情報処理 II 第 9 回

情報の電子化 (4) 画像、音声

桂田 祐史（かつらだ まさし）

2001 年 6 月 21 日

1 連絡事項

配布したプリントの印刷用データを用意するようにした。情報処理 II の WWW ページから、「印刷用 PS ファイル」を入手して lp コマンドでプリンターに送れば印刷できる。

第 8 回のプリント jouhousyori2-2001-08.ps をプリンター w1 で印刷

```
waltz21% lp -dw1 jouhousyori2-2001-08.ps
```

PDF 形式のデータも用意してあるので、 Acrobat Reader があれば、表示 & 印刷ができる。

2 前回までの説明の補足

2.1 最近の外部記憶媒体の容量

- (最近はあまり使われなくなってきたが) フロッピーディスクは、 2DD が 720KB, 2HD が 1.44MB
- CD-ROM (CD-R, CD-RW なども同様) は約 600MB
- DVD-ROM は CD-ROM の約 7 倍 (片面シングル・レイヤーの場合) 4.7GB (?)
- パソコン用ハードディスクは 20 ~ 30GB 程度のものが低価格 (2 万円程度) で普及

2.2 テキスト・ファイルのサイズ

(口頭では説明したことだが。)

通常、日本語は 1 文字 2 バイトで表現されている。だから日本語のテキスト・ファイルのサイズは大ざっぱに言って、文字数× 2 バイトと考えれば良い (実際には空白や改行、また英数字など 1 バイトで表現される文字などもあるので、あくまでも概算である)。

文書の文字数を概算するのは簡単だから (一行○文字、一ページ○行、○ページ)、文書の文字情報をテキスト・ファイルにした際のサイズの概算ができることになる。

3　レポート課題 5

以下のことを調べよ。〆切は 6 月末日とする。(ファイルのサイズについての感覚を身につけるのが主旨。)

(1) WS 上のファイルのサイズについて調べよ。バイト数以外に、フロッピーや CD-ROM にどれくらい入るかを記せ。

　(a) 文書ファイル。
　レポート、メイル、 C プログラム、 TEX のソース (`.tex`) など。
　(印刷して何ページくらいの文書が何バイトになるか。)

　(b) 実行可能プログラム
　例えば C プログラムはコンパイル前と後でどうサイズが変わるか。
　自分が普段使っているプログラム[1]をいくつか選び、そのプログラム・ファイルのサイズを調べよ？

　(c) 現在、自分が持っているファイルの総容量。

```
du -k ~
```

(結果の単位は KB である。)

(2) 自分が持っている本を一冊選び、その文字情報を記憶するファイルを作った場合、サイズはどれくらいになるか計算せよ。フロッピー・ディスクに記憶する場合、何枚必要か？また CD-ROM (容量 650MB 程度) には何冊分記憶できるか。

(3) (もし出来れば) 画像ファイル、音声ファイルなど。
(記録の形式、画像の場合は図の大きさ & 色数、音声の場合は時間等も分かる範囲で調べる。)

4　マルチメディア情報の電子化

文書はテキスト・ファイルとして電子化できることを詳しく見てきたが (なお、付録 A にも目を通して欲しい)、画像や音声についてはどうだろうか？

4.1　画像の電子化

コンピューターのディスプレイは、色 (明暗込み) をつけた点 (ピクセルあるいはドット) の集まりとして構成されている (カラー・ディスプレイ上の 1 ピクセルは R, G, B (red, green, blue) に光る 3 つの小さな点で構成されることが多い)。各ピクセルの色情報を数値化して、それを集めることによって画像情報をデジタル・データとして表現できることになる。

例えば、実習に用いているワークステーションのディスプレイ画面は、縦横約 1000 ピクセルずつ[2]、つまりおよそ 100 万ピクセルからなっていて、各ピクセルは 256 の色がつけられる。

[1] コマンドの実体がどこにあるか (パス名) は、例えば which コマンドを使って調べられる。前回のプリントあるいは `http://www.isc.meiji.ac.jp/%7Ere00018/syori2/search-for-command/` を見よ。

[2] `xdpyinfo` コマンドを実行すると調べられる。例えば、この文章を書いているワークステーションの画面は、横 1152, 縦 900 ピクセル。

ピクセル数は最近のワークステーション、パソコンでは大体この程度である (やや高級なもので 1600×1200 程度)。色数は 256 色は現在では最低線で、これは 1 ピクセルの色を 8 ビットで表わすからで、 16 ビットの場合は 65536 色、 24 ビットの場合は約 1600 万色となる[3]。

