Virtual Font について

（株）アスキー 出版技術部 濱野 尚人
hisato-h@ascii.co.jp

1991年8月30日

Abstract

TEX で PostScript のフォントを使って出力を得ようとするとき、CM フォントを使用する場合にはなかったいくつかの問題が生じる。その解決策の一つとして、Knuth は TEX3.0 の発表に前後して Virtual Font という概念を TEX システムに導入した。

Virtual Font を使用すれば、PS フォントのコード体系を CM フォントに合わせることができるし、CM フォントを PS フォントの代用とすることによって、普通のプレビューアや PS が搭載されていないプリンタに出力が可能になる。

1 dvi ファイルとプリンタドライバ

dvi ファイルはプリンタドライバにどのように印字すれば良いかを指示するためのファイルである。その指示はコンピュータの機械語に似た命令列の形になっている。プリンタドライバは、この命令を解釈し実行するインタプリタであると考えることができる。このインタプリタは内部レジスタをいくつか持っている。その中で重要なのは、カレントポイントの座標を保持する (h,v) レジスタと、カレントフォントを保持する f レジスタである。

dvi の命令は大きく 4 つに分類される。

- dvi ファイルの構成をドライバに伝える命令

  dvi ファイル全体に関わる情報や、ページの区切りを表現する。

- フォントを定義する命令

  dvi ファイルで使用しているフォントに、dvi ファイル内部のフォント番号を割り当る。

- レジスタの値を変更する命令

  カレントポイントやカレントフォントを変更する。

- 印字を行う命令

  カレントポイントの位置に、指定したコードの文字や罫線（水平/垂直線）を印字する。

dvi ファイルのサイズが小さくなるように、実際の命令体系には印字とカレントポイントの変更を同時に行うような複合命令が用意されていて、TEX の出力する dvi の中でも多用されている。特に *set* 命令（*set_char* 命令）では、文字を印字したあとで文字の幅だけ右に移動することになっていて、この文字幅が TEX が組版処理するときの tfm ファイルの値と一致しないと美しい出力が得られないので注意が必要である。

dvi ファイルでは、文字の位置とコードは具体的に指示されるが、文字の形はフォント名で指示されるだけである。具体的な文字の形（グリフ）は別の方法で用意する必要がある。従来は主に 2 通りの方法でグリフを得るのが普通であり、それぞれ次のような特徴がある。

