Fontologia

Fontowe ABC

Bogusław Jackowski
i Stanisław Wawrykiewicz

Co jakiś czas na liście dyskusyjnej GUST-u padają głosy świadczące z jednej strony o dużym zainteresowaniu sprawami fontowymi, a z drugiej strony – o częstym niezrozumieniu podstawowych spraw związanych z fontami.

Zainteresowanie fontami jest o tyle uzasadnione, że ostatnimi czasy pojawia się na rynku coraz więcej fontów PostScript-owych, dających się wykorzystać do składania polskich tekstów – ot, choćby fonty dystrybuowane przez firmę ThETA.

Te aspekty skłoniły nas do podjęcia próby spisania fontowego elementarza dla tych dzielnych adeptów fontologii, którzy nie boją się pułapek, od których niestety aż się roi w świecie fontów.

Fonty „TEX-owe" czyli pliki TFM. TEX-owi jest absolutnie obojętne, jakich fontów używamy: czy to są mapy bitowe, fonty PostScript-owe w formacie Type 1, czy też może egzotyczne fonty „własnego chowu" – nie ma to znaczenia. Jedyne, czego potrzebuje TEX, to informacja metryczna, czyli pliki `TFM` (TEX Font Metric). Pliki metryczne, jak sama nazwa wskazuje, zawierają w pierwszym rzędzie informację o rozmiarach znaków, ale także informację o wielkości kernów (podcięć) dla par znaków, informację o wielkości korekty kursywy (*italic correction*) dla poszczególnych znaków oraz – choć trudno to zakwalifikować jako informację metryczną – informację o ligaturach, tzn. informację, które pary znaków mają być zastępowane automatycznie innym znakiem, np. dwa umieszczone obok siebie znaki `--` zamienione zostaną przez TEX-a przez pojedynczy znak „–" zwany pauzą półfiretową (*endash*).

Krótko mówiąc, żeby w ogóle przetworzyć dokument TEX-em, potrzebne są pliki `TFM`.

Fonty „właściwe". Dopiero sterowniki zamieniając złożone dokumenty, czyli pliki `DVI`, na postać widoczną już to na ekranie, już to na papierze, już to na jakimś innym nośniku, potrzebują tych „właściwych fontów".

Fonty rodziny Computer Modern. Zgodnie z konwencją przyjętą przez Donalda E. Knutha nazwy fontów rodziny Computer Modern zawierają formalny rozmiar fontu w punktach, np. `cmr10` (font 10-punktowy), `cmbx7` (font 7-punktowy), itp. Należy tu podkreślić, że font `cmr12` nie jest równoważny fontowi zdefiniowanemu jako `cmr10 at 12pt` – chociaż oba fonty mają wielkość 12 punktów, to jednak proporcje znaków w obu fontach są inne. Tę samą cechę dziedziczą oczywiście fonty wywodzące się z rodziny Computer Modern (Computer Concrete, `EC`, `PL`), a także fonty rodziny `Euler`, fonty cyryliczne, i inne fonty tworzone w duchu konwencji rodziny Computer Modern.

Zmiany proporcji wraz ze zmianą stopnia pisma są jak najbardziej uzasadnione, małe fonty powinny dla lepszej czytelności mieć nieco większe i szersze oczka minuskuł w stosunku do majuskuł. Niemniej jednak wydaje się, że Donald E. Knuth nieco przesadził z liczbą odmian fontów zmieniając co jeden punkt proporcje znaków.

W praktyce wystarczyłyby nie więcej niż cztery zakresy: pierwszy dla fontów mniejszych niż 8-punktowe, drugi dla fontów od 8 do 12 punktów, trzeci dla fontów od 12 do – powiedzmy – 24 punktów i wreszcie czwarty dla fontów tzw. nagłówkowych. Ostatni z wymienionych zakresów nie pojawia się w fontach `Computer Modern`. Font calowy `cminch`, przeznaczony do nagłówków, jest w istocie przeskalowanym proporcjonalnie fontem `cmssbx10`. Innymi słowy dokładnie ten sam efekt uzyskałoby się za pomocą instrukcji `\font \f=cminch` i za pomocą instrukcji `\font \f=cmssbx10 at 1.44in` (sic!). Jedyna różnica polega na tym, że font `cminch` jest nieco „oszczędniejszy" – zawiera tylko duże litery i cyfry.

