PDFのコピペが文字化けするのはなぜか？
— CID/GIDと原ノ味フォント—

細田 真道

http://www.trueroad.jp

2021年2月26日

自己紹介

- 楽譜作成プログラム LilyPond コミッタ
  - ビルドシステム、フォント、PDF 出力等
- GNU 公式文書フォーマット Texinfo コミッタ
  - 国際化、XeTeX/LuaTeX、Unicode、日本語対応等
- 原ノ味（はらのあじ）フォント主催
  - 日本語 TeX デフォルト和文フォント（2020〜）
- 第 10 回日本 OSS 奨励賞受賞
  - LilyPond
- FIT2018 FIT 論文賞受賞

URL http://www.trueroad.jp **GitHub** trueroad
Twitter @trueroad_jp **Facebook** trueroad.jp
GPG Key fingerprint
49B8 ED79 B6A8 C46E 2F6D ABB3 FCD0 C162 1E80 A02D

はらのまち 原ノ町駅

- 2019 年には何回も行きました
  - 月刊ビジネスコミュニケーション 2020 年 9 月号 p.26〜
- 名前は似てますが原ノ味（はらのあじ）フォントとは関係ありません

はじめに

はじめに

- データ分析したいけど PDF しかない
  - テキスト抽出したら文字化けした！
- PDF を作って配りたい / PDF 帳票を生成したい
  - コピペが文字化けする！
  - コピペできない！

- これらを何とかします！

どんな文字化け？

どんな文字化け？

- 化ける文字
  - 「見」「高」「長」「玉」など
    - 比較的簡単な漢字
- 何が起きるか
  - PDF からテキストをコピペすると、似たような別の字に化ける
    - 住所を集計しようとすると、長野の「長」や埼玉の「玉」などが化けてうまくいかない
  - PDF でテキスト検索できない
- 化ける PDF と化けない PDF がある
  - 化ける PDF がたくさん出回っている
  - 本資料 PDF は化けません！（対策済み）

化けるPDFを作ってみる(1/3)

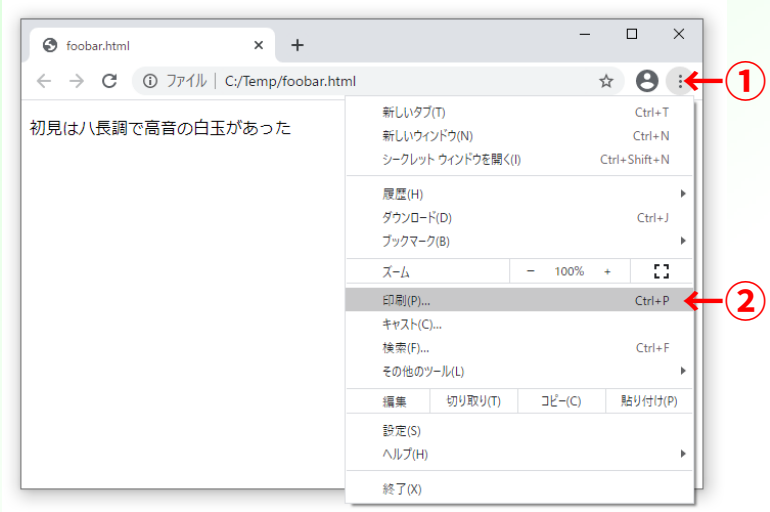
- 環境
  - Windows 10 20H2
  - Google Chrome 88.0.4324.182
  - Acrobat Reader DC 2021.001.20140
    - いずれも 2021 年 2 月現在の最新版
- 適当なHTML（文字化けしていない）