- METAFONT で生成されたビットマップフォントを使う

  CM フォントの METAFONT は公開されており、どこでも、どのプリンタでも利用できる。

- プリンタの内蔵フォントを使う

  プリンタ（写植機）によっては、CM フォントとは較べものにならないくらい豊富なフォントを持っており、それが利用できる。

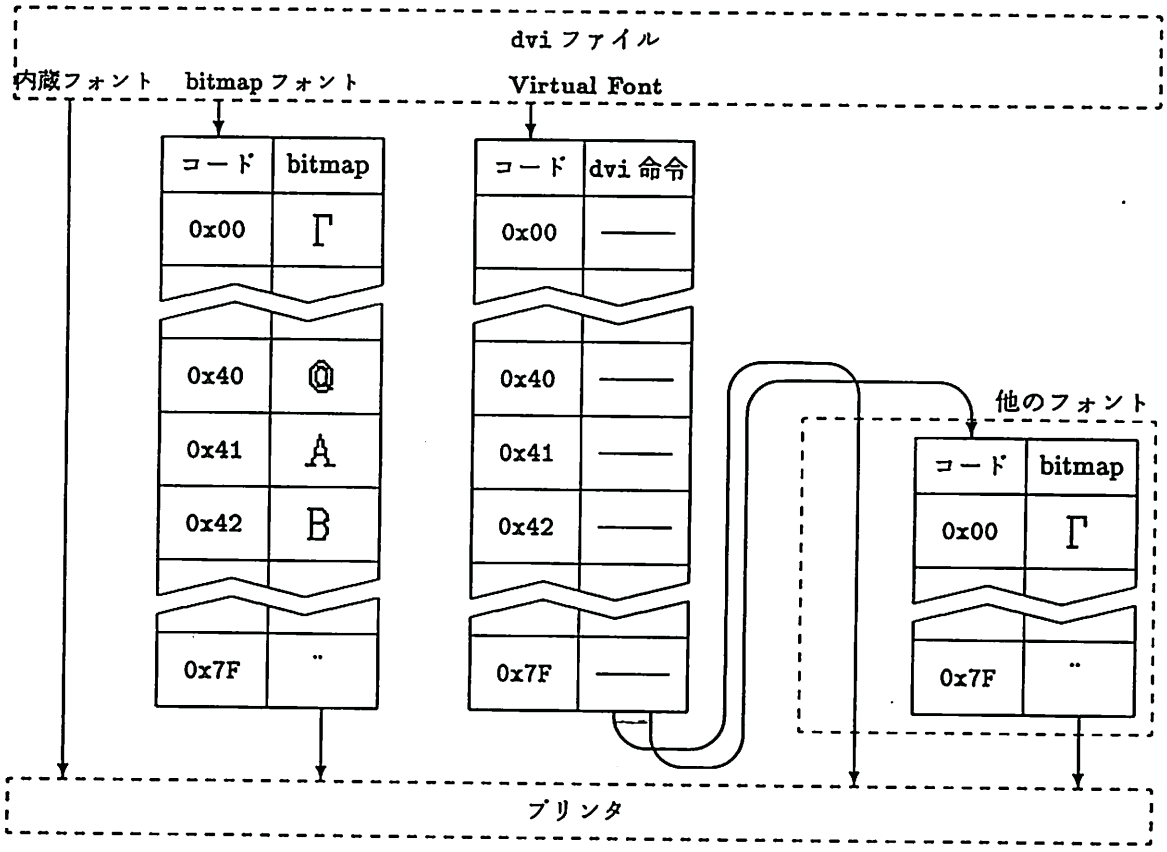
  ホストとプリンタ間でビットマップを転送する必要がなく、プリントアウト時間が短縮できる。

Virtual Font は、第三のグリフ実現法である。

2 Virtual Font

Virtual Font とは、グリフを、他のフォントのグリフを借用して実現するフォントである。これを実現するために vf ファイルを使用する。vf ファイルには、そのフォントに含まれる文字 1 文字 1 文字について、グリフをどのように実現するかが記述してある。

グリフの実現方法の記述には、dvi 命令が使われる。dvi 命令の記述によって、CM フォントや内蔵フォントや罫線（rule）を組み合せ、グリフを実現するのである。プリンタドライバから見ると、*set_char*、*put_char* という dvi 命令を実行するために vf ファイルの dvi 命令を解釈する必要があり、dvi のサブルーチンのような処理をすることになる。vf ファイルフォーマットと Virtual Font を扱うプリンタドライバの動作は後述する。



次に、Virtual Font を使用する例を挙げる。

2.1 cmsscsc10

標準のCMフォントファミリーには、サンセリフのスモールキャピタルフォントは含まれていない。Virtual Fontを使えば、cmss10とcmss7を組み合せて、これを実現することができる。

cmsscsc10.vfを、以下のように作成する。cmss10をデフォルトフォントとして使い、フォント番号1でcmss7を呼び出すように設定する。大文字や記号文字はそのままデフォルトのフォントで出力し、小文字はフォント番号1を呼び出して印字するように定義しておく。

さらに大文字と小文字の組み合せのカーニング等を調整したtfmファイル（cmsscsc10.tfm）を用意すれば、TEXからこのフォントを利用することができる。

2.2 PostScriptフォントをより使いやすく

dvi2psとAFMファイルから生成されたtfmファイルと特殊なマクロを使うことによって、PostScriptの内蔵フォントをTEXで利用することが可能になる。しかし、PostScriptのフォントはCMフォントと異なるコードレイアウトを持っており、plain.texなどのCMフォント用のマクロをそのまま使うことができない。また、CMフォントとPSフォントの混在は困難である。