この素朴な方法で 1 画面分の画像を記録したファイルのサイズはいくらになるか計算してみよう。すぐに分かることは

画像を記録したファイルはサイズが大きい。

そこで、

画像の記録にはデータの圧縮が不可欠

になる。デジタル・データの圧縮には、

(1) 可逆な圧縮　オリジナルのデータが完全に復元できる圧縮法[4]

(2) 不可逆な圧縮　オリジナルのデータが完全には復元できない圧縮法 — うまくやると圧縮後のデータのサイズを非常に小さくすることができる

の二つがある。画像の場合は、オリジナルと多少違っていても、十分役立つことが多いため、不可逆な圧縮が採用されることが多い。色々な圧縮法があり、画像データのフォーマットも実に多くの種類がある。

画像データは不可逆な圧縮をすることでサイズをかなり小さくできる。

画像圧縮技術は特許などがからんで「取り扱い注意」の部分がある。

例. WWW ページでも良く使われている GIF 画像は、 UNISYS という企業が特許を持っている圧縮法 (LZW) を利用することが多い。 UNISYS と契約した会社の作成したソフトで GIF 画像を作成する場合は問題ないが、そうでない場合は特許料を支払う必要がある。 (大した問題ではないと思われるかもしれないが、例えばフリーソフトは閉め出されることになる。今のところ、データを再生するプレーヤー (画像の場合はビューアー？) の開発を制限しているケースはないようだが[5]、万一プレーヤーも制限されるようになると、「作成したデータは本当に作成者のものなのか？」心配になってくる (筆者の杞憂であれば良いのだが)。)

[3] $2^8 = 256,\ 2^{16} = 65536 \fallingdotseq 6万5千,\ 2^{24} = 16777216 \fallingdotseq 1600万.$

[4] 例えば、文書データやプログラム・データなどは可逆な圧縮をする。 gzip, compress, zip, LHa などは可逆な圧縮をする。

[5] あるいは私が知らないだけか？少し脱線かも知れないが、今でも DVD の再生には制限がある。 (今のところ) 一般ユーザーがデータを作ることがないせいもあって、それほど問題はないのかもしれないが...

X や OpenWindows が動いている場合に試せる実験

```
oyabun% xdpyinfo                                         (結果は省略)
oyabun% xwd > image.xwd                                  →この後マウスの入力待ちになる。
                                                           ウィンドウを選択して左ボタンを
                                                           クリックする。
oyabun% ls -l image.xwd                                  ← サイズを見てみる。
(省略)                                                   → ルートウィンドウを選択した場合
                                                           全画面が記録されて約 1 MB
oyabun% xwud -in image.xwd                               ← 試しに画面に表示してみよう。
oyabun% cat image.xwd | xwdtopnm | pnmtopng > image.png  ← PNG というフォーマットに変換
oyabun% ls -l image.png                                  ← サイズを見てみる。
-rw-r--r-- 1 mk lab00 42181 6月 15日 08:57 image.png     → かなり小さくなった！
oyabun% rm image.xwd                                     ← ディスクの無駄使いなので削除
```

(情報科学センターは pnmtopng を用意していない。前回のように source ~re00018/syori2c としてから試してみよう。あるいは GIF 形式に変換する ppmtogif を試してみよう。)

これは動画では特に顕著である。テレビなどでは、1 秒間に約 30 の画像を次々に映すことで動画を表現している。

ワークステーションの画像をテレビのように 1 秒間に約 30 回描き換えるとして、1 時間にどれだけの量のデータが必要となるか？

4.2 音声の電子化

音声は適当な時間間隔で圧力変化を記録することで電子化されるのが基本である。

音楽用 CD では、44.1kHz のサンプリング・レートで[6]、圧力を 16 ビット (2 バイト) の数値で (普通は 2 チャンネルのステレオで) 記録する。1 分間、ステレオで録音されたデータの容量は、

$$44.1 \times 10^3 \times 2 \times 2 \times 60\mathrm{B} = 10584000\mathrm{B} \fallingdotseq 10.1\mathrm{MB}$$

もの巨大なサイズになる。

音声データに関しても、画像データと同様のことが言える。

音声データも不可逆な圧縮をすることでサイズをかなり小さくできる。

音声データを圧縮する方法も特許がからむことが多い。

例. mp3 (MPEG 1 Audio Layer-3) という圧縮技術では、44.1kHz, 16bit ステレオで、約 1/10 程度のサイズ (つまり 1 分 1 MB 程度) に圧縮できる (例えば 3 分 44.39 秒の曲が 3,590,559 バイト)。圧縮・復元の方法は公開されているが、実際に圧縮するプログラム (エンコーダー) は特許料を払わずに作成することはできない (？)。