foobar.html

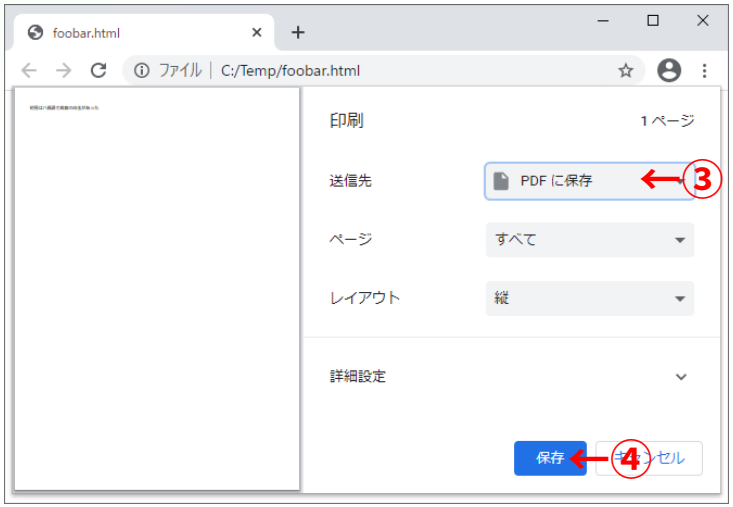
```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <p>初見はハ長調で高音の白玉があった</p>
  </body>
</html>
```

化けるPDFを作ってみる(2/3)