Sporo zamieszania wywołały także `plain`-owe instrukcje `\magstephalf` i `\magstep1`, `\magstep2`, ..., `\magstep5`, sugerujące, że jedyne „legalne" powiększenia fontów to 109,5%, 120%, 144%, 172,8%, 207,4% lub co najwyżej 248,8% (p. *The TEXbook*, str. 17). Mimo iż nie jest to nigdzie *expressis verbis* powiedziane, to jednak sporo nieporozumień narosło na tym tle. Trudno się temu dziwić, skoro prawie wszystkie pakiety zawierające fonty TEX-owe w postaci map bitowych (p. niżej) stosowały się do tej reguły, tzn. mapy bitowe generowane były tylko dla wymienionych wyżej powiększeń. Chcąc w dokumentach TEX-owych w sposób

jednolity korzystać z fontów tradycyjnych, czyli bitmapowych, i z fontów POSTSCRIPT-owych, nie należy przejmować się zanadto rozpowszechnionym obyczajem mającym swe korzenie w czasach, gdy komputery były wolniejsze i wygenerowanie fontu „od ręki" było zbyt kosztowne. Rzadko używana instrukcja postaci `\font ... at ...` jest w tym kontekście godna polecenia.

Fonty bitmapowe. W tradycyjnej konfiguracji systemu TEX podstawowymi fontami są mapy bitowe (pliki `GF` – *generic font*) generowane przez program METAFONT. Najczęściej używanym formatem jest jednak nie format `GF`, a nieco oszczędniejszy format `PK` (*packed*). Do zamiany służy program `GFTOPK`, z reguły wywoływany automatycznie w procesie generowania fontów.

Fonty bitmapowe generowane są dla *konkretnej wielkości pisma i konkretnego urządzenia*, tzn. zawierają pewne informacje pozwalające na danym urządzeniu uzyskać optymalny wygląd znaków. Stąd np. odrębne fonty dla drukarek laserowych i drukarek atramentowych o tej samej rozdzielczości 300 dpi.

TEX – jak już wspomnieliśmy – pozwala skalować całość dokumentu lub poszczególne fonty. Na przykład to samo przeskalowanie fontu można wyrazić na (co najmniej) trzy różne sposoby:

```
\font\A plr10 scaled 1728
\font\B plr10 scaled \magstep3
\font\C plr10 at 17,28pt
```

gdzie `plr10` oznacza nazwę fizycznie posiadanego na dysku pliku `plr10.tfm`. TEX przeskaluje font `plr10` tak, jak sobie tego życzymy, a ściślej – przeskaluje informację metryczną związaną z fontem. Żeby móc zobaczyć, jak wyglądają znaki przeskalowanego fontu, należy skorzystać z METAFONT-a, który oczywiście potrafi generować fonty dowolnie przeskalowane.

Fonty `PK` zwykle przechowywane są w katalogach o nazwach zawierających liczbę wynikającą z przemnożenia rozdzielczości przez przeskalowanie, np. fonty o rozdzielczości 300 dpi (*dots per inch*) powiększone o 20% (`\magstep1`) znajdą się w katalogu o nazwie `360\`, `360dpi\`, `dpi360` lub jakimś innym, podobnie nazwanym. Jeśli system operacyjny zezwala na nadawanie plikom długich nazw, to przeskalowanie może być wpisane w nazwę pliku, np. `fonts/pk/laserjet/pl/plr10.360pk` i zbędne są wtedy odrębne podkatalogi.

