実践 ファイル拡張子

@3846masa

「Word ファイルの拡張子を変えると PDF になる」は本当に嘘なのか

「Word ファイルの拡張子を `.pdf` に変えただけファイルが送られてきた」このようなとき、あなたはどのように説明しますか。「ファイル拡張子」とは何でしょうか。
ファイル拡張子は、「該当ファイルをどのように扱ってほしいかプログラムに示す文字列」です。あくまで「どのように扱ってほしいか示す」だけであり、**ファイル拡張子を変えてもファイル自体は変わりません。**

最初の問いに戻ります。おそらく多くの人は、「拡張子は単なるファイル名の一部であって、Word ファイルの拡張子を変えても PDF にはならない」と説明するでしょう。しかし、本当に実現できないのでしょうか。

この『実践 ファイル拡張子』では、Word ファイルと PDF を例に、「ファイル拡張子を変えたらファイルが変わる」ように見せかける方法を紹介します。

「ファイル拡張子を変えたらファイルが変わる」ように見せかける

例えば、次のようなプレーンテキストについて考えてみましょう。

```
print("Hello World!");
```

このプレーンテキストのファイル拡張子が `.rb` であれば、Ruby のプログラムコードであると解釈できます。また、`.pl` であれば Perl として、`.py` であれば Python としても解釈できます。**なぜならば、Ruby としても Perl としても Python としても、正しいコードになっているからです。**

さらにこれを PHP で動かすとすると、どうすればよいでしょうか。PHP ファイルは、コードの開始位置に <?php タグが必要です。他の言語では <?php タグを解釈できないため、うまく PHP だけで <?php タグを読み込ませる必要があります。
もっとも簡単な解決方法は、コメントを活用することです。次のように、<?php タグをコメントとして書いてみます。出力結果は若干違いますが、 Ruby / Perl / Python / PHP で動くコードになります。

```
# <?php
print("Hello World!");
```

ここまでをまとめると、**複数の解釈ができるファイルを作れると、「拡張子を変えるとファイルが変わる」ようにみえそうです。**複数のファイルフォーマットを組み合わせるには、 一方のファイルフォーマットを保ちつつ、もう一方をうまく埋め込む必要があります。

次の章からは、実際に「拡張子を変えると PDF になる Word ファイル」を作ってみます。

拡張子を変えると PDF になる Word ファイルを作る

PDF の仕様と抜け道

PDF は、%PDF-M.n のようなバージョン指定から始まり、%%EOF で終わります。ヘッダーの仕様をもう少し見ていきましょう。
PDF の仕様は Adobe のサイト[^adobe-site] から閲覧することができます。今回は version 1.7 のもの[^adobe-pdf] を参照していきます。

仕様にある APPENDIX H の H.3 Implementation Notes には、実装についての話が書いてあります。H.3.13 ではヘッダーに関する次の文章が掲載されています。

```
Acrobat viewers require only that the header appear somewhere within
the first 1024 bytes of the file.
```

意訳すると、「先頭から 1024 バイトにヘッダーがあるようにすること」になります。実際に、 Mozilla の開発する PDF.js[^pdfjs] の実装を見ると、先頭 1024 バイトの中で `%PDF-` の文字列を検索しています。[^pdfjs-github]
つまり、**Word ファイルの先頭 1024 bytes 以内に PDF のヘッダーがあれば、PDF として有効なファイルになります。**この抜け道を使って、Word ファイルに PDF を埋め込みましょう。

Word ファイルに別のファイルを埋め込む

Office Open XML フォーマットは、いわゆる Office 系ソフトウェアにおける、標準フォーマットとして制定されています。 いわゆる Word ファイルは、Office Open XML フォーマットで作られた文書ファイルのことを指します。

このフォーマットの特徴は、文書を構成するファイル群を ZIP フォーマットでまとめている点です。**すなわち、Word ファイルは拡張子を `.zip` にすると ZIP ファイルとして扱えます。**

```
# .docx を .zip に変える
$ mv document.docx document.zip
# unzip コマンドで展開する
$ unzip document.zip -d extract
```