- HTML を Chrome で開き、①メニューを出し、②「印刷」を選択

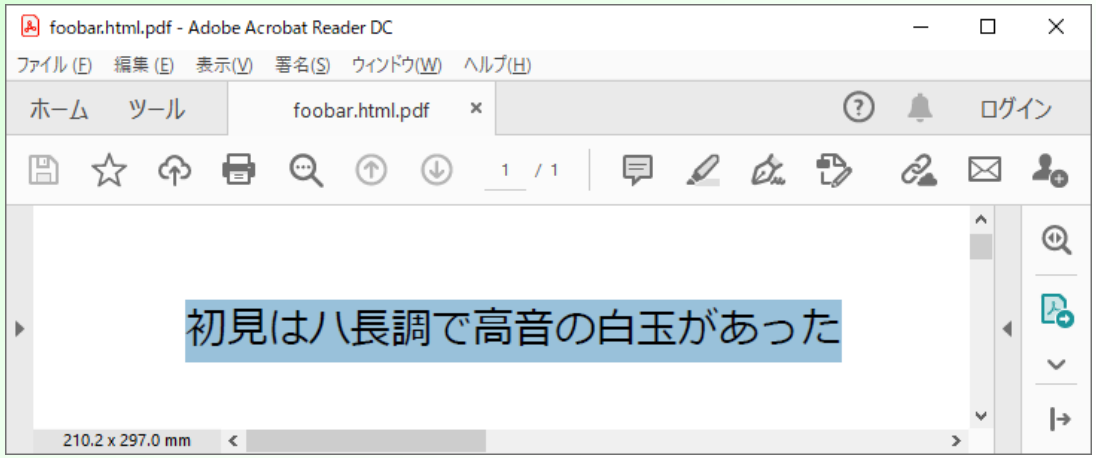


- ③送信先を「PDF に保存」にし、④「保存」ボタンを押して PDF 生成

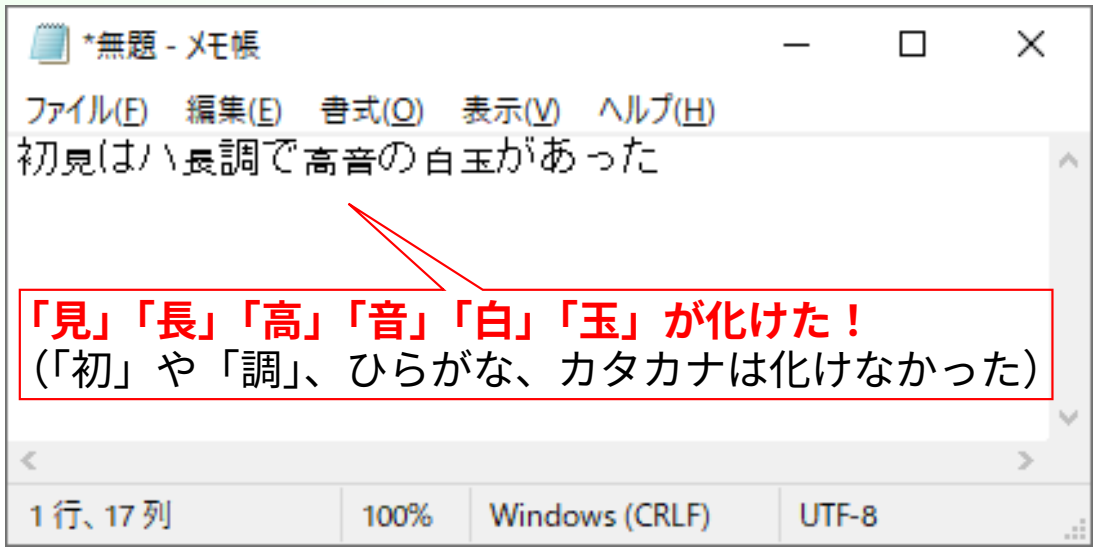


化けるPDFを作ってみる(3/3)

- PDF を Acrobat Reader で開き、Ctrl+A で全選択し、Ctrl+C でコピー



- メモ帳を開き、Ctrl+V でペースト



なぜ文字化けする？

なぜ文字化けする？

- PDF のしくみ[1]
  - PDF 内に普通のテキストは存在しない
    - どのフォントの、どのグリフ（字形）を、どこに配置するか、の情報のみ存在する
  - グリフはフォント固有の番号で指定する
    - CID/GID (Character ID/Glyph ID)[2]
    - 同じ形のグリフでもフォントによって番号が異なることがある
    - Unicode のような普通の文字コードは使われない

[1] 大多数の PDF の場合。あてはまらない PDF もあります。本ページ以降も同様です。

[2] ほぼ同じ意味ですがフォントの種類によって CID/GID どちらを使うかが異なります。

PDFの中身をのぞいてみよう

- 中身を直接見てもわからない
  - 一部が圧縮されている
- テキストエディタで見れるよう変換
  - QPDF を使う
    - Ubuntu や Fedora などパッケージあり
    - 圧縮の解除と改行やインデント等して見やすくした QDF 形式に変換[3]

```
$ qpdf --qdf foobar.html.pdf foobar.html.qdf
```

[3] QDF をテキストエディタで編集した後で PDF へ戻すコマンド fix-qdf もあります。

PDFの中身

- **BT** と **ET** で囲まれた部分を探す

foobar.html.qdf 抜粋（BT と ET で囲まれた部分）

```
BT
/P <</MCID 0 >>BDC
/F4 16 Tf
1 0 0 -1 8 33 Tm
<0a4209c4092e098109b609f9092609c80c20092d0b9c0be3090b09010922091e> Tj
EMC
ET
```

- パラメータが先、オペレータが後の後置記法
  - **Tf**: フォントを指定するオペレータ
    - “/F4” フォント（別途定義）をサイズ 16 で指定している
  - **Tm**: 座標系を指定するオペレータ
    - “1 0 0 -1 8 33” で座標系を指定している
  - **Tj**: 文字を出力するオペレータ
    - “<” と “>” で囲まれた部分で出力する文字を指定している

PDFの文字出力

foobar.html.qdf 抜粋（Tj オペレータとそのパラメータ）

```
<0a4209c4092e098109b609f9092609c80c20092d0b9c0be3090b09010922091e> Tj
```

- パラメータから 16 進数 4 桁ずつ取り出し GID[4]とする
- GID を使い “/F4” フォント（メイリオ）からグリフを取り出す

16 進数 4 桁	GID[5]	グリフ
0a42	GID+2626	初
09c4	GID+2500	見
092e	GID+2350	は
0981	GID+2433	ハ
…	…	…

[4]ここで “/F4” フォントとして使われているメイリオは GID で指定します。

[5]CID/GID は “CID+” または “GID+” に 10 進数を付けて表記します。

コピペの実現方法は？

- PDF 内部は CID/GID で文字を指定
  - フォントのグリフを指定する番号
    - 表示するには非常に都合が良い
  - Unicode などの文字コードとは関係が無い
- コピペするには文字コードが必要

