電子データの紙面化について

高坂宜宏 *

Digital Data Stored on Paper

Yoshihiro TAKASAKA

Abstract- A file server at the worksite was damaged last year and data that had been stored there over approximately 15 years were lost. The server administrator had claimed that even if the server is damaged, the data contained would not disappear because the same data are stored in each RAID. However, as the two hard discs were simultaneously damaged, data restoration was impossible. Magnetic discs and CD-ROMs may loose data over time due to the deterioration of the devices. To prevent data loss, a measure for recording digital data on paper, which last for several thousands of years, is considered.

keywords: DigitalData, paper

1 はじめに

昨年，職場のファイルサーバが破損し１５年程の間に職場で作成した書類データやソースデータが消えてしまった。当初はもちろん不測の事態に備えデータのバックアップをとり，あちらこちらのコンピュータなどで保存していた。年がたつにつれバックアップしていたサーバーが壊れたり，別の保存していたサーバーが壊れたりして，それを繰り返しているうちにいつのまにかこのファイルサーバだけにデータが残っている状況となっていた。日頃からデータが無くなっては困ると 気にはしていたが，サーバの管理者はレイドで保存しているため故障してもデータが消えることは無いと言っていたので安心していた。また，このサーバーには長年のデータが保存してあったためファイルが膨大で，CD-ROM などで自分のデータを保存することは容易ではなかった。今回のファイルサーバーの破損は，予期せぬ停電とサーバの老朽化によりディスクが２台同時に破損したため復旧が不可能となってしまったのである。

私にとってこのデータの保存方法を考えることは永久のテーマである。国家公務員試験の際の技術試験の時も「永久に消滅しないメディア」について書いた。

現在普及している磁気ディスクや CDROM などは，どうしても時期が経過するとディスクが劣化し，データが消えたり，読み取りにくくなってしまう可能性がある。そこで，二度とデータが消滅することの無いように，何千年も残る「紙」で記録を残すことを考えた。

*釧路高専教育研究支援センター 電気・電子・情報グループ

2 システム構成

紙面化をするにあたり，作業の手順を図１にしめす。電子データをテキストコード化する「エンコード部」，次にアスキーコード化する「アスキーコード部」, 印刷するために符号化データに変換する「符号化データ部」からなる。また，紙面より電子データに戻す場合には，「カラー調整部」と「ビット調整部」と「文字コード部」，「デコード部」からなる。

2.1 エンコード部

ここでは，Linux に用意されているコマンド「uuencode」を使用し電子データをテキストコード化している。このコマンドを使用することにより，バイナリーデータもテキスト化して扱うことができる。

2.2 アスキーコード部

印刷するためにデータを符号化するための準備としてアスキー文字を 16 進数に置き換えている。ここで

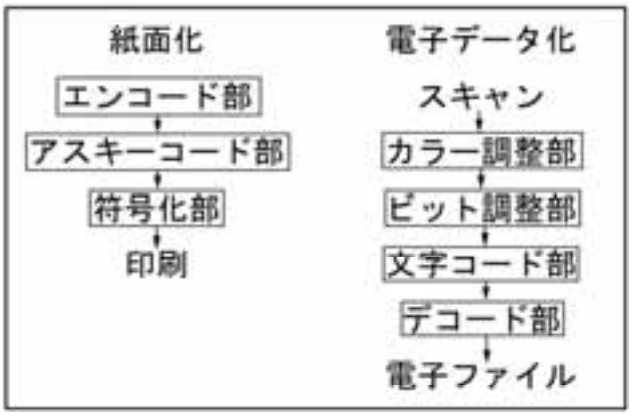

図 1: システム構成

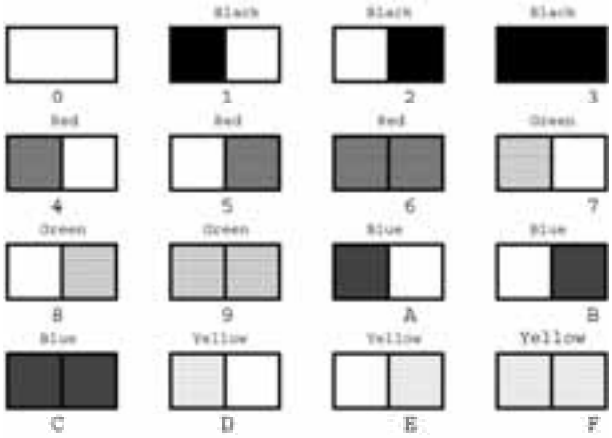

図 2: 符号化するためのカラーコード

は，C 言語でコマンドを作成し，一文字ずつ読み込み 16 進数へと変換している。

たとえば「b」ならば「62」,「4」なら「34」，空白なら「20」となる。

2.3 符号化データ部

16 進数をそれぞれ符号化する。ここでは図 2 のように 2 進数の符号を黒, 赤, 緑, 青, 黄色で 16 進数を表す。符号化するときのデータ数を減らすために「色」を使用することにし，黒, 赤, 緑, 青, 黄色を使用し 16 進数を表すことにした。ここでも C 言語でコマンドを作成し，XPM データに変換している。このデータを印刷すると紙面データが印刷される。