そこでVirtual Fontを使って、コードレイアウトはCMフォントと同じでグリフはPSフォントを利用するような新しいフォントを作ることにより、CMフォント用のマクロでPSフォントが利用可能になる。

例えばウムラウトアクセントのコードは、CMフォントの場合は0x7Fで、PSフォントの場合は0xC8である。vfファイルに、コード0x7Fが呼ばれたときはPSフォントの0xC8を呼び出すように記述すれば、\"の定義を変更することなしにPSフォントでウムラウトアクセントを利用することができる。

2.3 PostScriptフォントを使ったdviのプレビュー

PSフォントと同じ名前、同じ文字幅を持ち、CMフォントでグリフを表現するVirtual Fontを用意することによって、PSフォントを使ったdviをPSプリンタ以外のプリンタやプレビューアで出力可能になる。

組版や内容をプレビューアや高速なプリンタでチェックした後、本番の出力だけをPSプリンタで行うことができる。

2.4 tfmパラメータのチューニング

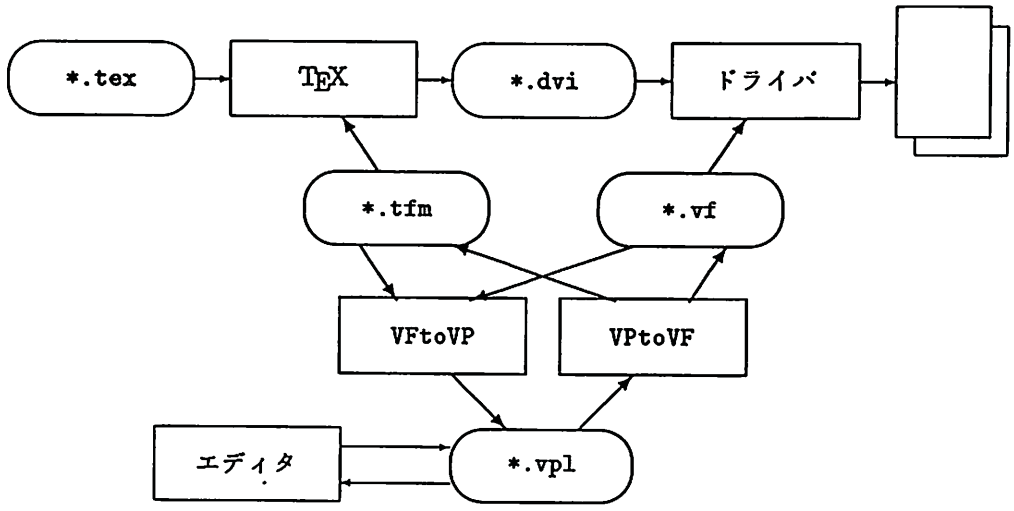
グリフを変更しないで、カーニング等のtfmパラメータをチューニングしたい場合がある。コンパチビリティのために、チューニングされたフォントはオリジナルと区別がつくような名前にする必要があるが、グリフまで別々に持つのはディスクのムダである。

Virtual Fontを使うことによって、オリジナルバージョンとチューニングバージョンでグリフを共有することができ、ディスクを節約できる。

3 Virtual Font関係のツール

TEX 3.0のディストリビューションには、VPtoVF、VFtoVPというツールが含まれている。これは、vfファイルtfmファイルと、vfファイルの内容をテキストファイルとして表現するvplファイルとを相互に変換するツールである。

新たにVirtual Fontを作成し利用する場合にはvfファイルとtfmファイルが必要になる。vplファイルをテキストエディタで作成し、VPtoVFを使うことによって、これらのファイルを作成することができる。また、これらのツールのWEBソースには、vfファイル、vplファイルのフォーマットの詳細が記述されている。