コピペの実現方法は？

- PDF 内部は CID/GID で文字を指定
  - フォントのグリフを指定する番号
    - 表示するには非常に都合が良い
  - Unicode などの文字コードとは関係が無い
- コピペするには文字コードが必要

- ToUnicode CMap を使う
  - CID/GID → Unicode 符号位置の変換テーブル
  - フォント毎に用意する
    - フォントによって対応関係が異なることがある
  - コピペできる PDF には埋め込まれている

ToUnicode CMap

foobar.html.qdf 抜粋
(“/F4” フォント用 ToUnicode CMap の一部)

```
14 beginbfchar
<0901> <3042>
<090B> <304C>
<091E> <305F>
<0922> <3063>
<0926> <3067>
<0981> <30CF>
<09B6> <2ED1>
<09C4> <2F92>
<09C8> <2FBC>
<09F9> <8ABF>
<0A42> <521D>
<0B9C> <2F69>
<0BE3> <2F5F>
<0C20> <2FB3>
endbfchar
```

CID/GID

ひらがな・カタカナ
「あ」「が」「た」「っ」「で」「ハ」

CJK 部首補助「長」

康熙部首「見」

康熙部首「高」

CJK 統合漢字「調」

CJK 統合漢字「初」

康熙部首「白」

康熙部首「音」

康熙部首「玉」

- 1 行が 16 進数 4 桁表記の **変換元 CID/GID** と **変換先 Unicode 符号位置** の順番で並んでいる[6]
- 化けた文字は **変換先** が **CJK 部首補助** (U+2E??) か **康熙部首** (U+2F??) [7] で 通常の漢字ではない！
- 化けなかった文字は **変換先** が **通常の漢字**（CJK 統合漢字: U+4E00〜U+9FFF 他）か **ひらがなカタカナ**（U+30??）

[6] これとは異なる形式のこともあります。

[7] Unicode 符号位置（コードポイント）は “U+” の後に 16 進数 4〜6 桁を付けて表記します。

康熙部首・CJK部首補助

- 康熙部首（康熙部首）
  - 康熙字典（1716年）の部首
  - https://www.unicode.org/charts/PDF/U2F00.pdf
- CJK部首補助
  - 康熙字典に載っていない部首
  - https://www.unicode.org/charts/PDF/U2E80.pdf

- 通常の漢字（CJK統合漢字）と同じ形のものがある
  - 同形が異Unicode符号位置にある
  - これらを取り違えると文字化けが発生

なぜ取り違えるか？

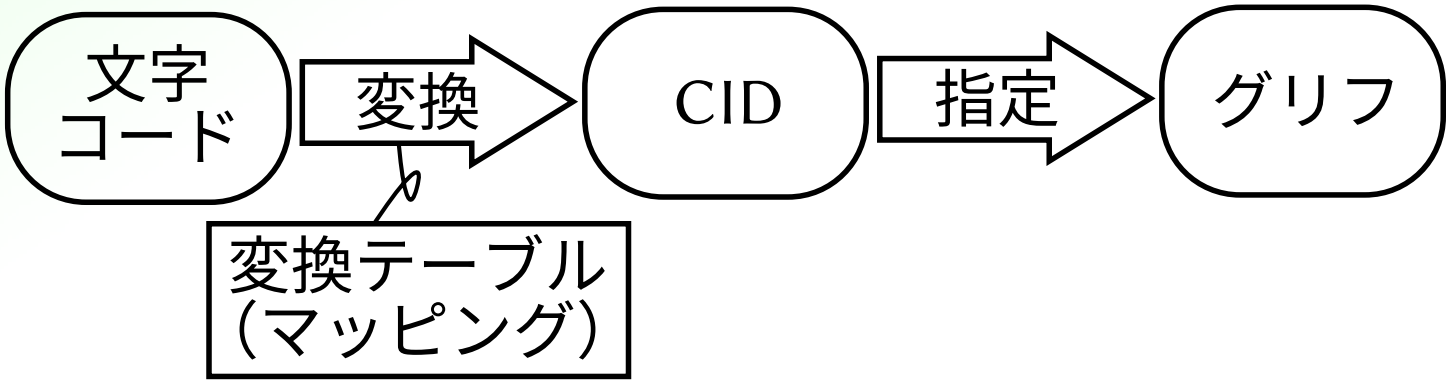
- 問題のある ToUnicode CMap を作るのは PDF 作成ツール
  - つまり PDF 作成ツールに問題がある[8]
  - 広く使われていても問題あるツールがある
- フォントにも依存する
  - 問題のある PDF 作成ツールで使うと、トリガーを引いてしまうフォント[9]とそうでないフォントがある
  - 最近のフォントの多くはトリガーになる

