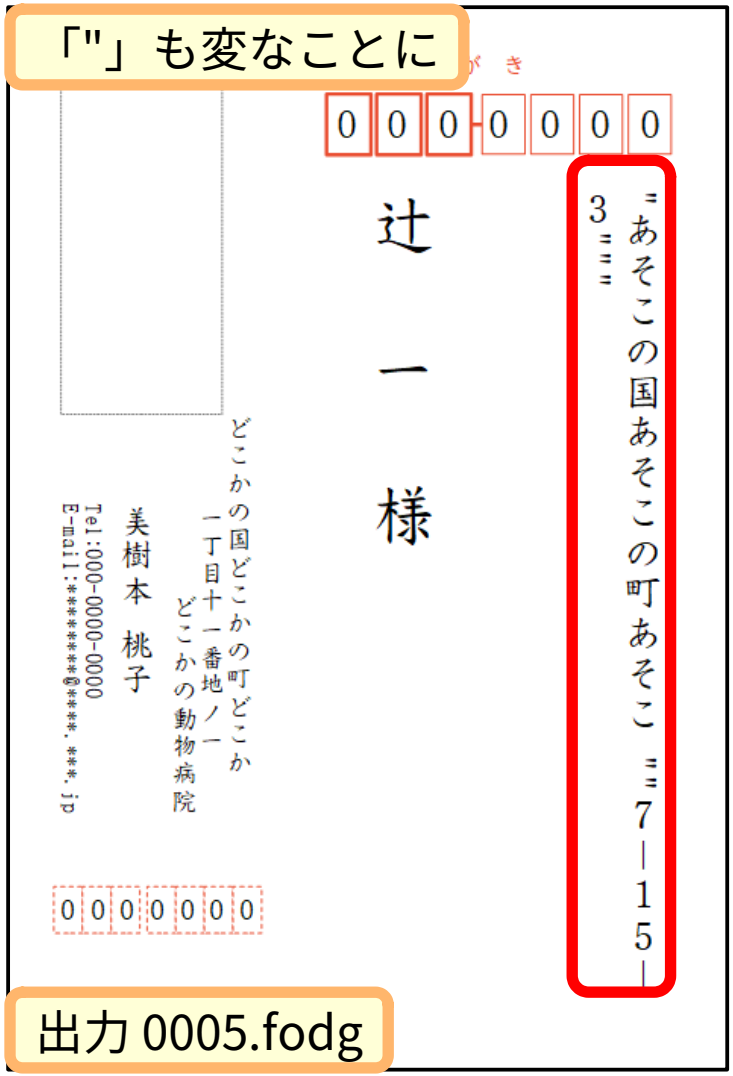
入力データ(CSV)の処理を真面目にやればこんなことには…。

データに「,」があると変なことに



出力 0004.fodg

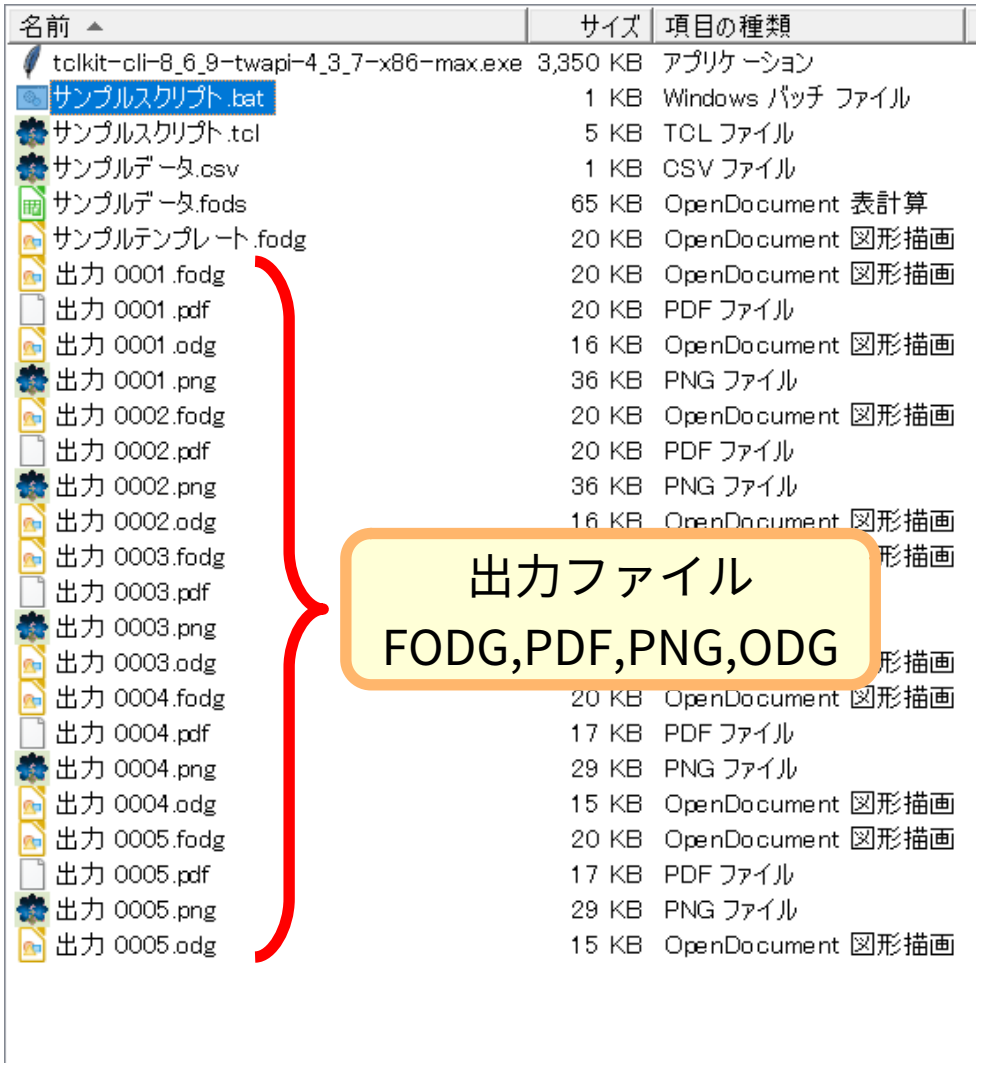
「"」も変なことに



出力 0005.fodg

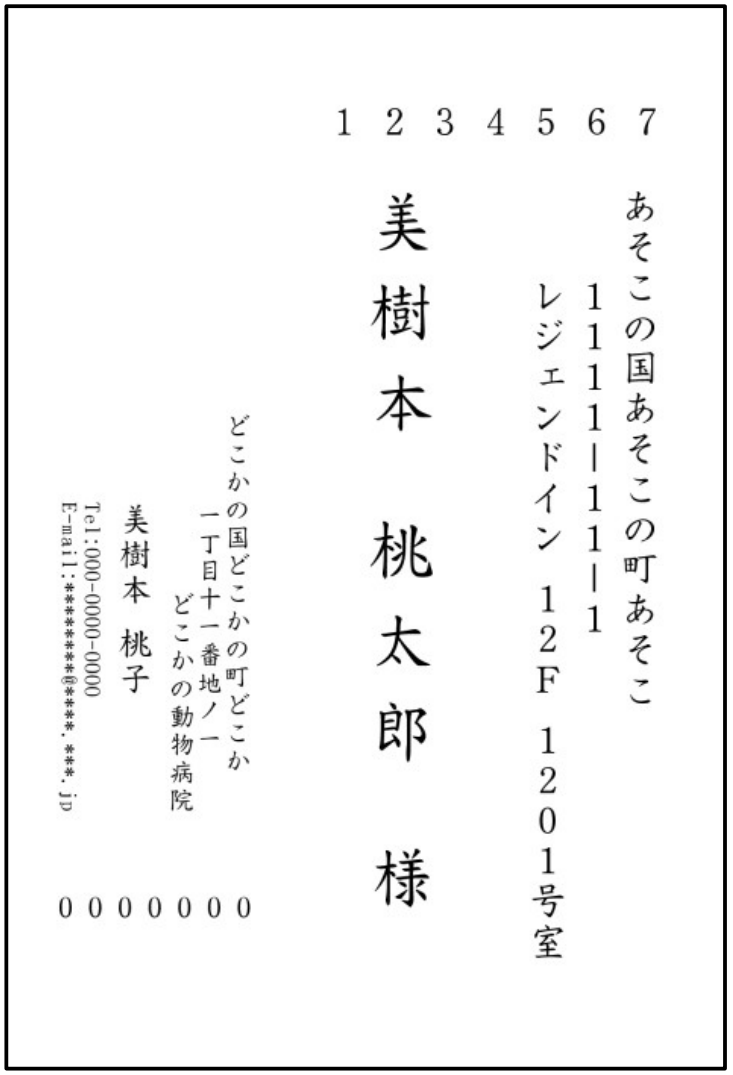
デモ2 ファイル形式変換と印刷

- スクリプトにLibreOfficeをコマンドライン呼び出しする命令を追加。
  - 各処理1行で済む。
  - --convert-toでファイル形式変換。
    exec $V(soffice) --convert-to pdf $V(filename)
    - PDF、PNG、ODGへ。
    - 処理にかなり時間がかかる。
      - レコード毎に起動しているから？
  - -pや--ptで直接プリンター印刷。
    exec $V(soffice) -p $V(filename)