ZIP 形式であれば、Word ファイルのなかに PDF を埋め込むことができそうです。しかし、フォーマットに沿わないファイルを含むと、Word で開くときに読み込みエラーが発生してしまいます。

```
# 適当なファイルを入れて zip コマンドで圧縮する
$ touch extract/empty.bin
$ (cd extract; zip ../modified -r .)
# .zip を .docx に変える
$ mv modified.zip modified.docx
```

Word ファイルに他のファイルを埋め込むにはどうすればよいでしょうか。ひとつの解決策が、"Object Linking and Embedding (OLE)" という仕組みです。**OLE を使うことで Word ファイルとして正しい状態のまま、任意のファイルを埋め込めます。**

試しに PDF が OLE で埋め込まれた Word ファイルを作ってみましょう。Windows に標準搭載されているワードパットを使うと、簡単にファイル埋め込みを実現できます。「オブジェクトの挿入」から「新規作成」「パッケージ」を選択して、埋め込みたいファイルを選びます。埋め込んだファイルは、`word/embeddings/oleObject1.bin` にあります。

Compound File Binary File Format

OLE で埋め込まれたファイルは、Compound File Binary File Format (CFB)[^ms-cfb] という形で埋め込まれます。

CFB は、512 bytes の Sector が繰り返されて構成されます。先頭の 512 bytes は Compound File Header で固定されています。前述のとおり、PDF のヘッダーを先頭 1024 bytes 以内に含める必要があります。**Header の次にある Sector 0 の 512 bytes に PDF のヘッダーが含まれないと、1024 bytes を超えてしまいます。**

File Allocation Table (FAT) Sector や Directory Sector では、「どの Sector に何が格納されているか」を定義できます。これを活用すると、**任意の位置に好きなデータを格納することが可能です。**

Compound File Header

Compound File Header には、最初の Directory Sector の位置や FAT Sector の数、位置などが格納されています。PDF を最初の Sector にするためには、Directory Sector や FAT Sector を PDF ファイルのあとに持ってくる必要があります。

Sector は 512 bytes ごとに区切られているため、`ceil(PDF のファイルサイズ / 512 bytes)` が PDF の格納に必要な Sector の数です。そして、Sector 0 に PDF を格納するならば、最初の Directory Sector や FAT Sector の位置は、PDF の格納に必要な Sector の数だけ後ろの Sector になる計算ができます。
FAT Sector では、各 Sector がどう繋がっているか、1 Sector 4 bytes で表現します。Sector は 512 bytes で構成されているため、ひとつの FAT Sector で 512 / 4

= 128 個の Sector について表現できます。よって、FAT Sector の数は `ceil(全体の Sector の数 / 128)` として計算できます。
全体の Sector の数は、PDF を格納する Sector の数、Director Sector の数、FAT Sector の数を足したものになります。ここまでの情報をもとに Compound File Header を作ります。例えば、32 KB の PDF を格納する Compound File Header は次のようになります。

```
0x0000 : D0 CF 11 E0 A1 B1 1A E1 00 00 00 00 00 00 00 00
0x0010 : 00 00 00 00 00 00 00 00 3E 00 03 00 FE FF 09 00
0x0020 : 06 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00
0x0030 : 40 00 00 00 00 00 00 00 00 10 00 00 FE FF FF FF
0x0040 : 00 00 00 00 FE FF FF FF 00 00 00 00 41 00 00 00
0x0050 : FF FF FF FF FF FF ..
```

最初の Directory Sector の位置は 0x30 - 0x33 に、FAT Sector の数は 0x2C - 0x2F、FAT Sector の位置は 0x4C 以降に 4 bytes ごと格納します。

OLE Packager Data Format

Compound File Header の次の Sector には、PDF を OLE Package として格納します。OLE Packager Data Format[^ole-package] にはファイル名やファイルパスを記述できますが、今回は先頭 1024 bytes 以内に PDF を入れることを優先します。64 KB の PDF を格納する最小限の OLE Package は次のとおりになります。

```
0x0200 : 1E 0F 00 00 02 00 00 00 00 00 03 00 01 00 00 00
0x0210 : 00 00 F0 00 00 25 50 44 46 ..
......
0xF210 : 25 45 4F 46 A0 00 00 00 00 00 00 00 00 00 00 00
0xF220 : 00 ..
```