[8] PostScript 経由のフロー等では PDF 作成ツールの前工程に問題がある場合もあります。

[9] フォント規格としては問題なく、フォントが悪いわけではありません。

PDFの作られ方

- テキストをPDF化するときは
  1. 文字コード (Unicode) を CID/GID へ変換
     - OpenType フォントに内蔵されている cmap テーブルを使う
  2. CID/GID でグリフを指定
     - Tj オペレータのパラメータに CID/GID をセット



cmap テーブル

- トリガーになるフォントの例（メイリオ）

変換元 Unicode 符号位置	変換先 GID
…	…
U+2F92	GID+2500
…	…
U+898B	GID+2500
…	…

康熙部首「見」

CJK 統合漢字「見」

GID が同じ！

- トリガーにならないフォントの例（MS ゴシック）

変換元 Unicode 符号位置	変換先 GID
…	…
U+898B	GID+12386
…	…

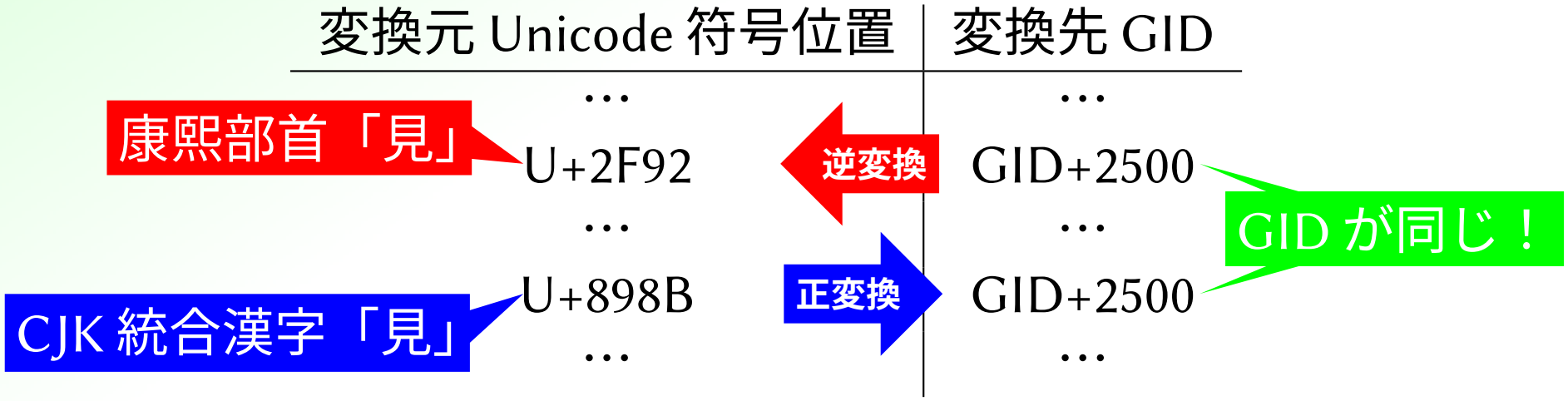
康熙部首「見」無し！

CJK 統合漢字「見」

同じ GID が他に無い

文字化け発生メカニズム

- 問題がある PDF 作成ツールの ToUnicode CMap 生成方法[10]
  - cmap 正変換後に変換元 Unicode 情報を捨てている
    - 変換前が **U+898B CJK 統合漢字「見」** だったことを忘れる
  - 変換先 CID/GID から cmap テーブルの逆変換で生成
    - GID+2500 から逆変換しようとする
  - cmap テーブルが n 対 1 対応なのを考慮していない
    - 番号が小さくて先に見つかる **U+2F92 康熙部首「見」** を採用
- これにより **U+898B が U+2F92 に化けてしまう！**