[6] 要するに 1 秒間に 44.1 × 1000 回測定する。人間は通常 20kHz までの音を聴くことができると言われていて、20kHz の音を記録するには最低でも倍のサンプリング・レートが必要なことから、この程度の値を採用した (らしい)。

A　データ・フォーマットの選択についての注意 — 文書を例として

文書を電子化するための形式 (データ・フォーマット) として、テキスト・ファイルを紹介したが、それは非常に単純なもので、文書の中にどういう文字がどういう順番で現れるかだけしか表現できない。

一方で世の中で実際に使われている (印刷された、あるいは手で書かれた) 文書には、レイアウトや、文字の大きさ、色なども重要な要素である場合がある。さらには図、写真などが含まれる文書も多い。これらの文書を電子化するには、テキスト・ファイルでは不十分であることは明らかであろう。

そのため、このような文書を扱うソフト (ワードプロセッサー、DTP ソフト、WWW ブラウザー、TEX のような文書整形システム, etc.) では、独自のデータ・フォーマットが採用されている。

独自形式のデータ・フォーマットで電子化された文書は、それを読むためにもそれに対応したソフトを必要とする。そのため次のような問題が生じうる。

- データを送った相手が対応したソフトを持っていないかもしれない。
- そもそもコンピューター環境によってはソフトが存在しないこともある。
- 現在はソフトがあっても、将来はそうでないかもしれない[7] — 保存の問題。
  それほどではなくても、ソフトの細かいバージョンの違いによるトラブルは日常茶飯事である。
- 辛うじて読めるにしても、柔軟な利用が困難なことがある。
  (例えば文書の中の単語の検索などが簡単に出来るか？よそのソフトのデータに簡単に取り込めるか？)

自分が入手・作成するデータのフォーマットとして何を選択するか、慎重に考える必要がときどき生じる。

ソフトを作るのは企業である場合が多く、企業の利益を守るために、データのフォーマットを非公開にしたりするなど、必ずしもユーザーの利益優先では考えてくれないことを肝に命じておこう。

きれいに見栄え良く表示・印刷できることは気持の良いことで、時には最も重要なことであるが (少し考えれば例はいくらでも見つかる)、ものによっては、多くの人が手軽に利用できること、また長期にわたって保存できることの方が重要な場合もある。そういうわけでテキスト・ファイルもかなり役立つことがある[8]。

[7] 例えば、私の最初の二つの論文はいずれもコンピューターを用いて書いたが、現在はそれをコンピューターの画面で読んだり、印刷したりするのは難しい。部屋の片隅でほこりをかぶっているパソコンを動かせば読めるかもしれないが、そのソフトが対応しているプリンターで、今でもちゃんと動くものは果たして、このキャンパスの中に存在するだろうか？

[8] 実際、私の最初の論文は、テキスト・ファイルとしても保存しておいたので、数式以外の文章の部分は今でも読むことができる。その手の「抜け道」については配慮することが重要である。

B　データの圧縮用コマンド

UNIX では、可逆な圧縮をするための一般的なコマンドとして、compress, gzip, bzip2 などがある。compress は以前はよく使われたが、やはり特許の問題であまり使われなくなってきた。最近は gzip が良く使われている。

```
waltz12% cd filter
waltz12% cp alice29.txt alice29.doc
waltz12% gzip alice29.doc              ← 圧縮する
waltz12% ls -l alice*                  → alice29.doc の代りに alice29.doc.gz という
                                         小さなファイルが出来る
waltz12% wc alice*                     ← ファイルのサイズを調べる。
waltz12% gzip -d alice29.doc.gz        → 復元する。
waltz12% ls -l alice*                  → alice29.doc.gz が無くなり、alice29.doc が現れる。
waltz12% diff alice29.doc alice29.txt  ← 変化がないかチェックする (変化はないはず)。
```

英文テキスト・ファイルなどでは 1/3 程度までにサイズが小さくなる。また `gzip -d` で decompress (圧縮の解除) すると、完全に元に戻る。

Windows などでは、複数のファイルを一つにまとめてから圧縮をするコマンド (アーカイバー (archiver) と呼ばれる) が普及している (Lha, WinZip, Cab, ...) [9]。

[9] UNIX では、複数のファイルを一つにまとめるコマンドとして、**tar** が良く使われている。tar と例えば gzip を組み合わせることで Windows のアーカイバーと同様のことができる。このあたりはプログラムの設計思想の違いであろう (単一の機能を持ったプログラムを複数組み合わせて利用して複雑な仕事をしようという UNIX と、一つのプログラムで色々なことに対応しようとする Windows)。