Standardowe sterowniki przetwarzające pliki `DVI` czy to na postać wyświetlaną na ekranie, czy też drukowaną, używają fontów `PK` i posiadają mechanizm przeszukiwania odpowiedniej struktury katalogów lub interpretacji nazw fontów przeskalowanych. Sterowniki te pozwalają zwykle na bardzo szybkie wyświetlenie składanego tekstu. Współczesne sterowniki posiadają także możliwość współpracy z METAFONT-em – potrafią przekazać informację o brakujących fontach i uruchomić METAFONT-a, wstrzymując pracę do czasu zakończenia procesu generowania brakujących fontów. Dzięki temu nie ma obecnie potrzeby trzymania na dysku olbrzymich bibliotek fontów w formacie `PK`, wystarczą pliki źródłowe fontów i odpowiednio skonfigurowane łącze sterownik-METAFONT (program lub skrypt) by zautomatyzować proces generowania fontów. Dla bardzo rozpowszechnionych sterowników z pakietu `emTeX` będzie to program `MFjob` (`MS-DOS`, `OS/2`), zaś dla systemów opartych na `web2c` – skrypt `MakeTeXPK` (różne wersje `U**X`, `Windows95` i `NT`, `Amiga`).

Fonty PostScript-owe czyli pliki AFM oraz PFB (lub PFA). Jeżeli chodzi o POSTSCRIPT, to ten „właściwy font" – mowa tu o fontach zapisanych w formacie Adobe Type 1 – zawarty jest w pliku z rozszerzeniem `.pfb` (*POSTSCRIPT Font Binary*) lub `.pfa` (*POSTSCRIPT Font ASCII*). Natomiast o plikach `TFM` w świecie POSTSCRIPT-owym mało kto słyszał. Na szczęście dobrzy ludzie z firmy Adobe wymyślili coś podobnego do tego, co wymyślił Donald E. Knuth, mianowicie pliki metryczne AFM (*Adobe Font Metric*). Różnią się one trochę od plików `TFM` (np. pliki `AFM` są plikami tekstowymi), ale mimo różnic jest możliwe by na podstawie pliku `AFM` utworzyć plik `TFM`.

Reprezentacja znaków. Zanim pójdziemy dalej trzeba wyjaśnić jedną rzecz, mianowicie w jaki sposób są reprezentowane znaki. Jeżeli chodzi o plik `DVI` to odpowiedź jest trywialna: jako liczby z zakresu 0–255. Przy tworzeniu pliku POSTSCRIPT-owego za pomocą sterownika sytuacja *nie ulega* zmianie, to znaczy w pliku POSTSCRIPT-owym znaki też są reprezentowane w zasadzie jako liczby z zakresu 0–255, czasem w reprezentacji ASCII, a czasem w reprezentacji ósemkowej. Natomiast we „właściwym foncie" POSTSCRIPT-owym, podobnie zresztą jak w foncie

METAFONT-owym, jest inaczej: znaki są zasadniczo reprezentowane za pomocą nazw w rodzaju `period`, `periodcentered`, `lslash`, `aogonek`, itp.

Formalnie rzecz biorąc font POSTSCRIPT-owy jest programem, a nazwy znaków są nazwami podprogramów opisujących kształt znaków za pomocą – znów podobnie jak w programach METAFONT-owych – krzywych trzeciego stopnia, zwanych krzywymi Béziera. Dodatkowo podprogramy te opisują niektóre szczegóły procesu zamiany obwiedni na mapy bitowe (tzw. *hinty* czyli podpowiedzi), bo przecież ostatecznie znak zostanie wyświetlony na ekranie, wydrukowany na drukarce lub naświetlony na diapozytywie, czyli musi zostać zamieniony na postać dyskretną.

Nazwy i kody znaków. Skąd zatem POSTSCRIPT trafiając na jakąś liczbę „wie", że ma użyć procedury o nazwie, powiedzmy, „aogonek"?

Aaa, właśnie! Tu jest pies pogrzebany. We „właściwym foncie" POSTSCRIPT-owym jest zawarta dodatkowa informacja, mianowicie tabela par (liczba, nazwa) przyporządkowująca liczbom z zakresu 0–255 nazwy podprogramów opisujących znaki. Tablica ta (czasem zwana wektorem) nosi nazwę „Encoding", co jest może mało istotne, natomiast istotne jest to, że tablicę tę można „w locie" zmienić. Z tej możliwości korzysta m. in. sterownik `DVIPS` (p. niżej).

Konwersja AFM → TFM. W pliku AFM znajduje się również informacja o przyporządkowaniu liczba-nazwa, (w zasadzie można zakładać, że jest to to samo przyporządkowanie, co w pliku `PFB/PFA`). Na ogół nie jest to przyporządkowanie, o które by nam chodziło. Ale nie ma biedy – `DVIPS`-owi można podać *dowolne* przyporządkowanie, które zostanie użyte w tworzonym przezeń pliku POSTSCRIPT-owym.