[10] あくまでも推定ですが、昔の XeTeX (xdvipdfmx) はこれで化けました。

文字化けを修正するには？

文字化けを修正するには？

- 誰かからもらったPDFが文字化けするときどうすればよいか
  - データ分析したい
    - テキスト抽出してからデータ分析したい
  - PDFビュワー上で検索したい
    - PDFのまま必要なフレーズを探したい

正規化する

- テキスト抽出したら
  「康熙部首ブロック」「CJK 部首補助ブロック」
  の文字を対応する
  「CJK 統合漢字ブロック」
  の文字へ置き換える
- データ分析用ならこれで十分では？
  - 他の正規化も必要になることが多いはず
    - 全角半角の統一
    - 丸数字を丸のない数字へ統一
    - 等々

ToUnicode CMap書き換え

- かなり荒業ではあるが
  PDF 内部の ToUnicode CMap を
  書き換える方法もある
  - 拙作ツール pdf-fix-tuc
    `https://github.com/trueroad/pdf-fix-tuc`
    - ToUnicode CMap の康熙部首などを
      CJK 統合漢字へ書き換えるツール
- PDF ビュワーで検索も
  できるようになる

文字化けしないPDFを作るには？

トリガーを引かないフォントだけ使う

- お勧めはしない
  - 古いフォントが多い
    - IPA 明朝/IPA ゴシック/
      MS 明朝/MS ゴシック etc.
  - 最近のフォントはトリガーを引く
    - IPAex 明朝/IPAex ゴシック/メイリオ/游 etc.
  - ライセンスも心配
    - MS 明朝/MS ゴシックは Windows の一部
      （Linux など他の OS で使える？）
    - アップデートで仕様が変わる可能性も？
  - 解決になっていない

対策済みPDF作成ツールを使う

- 最新の日本語 TEX は対策済み
  - LuaTEX-ja/pTEX+dvipdfmx etc.
    - Ubuntu や Fedora などパッケージがあり、インストールも簡単
  - サーバサイドでの PDF 帳票や報告書の自動生成にも使えそう
  - 定型フォーマットに文章や数字を流し込んで PDF を作るのは得意
    - この資料も TEX で作成
- 他のツールは？
- どんな対策が必要？

どんな対策？

- CJKの複雑な事情を考慮する必要がある
  - 欧米基準で何も考えないと化けてしまう
- 文字化け発生メカニズムをつぶす
  - 正変換後も変換元 Unicode を覚えておく
  - cmap テーブルが n 対 1 なのを考慮する
  - もう一つの対策も…
- どれかができれば対策 OK
  - 日本語 TEX の対策例を紹介します
  - 問題のある OSS があるなら
    ぜひ貢献してください！

LuaTeX の対策

- 変換元 Unicode 符号位置を覚えておく
  - 極めて正攻法、一番よい方法
  - ただし、アーキテクチャ的に
    元の Unicode 符号位置が失われてしまう
    フロー[11]にせざるを得ない場合は使えない

[11] CID/GID しか格納されていない中間形式 (PostScript や XeTeX の xdv 等) を経由するなど。

pTEX+dvipdfmx/XETEX の対策

- cmap テーブル n 対 1 対応の考慮
  - 康熙部首ブロックなどの優先順位を下げる
  - 複数の Unicode 符号位置候補がある場合は優先順位が高いものを使う
    - 大抵の場合で使えるが GSUB テーブルのグリフ置き換え[12]などがあると破綻も