16 進数で「626567696e203634」とある場合には色情報に変換すると「RR.KRR.RRRG.RRGGRR.Y.K..KKRRKKR.」となる。ここで「K」は黒,「R」は赤,「G」は緑,「B」は青,「Y」は黄色,「.」は 0 をあらわしている。

2.4 カラー調整部

スキャンしたデータを XPM で保存し，そのデータを読み込み複数の色を，必要な黒, 赤, 緑, 青, 黄色のみの色にしぼりこむ。ここでは，Perl[1] で作成し XPM のカラー情報を読み込み色の指定を白, 黒, 赤, 緑, 青, 黄色のみの色にする。また，データもそれらの色に変更する。

2.5 ビット調整部

ひとつの符号データに複数のビットが存在するので，前後左右のデータを検索し，ひとつのデータにまとめている。ここも Perl で作成し，データを一度収納してまわりの色を確認してデータをとりだしている。

2.6 文字コード部

符号データから 16 進数に変換し文字コードにする。ここでは C 言語で作成している。次のデコードをするために文字コードへと変換している。

2.7 デコード部

最後に「uudecode」を使用してデータを復調し元のデータファイルへ変換する。

3 電子データの紙面化の実行

今回は，図 3 にしめす C 言語の簡単なソースファイル「a.c」を紙面化することにした。まず,「a.c」を「エンコード部」で変換し，図 4 のようにテキストコード化する。次に，データを紙面化する際に符号化しやすいように数値に置き換えるため「アスキーコード部」を実行する。この結果は図 5 のようになる。最後に「符号化データ部」を実行すると，図 6 のようになり，図 7 にしめすような紙面に印刷できるデータとなる。

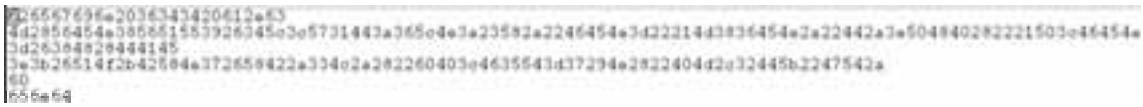

図 5: 数値化されたコード

図 6: 符号化したコード

```
#include<stdio.h>

int main()
{
   printf("Hello...\n");
   return (-1);
}
```

図 3: 変換するソースファイル

```
begin 644 a.c
M(VENBVQU9&4\<W1D:6\N:#X*"FEN="!M86EN*"D*>PH@("!P<FEN=&8H(DAE
>;&QD+BXN7&XB*3L*("`@<F5T=7)N("@M,2D["GT*
`
end
```

図 4: テキストコード化したファイル

4　紙面化されたデータの読み込み

まず，紙面化された電子データをスキャナーで読み込むために，スキャナーに印刷された紙面化データをセットし該当する部分をスキャンする。図 8 はスキャン後のデータの一部である。これは XPM のファイルデータであるが，ファイルのカラーの表現が 2 文字になっていたり色の深さが深くなっていて複雑になっている。これを簡単にするために「カラー調整部」を実行し，図 9 のように紙面化した場合と同じような簡素な XPM ファイルへと変換する。ここでは，データを読み取れる形式に変更し指定した色に統合して中間色を無くしている。今回は手動でパラメータを調整した。続いて「ビット調整部」を実行し，1 データあたりのデータを複数のドットから 1 ドットにしている。今回は，この部分は手動でパラメータを変化させて調整した。さらに「文字コード部」でアスキー文字からなるテキストコードに変換し「デコード部」を実行しもとの「a.c」を作成する。

5　問題点

電子データを紙面化することを安易に考えたが，なかなか思うようにいかなかった。現在のところ，電子データに戻す場合にまだ手間がかかり 100%元に戻るとはいえない。今のところ電子データを「紙面化データ」に変換することについては大きな問題はないが，紙面化データを「電子データ」に戻すにはいろいろな問題が多い。

まず一つ目には，符号化に色を使用したことである。スキャナーで読み込んだときに思いがけない中間色が現れ，色を検索するときに不都合が生じてしまうことである。この問題を解決するためには，繰り返し色々なデータや用紙，さらには複数のスキャナー装置を使い，「色」の特性を調べることが必要だと感じた。また，符号化する時の色の数を少なくし，読み取るときの簡