4 vfファイルの構造

vf ファイルフォーマットについて簡単に説明する。詳細は VPtoVF、VFtoVP のソースを参照して欲しい。

vf ファイルは dvi や pk に似た構造を持つバイナリファイルで、やはり、プリアンブル、本体、ポストアンブルに分かれている。

4.1 プリアンブル

プリアンブルの最初は *pre* 命令で、その後に *fnt_def* 命令がいくつか並ぶ。vf の文字定義の *dvi* フィールド中の *set_char*、*put_char* などで使うフォントは、ここで定義しておかなければならない。

pre 命令は主にこの Virtual Font のデザインサイズを定義する。

- *pre* <247> $i[1]$ $k[1]$ $x[k]$ $cs[4]$ $ds[4]$
  - i は *id* バイトで、現在は 202 である。
  - $x[k]$ はコメントで、通常は vf のソースが何であるかを表している。
  - cs と ds は、Virtual Font のチェックサムとデザインサイズである。この Virtual Font の tfm の header の最初の 2 ワードと等しくなければならない。

fnt_def 命令は dvi のそれと同じ構造である。

- *font_def1* <243> $k[1]$ $c[4]$ $s[4]$ $d[4]$ $a[1]$ $l[1]$ $n[a+l]$
  font_def2 <244> $k[2]$ $c[4]$ $s[4]$ $d[4]$ $a[1]$ $l[1]$ $n[a+l]$
  font_def3 <245> $k[3]$ $c[4]$ $s[4]$ $d[4]$ $a[1]$ $l[1]$ $n[a+l]$
  font_def4 <246> $k[4]$ $c[4]$ $s[4]$ $d[4]$ $a[1]$ $l[1]$ $n[a+l]$
  - 最初に定義されたフォントはデフォルトフォントとして扱われる。
  - フォント番号 k は “ローカル” である。この Virtual Font を使っている dvi のフォント番号とは、何の関係もない。
  - 寸法 s は、定義されたローカルフォントの scaled size を表し、Virtual Font のデザインサイズとの比を 2^{-20} を単位として表す。もし、ローカルフォントがその Virtual Font 自身のデザインサイズと同じ大きさで使われるなら、s は 2^{20} となる。$2^{24} > s > 0$ でなければならない。

- 寸法 d はローカルフォントのデザインサイズを、2^{-20}ポイントを単位ととして表している。該当する tfm のデザインサイズと一致するはずである。

4.2 文字定義

プリアンブルの次には、vf ファイルの本体である、文字定義の並びがある。そのフォントに含まれる文字には、必ず 1 つの文字定義が存在しなければならない。

- *long_char* <242> *pl*[4] *cc*[4] *tfm*[4] *dvi*[*pl*]
- *short_char0* <0> *cc*[1] *tfm*[3] *dvi*[$pl = 0$]
  short_char1 <1> *cc*[1] *tfm*[3] *dvi*[$pl = 1$]
  ⋮
  short_char241 <241> *cc*[1] *tfm*[3] *dvi*[$pl = 241$]
  ($0 \leq pl < 242$、$0 \leq cc < 256$、$0 \leq tfm < 2^{24}$の条件を満たす文字を定義する)
  - *pl* は *dvi* フィールドの長さを示している。*short_char* 命令では OP コード自体が *pl* を表している。
  - *cc* は文字コードである。
  - *tfm* は、この Virtual Font の tfm からコピーされた文字幅である。
  - *dvi* は dvi 命令の連続である。*bop*、*eop* と 243 以上 249 以下のコードを持つ命令以外は、使うことができる。ここで使うフォント選択命令 (*fnt*、*fnt_num*) は、dvi ファイルのフォント定義ではなく、vf ファイルのプリアンブルのフォント定義を参照する。

dvi 命令で使われる寸法は、Virtual Font のデザインサイズの 2^{-20}倍を単位としている。たとえば Virtual Font のデザインサイズが 10pt のときは、カレントポイントを 5pt 下げる dvi 命令は、*down* 2^{19} となる。Virtual Font がデザインサイズと異なるサイズ、たとえば 12pt (\magstep1) で使われたときは、この *down* 命令で 6pt 下がることになる。それぞれの寸法の絶対値は、2^{24}を超えてはならない。