Konwertery `AFM→TFM` na podstawie nazw generują stosowne przyporządkowanie, lub nawet wiele przyporządkowań, a dla każdego z takich przyporządkowań tworzony jest odpowiadający mu plik `TFM`.

Najpowszechniej używanym konwerterem jest program `AFM2TFM`, należący do standardowej dystrybucji sterownika `DVIPS`. Inny konwerter o nazwie `TOIL`, bazujący na programach `AWK` oraz METAFONT, opisany jest pokrótce na stronach 22–23 niniejszego biuletynu. Warto też wspomnieć o pakiecie `fontinst`, oferującym liczne możliwości, bazującym na fontach wirtualnych (p. str. 12–21 niniejszego biuletynu).

Konwersja AFM → TFM – poprawność nazw znaków. Aby konwertery `AFM→TFM` mogły działać jak należy, nazwy podprogramów muszą być zgodne z ich treścią. Tak niestety nie zawsze jest, np. pod nazwą „Sterling" zamiast opisu znaku funta można znaleźć opis polskiego „Ł" czyli „Lslash". Pomysłów na cudaczne nazewnictwo jest mniej więcej tyle, ile tzw. standardów kodowania znaków, czyli $\approx \infty$.

Z niepoprawnymi nazwami można sobie poradzić. Co nam zależy, przecież „onesuperior" brzmi równie ładnie jak „aogonek", nieprawdaż? Problem jedynie w tym, że z góry nie wiadomo, że producent uznał „onesuperior" za odpowiednią nazwę dla polskiej litery „ą". W fontach firmy ThETA (p. niżej) akurat tak jest, ale tak bynajmniej być nie musi, w końcu jakiś szaleniec może się pogodzić z propozycją Adobe i użyć dla podprogramu opisującego „ą" nazwy „agonek"...

Jakby nie dość było problemów, oprócz nazewnictwa ewidentnie niepoprawnego możemy się zetknąć z obocznością! Otóż niektórzy producenci dla określenia znaków „ż" i „Ż" używają nazw „zdot" i „Zdot", a niektórzy „zdotaccent" i „Zdotaccent". Nie znalazłem jednoznacznego stanowiska firmy Adobe w tej sprawie, ale na ogół obowiązuje zasada „sklejania" nazw: „a" + „grave" = „agrave"; skoro więc istnieje znak „dotaccent" (nazwa używana przez Adobe), to „ż" należy rozumieć jako „z" + „dotaccent" czyli „zdotaccent". Innymi słowy zalecać by należało używanie nazw „zdotaccent" i „Zdotaccent", ale być może konwertery `AFM→TFM` powinny być uwrażliwione na tę oboczność.

Konwersja AFM → TFM – kerny i ligatury. Plik `TFM` zawiera jeszcze informację o parach kernowych i ligaturach. Pary kernowe, uwzględniane „jak leci" przez większość konwerterów `AFM→TFM`, na ogół są umieszczane jak należy w plikach `AFM`, chociaż czasami można trafić na felerne pliki `AFM` bez par kernowych. Obecność par kernowych poznajemy po występowaniu wierszy zaczynających się od słowa kluczowego „`KPX`". Warto mieć na uwadze, że bywają pliki `AFM` posiadające tylko część par kernowych – np. znaki diakrytyczne czasami nie są uwzględniane – dobrze jest sprawdzić plik `AFM` pod tym kątem.

Z ligaturami jest znacznie gorzej – można je bardzo rzadko uświadczyć, mimo iż teoretycznie format AFM pozwala na definiowanie ligatur, np. poniższy przykładowy wiersz pliku AFM (tu przełamany ze względu na ograniczoną szerokość łamu):

```
C 45; WX 329; N hyphen; B 17 188 313 303;
    L hyphen endash;
```

zgodnie ze specyfikacją Adobe ustala, że znak o kodzie 45 ma szerokość 329 jednostek (jednostka = 1/1000 em), nosi nazwę hyphen, że naroża prostokąta ograniczającego obrys znaku mają współrzędne wynoszące w tych samych jednostkach $(17, 188)$ oraz $(313, 303)$, i wreszcie że dwa znaki hyphen mają być zamieniane na znak o nazwie endash, czyli innymi słowy znak o nazwie endash jest ligaturą dwóch znaków o nazwie hyphen.