図 7: 紙面化データ

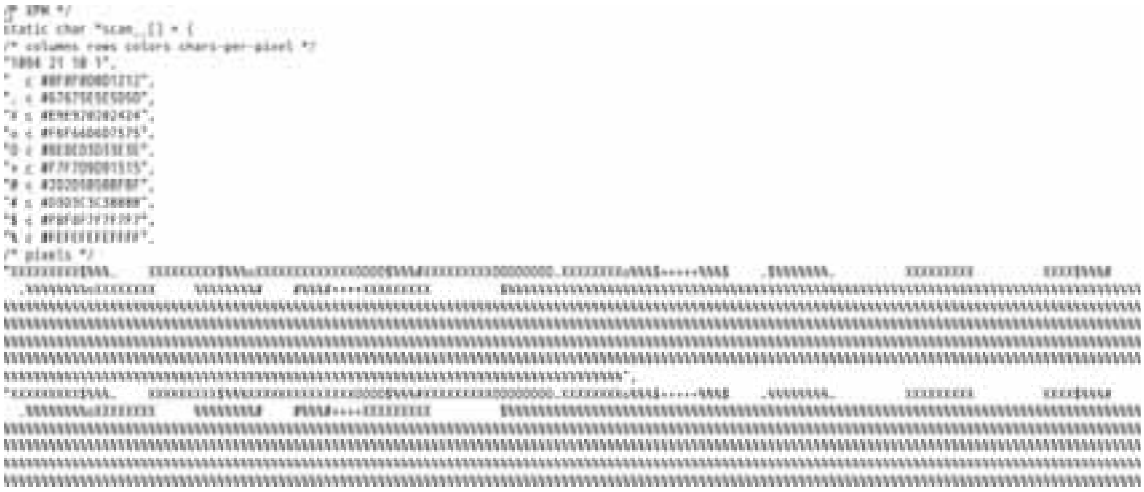

図 8: スキャン後のデータ

素化をはかる必要がある。今後時間を作り是非 調べてみたい。

二つ目には，紙面化データを読み込む解像度である。良い結果を得るには高解像度でスキャンするべきなのか低解像度でスキャンすべきなのかが まだ はっきりしないことである。高解像度でスキャンすると鮮明で良い画像として紙面化データを読み込めそうだがいざデータを解析しようとすると，本来のデータでは無い細かいゴミのデータまで読み取ってしまうのである。これはスキャンしたときのデータ変換で生じるものだと思われる。このゴミデータがはっきり存在すると解析時点でどのデータが本来のデータなのか判断が難しくなってしまう。また，低解像度でとると良いような気がしてくるが，紙面化した時とまったく同じ解像度なら問題ないが，少しでもずれると必要なデータが読み取れなかったり，周りのデータと混じってしまい，解読不可能なデータになってしまうおそれがある。この解像度の問題も経験値だと思うので納得のいくようにいろいろな解像度で今後 試してみたい。

三つ目には, いかに符号化されたビットを認識するかである。現時点では手動でおこなっているが今後は，自動的にビットの大きさを判断し，正確な色が読み取れるようにしていきたい。

四つ目には，紙面化データとして印刷するときの鮮明さである。いかにはっきりと印刷するかでもちろんスキャナーでの読み込みの良し悪しも決まってくるので，このへんも工夫していきたい。

その他にも，容量の大きなデータなどにもどのように対応していくかなど改良点が多いので今後もこの研究を続けていきたい。

6　むすび

今回の「電子データの紙面化」は，始めたばかりの研究で手探り状態であったため，動作させることで精一杯であった。このシステムの実行についても個々に作成したコマンドを手入力して実行し出力ファイルを確認後さらにつぎのコマンドを実行するような形にとどまった。さらに，紙面化データを読み込む時には手動でデータ変換をしたり，パラメータを調整するなどマンパワーによるところが多かった。

今後は，コマンドひとつで電子データの紙面化をしたり 紙面化データを読み取ったりできるようにしていきたい。最後に，現在のシステムでは煩雑でとても使えたものではないので日々改良し，将来的には一般的に使用できるようにしたい。

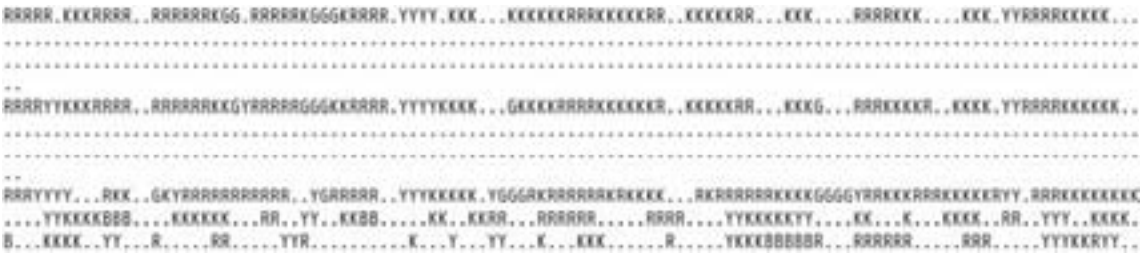

図 9: カラー調整後のデータ

参考文献

[1] Larry Wall and Randal L. Schwartz,”Perl プログラミング”, ソフトバンク株式会社,1995.06.05