[12] 横書き用「𪜈」を vert フィーチャで縦書き用「龕」へ置き換えるなど。

pTEX+dvipdfmx/XETEX の対策

- cmap テーブル n 対 1 対応の考慮
  - 康熙部首ブロックなどの優先順位を下げる
  - 複数の Unicode 符号位置候補がある場合は優先順位が高いものを使う
    - 大抵の場合で使えるが GSUB テーブルのグリフ置き換え[12]などがあると破綻も
- ToUnicode CMap を生成しない **New!!**
  - Adobe-Japan1 規格などの CID-keyed フォントの場合に限る
  - フォントは限定されるもののデメリットがほとんどない

[12] 横書き用「𠮷」を vert フィーチャで縦書き用「𠮟」へ置き換えるなど。

Adobe-Japan1 (AJ1)

- 日本語で使われる様々な文字を集め、識別のため CID を割り当てたもの
  - AJ1 規格のフォント同士は CID が同じなら同じ文字を表す
    - 「見」は必ず CID+1887
- 初版の AJ1-0 は 1992 年制定
- 最新の AJ1-7 は 2019 年制定
  - OpenType フォント規格（初版 1997 年）より古い歴史がある
    - OpenType の cmap テーブルに依存せず文字コードから CID を得る機構がある

文字コードを CID/GID へ変換

- 現代的な機構（前述の方法）
  - OpenType の cmap テーブルを使う
  - フォント毎に CID/GID が違ってよい
    - AJ1 でもよい
  - LuaTEX/XƎTEX などのモダン TEX や普通の Windows アプリなどで使われる
- OpenType 登場前の機構
  - AJ1 専用の外部テーブルを使う
    - CMap リソース
  - すべての日本語フォントは CID が同一
    - AJ1 でなければならない
  - pTEX など伝統的な日本語 TEX で使われる
    - まだバリバリ現役で広く使われている

AJ1 を PDF からコピペする

- AJ1 規格のフォントなら
  共通の ToUnicode CMap が使える
  - フォント個別に用意する必要が無い
  - 多くの PDF ビュワーが自分で持っている
    - PDF 内に埋め込む必要なし
  - コピペに最適な調整済になっている
    - 康熙部首文字化けしない
    - GSUB 置き換えにも対応
    - 等々
  - どんな PDF でもコピペできる
    - ToUnicode CMap を生成できない
      PDF 作成ツールでも問題なし

AJ1 フォント

- AJ1 フォントのメリット
  - pTEX の和文フォントは AJ1 が最適
  - ToUnicode CMap 不要でコピペ可能
  - 等々
- しかし、
  - 実用的な AJ1 フォントは
    非フリーだけだった
- 無いなら作る！
  - 原ノ味フォントを制作することに

はらのあじ
原ノ味フォント

はらのあじ
原ノ味フォント

- 明朝・ゴシックそれぞれ 7 ウェイト全 14 フォント

原ノ味明朝 ExtraLight	原ノ味角ゴシック ExtraLight
原ノ味明朝 Light	原ノ味角ゴシック Light
原ノ味明朝 Regular	原ノ味角ゴシック Normal
原ノ味明朝 Medium	原ノ味角ゴシック Regular
原ノ味明朝 SemiBold	原ノ味角ゴシック Medium
原ノ味明朝 Bold	原ノ味角ゴシック Bold
原ノ味明朝 Heavy	原ノ味角ゴシック Heavy

- AJ1-6 漢字グリフすべて搭載（JIS X 0208/JIS X 0213 全漢字グリフ含む）
- JIS X 0208 グリフすべて搭載（漢字・非漢字・横書き・縦書きすべて含む）

原ノ味フォント採用例

- 日本語 TeX デフォルト和文フォント
  - TeX Live 2020 から採用
- 原ノ味フォントを全面的に使用した書籍多数
  - iOS テスト全書（PEAKS、2019 年 12 月）
  - 機械学習 100 ＋ページ エッセンス
    （インプレス、2019 年 12 月）
  - ［改訂第 8 版］LaTeX2e 美文書作成入門
    （技術評論社、2020 年 11 月）
    - 付録に「原ノ味フォント全グリフ」収録
  - その他 TeX 組版で出版される書籍の多くが採用
- この資料も原ノ味フォント使用！