W związku z powszechnym brakiem informacji o ligaturach w plikach AFM, konwertery AFM → TFM muszą albo się domyślić, jakie by tu ligatury można wstawić, albo – co chyba jest lepszym rozwiązaniem – zostać poinformowane o prawidłowych dla danego fontu (grupy fontów) ligaturach.

Pliki konfiguracyjne sterownika DVIPS. Popularny sterownik DVIPS, autorstwa Tomasa Rokickiego, zamieniający pliki DVI na pliki PostScript-owe musi wiedzieć – podobnie jak każdy sterownik – gdzie znajdują się fonty TEX-owe i PostScript-owe. Informację tę zawierają pliki config.ps i psfonts.map, poszukiwane w kartotekach wskazywanych przez zmienną systemową TEXCONFIG.

Plik config.ps zawiera informację o lokalizacji plików TFM, plików PK, fontów wirtualnych, oraz fontów PostScript-owych. Natomiast plik psfonts.map zawiera informacje związane wyłącznie z dostępnymi w danej instalacji fontami PostScript-owymi. W pliku psfonts.map informacje o poszczególnych fontach umieszczane są w jednym wierszu (co czasem może być kłopotliwe). Typowy wiersz wygląda następująco:

```
qplr QuasiPalladio-Regular
     "enc-qpl ReEncodeFont"
     <qpl.enc <qplr.pfb
```

Pierwsza pozycja określa nazwę TEX-ową fontu, druga – wewnętrzną nazwę PostScript-ową fontu (robi z niej użytek interpreter PostScript-u), następnie pojawia się informacja o dodatkowych zabiegach, jakim ma być poddany font (może to być poziome przeskalowanie fontu, pochylenie lub zmiana kodowania – w tym wypadku chodzi o zmianę kodowania), a na końcu pojawiają się nazwy plików, które mają zostać włączone do wynikowaego pliku PostScript-owego – w tym wypadku jest to plik kodowania qpl.enc oraz font qplr.pfb.

Podanie nazwy pliku fontowego nie jest konieczne – w takim wypadku DVIPS założy, że chodzi o font wbudowany w urządzenie. Gorąco odradzamy korzystanie z tej możliwości – opłakane skutki korzystania z fontów wbudowanych można co jakiś czas oglądać w materiałach nawet renomowanych firm: a to zamiast polskich znaków występują „krzaczki", a to litery zachodzą na siebie, a to skład się zupełnie rozjeżdża…

Plik kodowania zawiera tablicę 256 nazw znaków, określającą omawiane wyżej przyporządkowanie kod-nazwa. Jest to w istocie fragment kodu PostScript-owego postaci:

```
/enc-qpl[
/.notdef
/Delta
/beta
...
/thorn
/quotedblbase
] def
```

O tym, jaki kod otrzymuje dany znak, decyduje kolejność wystąpienia: w tym wypadku na pozycji zerowej (w języku PostScript liczenie zawsze odbywa się od zera) występuje znak o nazwie .notdef (znak „ciach" / w notacji PostScript-owej oznacza mniej więcej to samo co znak „w-tył-ciach" \ w notacji TEX-owej), co oznacza, że w foncie w istocie nie ma znaku o tym kodzie; znak Delta otrzymuje kod 1, beta – kod 2, …, znak thorn – kod 254, i wreszcie znak quotedblbase – kod 255.

Warto mieć świadomość, że DVIPS włącza wszelkie pliki prawie ich nie analizując, jedyny zabieg przezeń wykonywany to zamiana postaci binarnej fontu Type 1 (PFB) na postać ASCII (PFA).