Directory Sector

CFB はデータを階層構造で持っています。最上位のディレクトリは Root Directory で、OLE ではその下に `¥x01Ole10Native` Entry があります。
`¥x01Ole10Native` Entry は、さきほどの "OLE Packager Data" に該当するため、Sector Id も最初の Sector 0 になります。

File Allocation Table (FAT) Sector

FAT Sector は、4 bytes ずつの値で Sector の連続性を表現します。Sector の終わりは、0xFFFFFFFE を書き、FAT Sector は 0xFFFFFFFD を書きます。Sector が存在しない、もしくは使われていない場合は、0xFFFFFFFF を書きます。例えば、2KB の PDF を最初の Sector に持ち、Directory Sector と FAT Sector が続く場合は、このような記述になります。

```
0xNN00 : 01 00 00 00 02 00 00 00 03 00 00 00 FE FF FF FF
0xNN10 : FE FF FF FF FD FF FF FF FF FF FF FF FF FF FF FF
```

ZIP ファイルと圧縮アルゴリズム

oleObject1.bin は、前述の方法でファイルを作れば、おおよそ 533 byte の位置に PDF ヘッダーがきます。oleObject1.bin は Office Open XML フォーマットの一部であるため、今度は Office Open XML フォーマット内部の oleObject1.bin の配置について考えます。
まず、oleObject1.bin が PDF としてそのまま読み込めるようにしなければなりません。Office Open XML フォーマットは ZIP ファイルであるため、そのままでは圧縮されてしまい、そのままのデータとして読み込めません。**そこで、ZIP ファイルでは、任意の圧縮アルゴリズムが使えるため、非圧縮（store）アルゴリズムを使うようにします。**zip コマンドでは、-0 オプションで非圧縮で格納できます。
また、先頭 1024 bytes に oleObject1.bin 内の PDF が含まれるようにしたいため、ZIP ファイルを作るときは最初に oleObject1.bin を入れるようにします。ZIP ファイルのヘッダーは数十 bytes しかないため、おおよそ 620 byte あたりに PDF ヘッダーがくるようなファイルが作れます。**これで、Word ファイルと PDF が 1 ファイルに合わさったファイルが作れました。**

```
# 最初に oleObject1.bin を非圧縮で格納する
$ (cd extract; ¥
    zip ../modified -0 'word/embeddings/oleObject1.bin')
# oleObject1.bin 以外のファイルを格納する
$ (cd extract; ¥
    zip ../modified -r . -x 'word/embeddings/oleObject1.bin')
```

まとめ

Word ファイルと PDF のどちらでも認識できるファイルは、次のようにして作成できます。

1. PDF が最初の Sector になっている `oleObject1.bin` を作る
2. `oleObject1.bin` が非圧縮で先頭ファイルになっている Word ファイルを作る
3. Word ファイルとして開きたい場合は、拡張子を `.docx` にする
4. PDF として開きたい場合は、拡張子を `.pdf` にする

実際に試したところ、拡張子を `.docx` にすれば、Microsoft Word で問題なく開くことができました。また、拡張子を `.pdf` にしたら、Microsoft Edge や Google Chrome、macOS のプレビューなどで開くことができました。しかし、残念ながら Acrobat Reader ではセキュリティ上の制約^acrobat-security によって、PDF ヘッダーが先頭にない場合にエラーになるため、開くことができませんでした。
Adobe 公式のツールで開けないですが、Windows で PDF を開く標準ソフトである Microsoft Edge や、macOS 標準のプレビューで開くことができたため、成功といえそうです。**結論、「ファイル拡張子を変えたらファイルが変わる」Word ファイルと PDF が技術的には作れることがわかりました。** ただし、PDF ビュアー依存を考えると**実用性には乏しいのでお蔵入り**な技術でした。

Word ファイルと PDF を 1 つのファイルにまとめるコードと、実際に Word ファイルと PDF を一緒にしたファイルを O'CREILLY GitHub Orgs^ocreilly-github に公開予定です。みなさんも試してみてください。