    - --ptなら使用プリンターを指定できる。
    - 動作確認済み。



デモ2 ファイル形式変換と印刷 PDF&印刷では

- 今回作ったテンプレートでは郵便番号枠などを『「印刷可能」でない』別レイヤーに作成したので、PDFや印刷では出力されません。
- 一方でPNGには出力されるようです。前スライドまでの出力例は、ここで作ったPNGファイルを使ってます。



バグでは？との指摘も。本来ならばPNGでも消えるべき？

デモ3 ホントに処理してるの？

- 「デモのため単にファイルコピーしてるだけとかなんじゃないの？」なんて可能性も考えられます。
- 時間があれば…（期待薄）、新規作成のテンプレートで実演してみれるといいかなぁ…なんて…。

補足1 更なる高みを夢見て…

- テンプレートはDrawファイル限定か？
  - Writerファイル(.fodt)でもOKです（スクリプトの読込ファイル名は要変更）。
  - Calcは保持データの扱いで少々難しさがあるかも知れません。
- 文字列の差し込みしかできないのか？
  - FODFのXML構文を触ればかなりのことができます。図も描けます。
  - XMLを真面目に扱わなくても文字列置換である程度のことはできますが、高度な文字列検索テクニックが必要になってきます。
  - 今回の例でも、改行やタブ文字の挿入、文字スタイル適用などは可能です（XMLタグで置換）。
- レコード毎に別ではなく1ファイルにまとめられないのか？
  - 未検証ですが、Drawなら可能性はありそうです。
  - Writerはページの括りがデータとしてまとまってなさそうで、かなり難しそう…。
  - PDFにしてから別ツールでくっつける方が楽な気がします。

補足2 ドロー文書を帳票テンプレートに使う メリット・デメリット

- メリット
  - 要素（図形）を各々独立して自由に配置できる。
    - 方眼を細かく切らなくていい♪
  - レイアウトや描画位置を精度良く定められる。
- デメリット
  - 手作業で追加編集する場合に、操作が面倒。
    - キーボードでの要素選択（入力欄移動）が困難。
    - 要素選択後、文字入力に更にワンステップ必要。
- 自動処理にはワープロや表計算より向いてると思います。
  - 手入力の書類雛形には課題がある…。