デバイスドライバは、*dvi* フィールドを暗黙の *push*、*pop* で囲まれたサブルーチンやマクロのように処理する。各々のサブルーチンの最初では、$w = x = y = z = 0$、fはプリアンブルで最初に定義されたフォント番号 (フォントが定義してなければ未定義) になる。

dvi フィールドの処理が終った後、h、v、w、x、y、z、fレジスタは前の値に戻される。もしサブルーチンが *set_char* か、*set* 命令で起動されたのなら、普通の文字が印字されるのと同じように、h レジスタに tfm width が加えられる。

4.3 ポストアンブル

vf ファイルの最後には 1 個以上の *post* 命令があって、ファイルサイズが 4 の倍数になるようになっている。

- *post* <248>

5 日本語 Virtual Font (拡張案)

vf は、特に拡張することなく、日本語を扱うことが可能である。しかし、1 文字に対して 1 つの文字定義が必要になり現実的ではない。jfm と同様にいくつかの文字をグループにして、そこに属する文字の表現を一括して行えるような仕組みが必要である。

なお、以下に説明するフォーマットは検討中であり、変更もあり得る。

5.1 マジックナンバ

ここで提案する拡張 vf フォーマットは、*pre* 命令の i の値は 54 とする。

5.2 グループ化

- *group* <249> *pl*[4] *cc*[4] *mem*[*pl*]
  - *mem* にはこのグループに含まれる文字のコードが、1 文字あたり 4 バイトで入っている。
  - *pl* は、*mem* のサイズをバイト数で表す。
  - このグループに含まれる文字は、*cc* で表される文字と、文字幅と文字定義を共有する。

文字定義もなくグループにも属していない文字は、コード 0 の文字と、文字幅と文字定義を共有する。

5.3 文字コード置換

文字定義の *dvi* フィールドに現れた *set2* 0、*put2* 0 は、Virtual Font 自身の文字コードの *set* 命令、*put* 命令に置換される。たとえば、コードが 0x2422 の文字定義の中の *set2* 0 は、*set2* 0x2422 に置換される。

グループ化機能を使っていくつかの文字で一つの文字定義を共有していても、この機能によって、それぞれの文字を印字することができる。

5.4 *dir* 命令

Virtual Font を使って縦組み用のフォントを作ることを考えて、pTEX が出力する dvi に使われている命令である *dir* 命令を、文字定義の *dvi* フィールドに使えるようにする。

pTEX の dvi の *dir* 命令はページに対して座標系を回転する命令であったが、文字定義の *dir* 命令は、dvi の座標系に対して Virtual Font の座標系を回転する命令である。

- *dir* <255> *d*[1]
  - ローカルな座標系を dvi の座標系に対して時計回りに $d \times 90°$ 回転する。$0 \leq d \leq 3$ である。

既に dvi ファイルの中で座標系が回転され "tate" になっているときに、Virtual Font が使われてその文字定義の中で "*dir* 1" を実行すると、Virtual Font の座標系は上下左右が逆になる。

参考文献

[1] Knuth, D. E., 『Virtual Fonts: More Fun for Grand Wizards』, TUGboat 11, no. 1, pp.13–23, 1990

[2] Knuth, D. E., 『The VPtoVF processor』, 『The VFtoVP processor』, TEX 3.0 ディストリビューションテープ, ./TeX3.0/fontutil, 1989

[3] アスキー出版技術部責任編集, 『日本語 TEX テクニカルブック I』, (株) アスキー, 1990

[4] 濱野 尚人 他, 『TEX の出版への応用—縦組み機能の組み込み—』, 第 10 回 日本 TEX ユーザーグループ例会資料, 1990