Jeżeli font jest już w postaci ASCII, to włączany jest do dokumentu „jak leci", bez żadnej ingerencji ze strony sterownika. Oznacza to, że używając DVIPS-a możemy korzystać ze

wszystkich możliwych fontów POSTSCRIPT-owych, nie tylko najbardziej popularnych fontów Type 1, ale w szczególności także fontów TrueType, które dla interpretera POSTSCRIPT-u są po prostu fontami Type 42 – wystarczy mieć plik `TFM` i font Type 42 w postaci ASCII.

Łatanie fontów – niepoprawny font. Używając różnych wymyślnych narzędzi (np. `T1UTILS` Lee Hetheringtona, do wzięcia z GUST-owego archiwum `/fonts/utilities/t1utils.zip`) można się zorientować, jakim znakom zostały przypisane jakie nazwy i w razie czego je poprawić, można też uzupełniać kerny, itp., ale to jest żmudne i przykre, bo niepotrzebne zajęcie.

Łatanie fontów – brak plików AFM. Czasem fonty POSTSCRIPT-owe są dystrubuowane z plikami `PFM` (*Printer Font Metric*) zamiast z plikami AFM.

Brak plików `AFM` *praktycznie przekreśla* możliwość użycia fontu POSTSCRIPT-owego w TEX-u, bowiem pliki `PFM` zawierają znacznie uboższą informację niż pliki AFM, w szczególności *nie zawierają* nazw znaków, czyli nie określają żadnej tablicy kodowania (*encoding vector*, p. wyżej). Można zakładać, że chodzi o takie samo przyporządkowanie, jakie jest wpisane w pliku `PFB/PFA`, chociaż w ogólności tak być nie musi. Ponadto w pliku `PFM` znajdują się dane na temat par kernowych jedynie dla co najwyżej 256 znaków, a font POSTSCRIPT-owy może ich zawierać znacznie więcej (chociaż POSTSCRIPT nie może wykorzystywać równocześnie więcej niż 256 znaków).

Jeśli jednak szczęście nam dopisze, to z plików `PFM` oraz `PFB` (bądź `PFA`) możemy odtworzyć informację niezbędną do utworzenia pliku `AFM` – można spróbować użyć pakietu `P2A`, który niedawno wzbogacił archiwum GUST (`fonts/utilities/p2a/`).

Casus ThETA. Ostatnio na rynku Polskim pojawiły się godne uwagi polonizacje fontów firmy BitStream, sygnowane przez firmę ThETA.

Niestety firma ThETA wyraźnie zignorowała rynek TEX-owy, chociaż może i słusznie zignorowała, skoro TEX-owców jakoś nie widać i nie słychać, no może z wyjątkiem kilkorga spośród 180 niegroźnych wariatów zapisanych do GUST-u…

Otóż firma ThETA wydała jak do tej pory 3 płyty CD z fontami, każda kolejna płyta zawierała inne fonty i była coraz mniej przydatna dla TEX-owców:

- Pierwsza płyta (cena ok. 500 zł) miała nazwy znaków w zasadzie poprawne z dokładnością do `zdot` zamiast `zdotaccent`. To zapewne spowodowało, że płyta ta ochrzczona została „TEX-ową” przez ludową tradycję, tworzenie plików `TFM` przebiegało bezboleśnie, i używanie ich ze „starym” Ghostscriptem nie wróżyło niebezpieczeństw. Niestety, fonty nie były formalnie poprawne, co powodowało, że niektóre RIP-y (interpretery POSTSCRIPT naświetlarek) sygnalizowały błąd podczas ich przetwarzania; również wersja 5.10 Ghostscriptu nie akceptuje tych fontów.

  Niepoprawność formalna polegała na braku w tych fontach *wymaganego* przez dokumentację Adobe podprogramu (znaku) o egzotycznej nazwie `.notdef`. Podprogram ten jest używany, gdy z jakichś powodów program POSTSCRIPT-owy odwołuje się do znaku, którego nie ma w foncie; w większości fontów jest to znak nie zawierający żadnego elementu graficznego, odpowiadający w zasadzie spacji.