原ノ味フォントのグリフ

- 商業出版に耐えうる品質の
  大量のグリフ（字形）を
  個人で作れるわけがないですよね

原ノ味フォントのグリフ

- 商業出版に耐えうる品質の
  大量のグリフ（字形）を
  個人で作れるわけがないですよね

- 巨人の肩に立つ
  - Adobe のオープンソースフォント
    源ノ明朝/源ノ角ゴシック[13]
    のグリフを利用させていただいてます

[13] Google の Noto Serif CJK/Noto Sans CJK とほぼ同じフォントです。

源ノ明朝/源ノ角ゴシック

- 豊富なウェイト、グリフ
  - 明朝・ゴシック 7 ウェイト全 14 フォント
  - AJ1-6 全漢字グリフ搭載
- Adobe-Identity0 (AI0) フォント
  - 非 AJ1 フォント
    - 日本語用ではなく汎 CJK 用
    - CID がバラバラ、明朝とゴシックでも違う
    - 現代的で OpenType が前提
- オープンソースフォント
  - SIL Open Font License 1.1 (OFL-1.1)
    - 一定の条件下で改変や再配布可能
- これを AJ1 に組み替える！

生成プログラム

- 全自動で生成
  - GUI での手動調整は一切なし
  - プログラム化できないことはしない
- OSS として公開
  - https://github.com/trueroad/HaranoAjiFonts-generator
  - 派生フォントとしては珍しいのでは？
    - フリーフォントの派生フォントは
      ライセンス上フリーフォントになるが、
      生成プログラムを公開しているものは
      ほとんど無さそう
  - 自分で原ノ味フォントを生成できる
  - OSS だからカスタマイズもできる

OSSにしてよかったこと(1/3)

- 位置調整が必要なグリフがあった
  - ギリシャ文字・キリル文字等
  - 源ノと AJ1 規格では字幅が異なる
    - 源ノはプロポーショナル幅（グリフ依存）
    - AJ1 は全角幅
  - 字幅を広げただけでは
    グリフが左に寄って不格好に
- 平行移動させるのは大変
  - グリフ内部形式 (CFF Charstring) が複雑
- 平行移動・拡大縮小できるコードを
  Pull request していただけた
  - 不格好なグリフがなくなった！

OSSにしてよかったこと(2/3)

- 縦書きグリフに不足があった
  - JIS X 0208 で 4 グリフだけ不足
    - 横書き用「‖」「゜」「′」「″」は存在
- 90 度回転＋平行移動できれば作れそう
- いただいた平行移動コードをベースに 90 度回転コードを実装！
- 縦書きグリフ不足解消
  - JIS X 0208 全グリフ搭載を実現！
    - 縦書き用「═」「 ｡」「 ､」「 〃」を追加

OSSにしてよかったこと(3/3)

- プロポーショナルかなグリフ
  - 搭載要望と実装アイディアをいただいた
- いただいたアイディアを実装！
  - 全角かな（字幅が同じ）
    - いろはにほへとちりぬるを
  - プロポーショナルかな（字幅が異なる）
    - いろはにほへとちりぬるを
- その他のかなグリフにも応用
  - 縦書きプロポーショナルかな、組方向最適化かなの搭載につながる
    - かなグリフが充実！

おわりに

おわりに

- どんな文字化け？
- なぜ文字化けする？
- 文字化けを修正するには？
- 文字化けしないPDFを作るには？
- 原ノ味フォント

関連資料

- 本資料
  - ソースファイルや関連資料を公開します
    - https://github.com/trueroad/tr-NTTtech05
  - 不十分なところや間違いなどあれば
    ご連絡ください
- より詳しく知りたいなら
  - TeXConf 2019 一般講演
    「原ノ味フォントと ToUnicode CMap」
    - https://github.com/trueroad/tr-TeXConf2019