  Było też drugie wydanie pierwszej płyty, „poprawione” w duchu opisanej niżej drugiej płyty.
- Z drugą płytą (cena ok. 450 zł) było lepiej jeżeli chodzi o znak `.notdef`, natomiast gorzej, jeśli chodzi o nazwy znaków. Oto errata:

JEST	WINNO BYĆ
ordmasculine	ae
ordfeminine	AE
tilde	ecircumflex
circumflex	Ecircumflex
onequarter	ntilde
logicalnot	Ntilde
c157	oe
c141	OE
cent	questiondown
florin	Ydieresis
c142	Zcaron
c158	zcaron
onesuperior	aogonek
yen	Aogonek
ae	cacute
AE	Cacute
ecircumflex	eogonek
Ecircumflex	Eogonek
threesuperior	lslash
sterling	Lslash
ntilde	nacute
Ntilde	Nacute
oe	sacute
OE	Sacute
questiondown	zdotaccent
macron	Zdotaccent
Ydieresis	zacute
c143	Zacute

Indagowany na tę okoliczność autor polonizacji Jakub Tatarkiewicz stwierdził, że tak jest dobrze, i że nie będzie dla paru zapaleńców przygotowywał specjalnej „TEX-owej" (czyt.: poprawnej) wersji.

- Z trzecią płytą (cena ok. 300 zł) jest zdecydowanie najgorzej, bo firma ThETA nie dostarczyła plików PFM, poprzestając na plikach PFB i PFM.

Na szczęście dla TEX-owców „podłatanie" fontów ThETA nie jest trudne. W archiwum GUST-u znajduje się pakiet THETAFIX (GUST/contrib/t1fonts/), korzystający ze wspomnianych programów Lee Hetheringtona oraz programu AWK (w wersji Gnu), dokonujący stosownych zmian w plikach składających się na fonty POSTSCRIPT-owe z dwóch pierwszych płyt. Trzecią płytę można podłatać za pomocą wspomnianego pakietu P2A.

Po takich poprawkach i utworzeniu plików TFM większość fontów powinna się dać bez kłopotów używać w dokumentach TEX-owych. Niestety większość, a nie wszystkie, bowiem kilkaset fontów z pierwszych dwóch płyt można pogrupować w kilkadziesiąt różnych klas, w zależności od zestawu znaków. Pakiet THETAFIX jest przystosowany do jednej z klas, tej najbardziej licznej. Ale w złośliwych przypadkach coś może nie zadziałać – kłopoty są ceną, jaką się płaci za tanie fonty... Tak, tak, paręset złotych za kilkaset fontów to stosunkowo tanio.

Podsumowanie. Warto pamiętać, że font musi ładnie wyglądać nie tylko w druku, ale i jego dość skomplikowana „wewnętrzna" struktura musi być zrobiona porządnie.

Jednym z najbardziej dających się we znaki problemów jest problem układu znaków w foncie. Staranne przemyślenie tego problemu *musi* prowadzić do wniosku, że firma Adobe już dawno temu znalazła *wystarczające* w większości przypadków rozwiązanie problemu konfigurowalności układu znaków w foncie. To bardzo mocny argument na rzecz korzystania z fontów POSTSCRIPT-owych w formacie Type 1.

Wprawdzie detale związane z instalowaniem fontów mogą się wydawać skomplikowane, ale taka jest natura rzeczy. A do czego prowadzi próba ukrycia tej natury przed użytkownikiem, wielu z nas boleśnie odczuło w przypadku korzystania fontów TrueType w systemie Windows. Wprawdzie instaluje się je bardzo prosto (albo i – nie wiedzieć czemu – nie instaluje), ale praktycznie nie ma *żadnej* możliwości rekonfiguracji. Ktoś, kto przenosił fonty TrueType z jednej wersji systemu Windows do innej, wie o czym mowa – a to font się w ogóle nie daje zainstalować, a to inne znaki się pojawiają, niż byśmy chcieli, a to nasza ulubiona aplikacja albo tego fontu nie widzi, albo się na nim wywraca...

Mając to na względzie uważamy, że warto poświęcić nieco czasu i wysiłku na zrozumienie podstaw konfigurowania fontów, zwłaszcza fontów POSTSCRIPT-owych w formacie Adobe Type 1 – to naprawdę w pewnym momencie staje się proste, a tym samym oczywiście niezwykle użyteczne.

◇ Bogusław Jackowski
B.Jackowski@gust.org.pl
Stanisław Wawrykiewicz
StaW@gust.org.pl