Narzędzia TeX-owe w XP, czyli jak ugryźć kaktus

Paweł Jackowski
GUST
pawel.jackowski@wp.pl

Pracę zgłosił: Andrzej Borzyszkowski

Streszczenie

W błędzie jest ten kto sądzi, że przejście z systemów Windows 95/98/Me na systemy Windows NT/2000/XP musi oznaczać katastrofę dla środowiska TeX-owego. Zawiedzie się też ten kto liczy na to, że zmiana systemu operacyjnego odbędzie się całkiem bezboleśnie. Sporo niespodzianek czeka szczególnie tych, którzy chcą używać świeżych wersji systemów Windows, nie rezygnując z tradycji korzystania w środowisku TeX-owym z wiersza poleceń.

Początki są proste. Na wstępie dobre wieści. Jeżeli ktoś używa równolegle dwóch systemów Windows, np. Windows 98 i Windows XP, i chce korzystać z TeX-a w jednym i w drugim, nie ma ku temu żadnych przeciwskazań. Jeżeli TeX jest zainstalowany i poprawnie skonfigurowany w jednym z nich, w drugim wystarczy jedynie utworzyć lub uzupełnić zmienne środowiskowe, wykorzystywane przez daną dystrybucję TeX-a. W przypadku nowszych dystrybucji TeX Live są to zmienna o nazwie `TEXMFCNF` wskazująca lokalizację głównych plików konfiguracyjnych oraz zmienna `TEXMFTEMP` odwołująca się do kartoteki tymczasowej. Modyfikacji wymaga zmienna `PATH`, w której trzeba dopisać ścieżkę dostępu do binariów instalacji (domyślnie `...\TeXLive\bin\win32`). Tyle wystarczy, by system składu TeX działał poprawnie w obydwóch systemach w tej samej konfiguracji, z tym samym zestawem makr i fontów. Skoro wszystko działa, to gdzie ten kaktus?

Zmiana w zmiennych. Ktoś, kto używa TeX-a, ale bez szerokiej gamy dodatkowych narzędzi, może się więcej ze zmiennymi nie spotkać. Niespodzianki czekają jednak tych, którzy używają rozmaitych programów narzędziowych, a także urozmaicają środowisko własnymi. Weźmy pod lupę prościutki skrypt napisany w języku `AWK`:

```
BEGIN{
 print ENVIRON["PATH"]
}
```

W systemie Windows 98 program wypisze do standardowego wyjścia ciąg znaków przechowywany w zmiennej systemowej `PATH`. A co się stanie, gdy program uruchomimy w Windows XP? Na wyjściu dostaniemy napis pusty. W XP nie istnieje bowiem zmienna `PATH`. Istnieje zmienna `Path`, a dla interpretera `AWK` stanowi to różnicę. Z krótkiego wywiadu przeprowadzonego przez autora wśród użytkowników różnych systemów Windows wynika, że Windows 95/98/Me posługują się zmienną `PATH`, zaś wersje NT/2000/XP używają zmiennej `Path`. Po co ta zmiana? Ktoś ma pomysł?

Z wywiadu wynika również, iż pod względem zmiennych systemowych i całego środowiska `MS-DOS`, systemy 95/98/Me są do siebie podobne. Istotne różnice pojawiły się w Windows NT, którego cechy odziedziczyły z kolei Windows 2000 i XP. Z przyczyn technicznych autor nie jest w stanie szczegółowo sprawdzić prawdziwości zawartych w artykule informacji dla wszystkich gatunków i ras Windows wraz z ich wersjami językowymi, stąd też w tekście odwołania głównie do Windows 98 i XP. W ogólności można jednak przyjąć, że to, co zadziała w Windows 98, zadziała też w 95 i Me, zaś informacje dotyczące XP stosują się również do wersji NT i 2000.

Interpretery poleceń. Pewnym pocieszeniem dla borykających się z nazwami zmiennych może być fakt, że program wsadowy postaci

```
echo %pAtH%
```

zadziała niezależnie od używanego systemu. Programy wsadowe przetwarzane są przez interpreter wiersza poleceń, a ten, w każdej swej odmianie, ignoruje wielkość znaków. Nie oznacza to jednak, że interpretery poleceń są ze sobą zgodne i działają tak samo w każdej wersji systemu Windows. Tutaj ten kaktus kolców ma najwięcej.

Domyślny interpreter poleceń używany w systemach Windows 95/98/ME nazywa się `COMMAND.COM`, a ścieżka dostępu do niego określona jest przez zmienną `COMSPEC`. W systemach NT/2000/XP scieżka dostępu do interpretera wiersza poleceń przechowywana jest w zmiennej `ComSpec`, a sam program nazywa się `cmd.exe`. W XP znajdziemy także interpreter `command.com` (do wyboru, do koloru). Ten ostatni stosuje tzw. krótkie nazwy ścieżek. Poza tym działa tak samo, jak `cmd.exe`... Co nie znaczy, że działa podobnie do interpretera w Windows 98. To byłoby za proste.

Dobrą ilustracją problemów, na jakie może się nadziać użytkownik środowiska MS-DOS, są próby przenoszenia programu TOIL pod nowsze platformy Windows. Pakiet TOIL autorstwa firmy BOP, przeznaczony do instalacji fontów Type1 dla potrzeb składu TEX-owego, został udostępniony na konferencji BachoTEX 1997. Dwa lata później wprowadzone zostały ostatnie poprawki do wersji dystrybucyjnej, umożliwiające korzystanie z pakietu w systemach Windows 95/98. Dziś te systemy nie w modzie, a TOIL, jako program wielofunkcyjny i od początku przeznaczony do pracy wsadowej, musiał się tu i ówdzie potykać o niezgodności w kolejnych wersjach systemu MS-DOS.

Główną czynnością wykonywaną przez TOIL-a jest przetworzenie pliku metrycznego AFM na kod zrozumiały dla METAFONT-a. Czyni to skrypt napisany w języku AWK, wykonywany przez interpreter GAWK. METAFONT z kolei ma za zadanie wygenerować metrykę TEX-ową fontu, czyli plik TFM. Do zadań dodatkowych należy utworzenie plików potrzebnych sterownikowi DVIPS, czyli pliku kodowania oraz pliku `psfonts.map` (może mieć różne nazwy, więc nazywajmy go umownie fontmapą). Na końcu TOIL przenosi efekty swej pracy do miejsc określonych przez użytkownika, korzystając ze standardowych funkcji interpretera wiersza poleceń.

Tak się nieszczęśliwie składa, że na każdym z tych kroków, we współczesnych systemach operacyjnych, coś może nie działać. Po pierwsze używany przez TOIL-a interpreter GAWK może z różnych przyczyn szwankować w nowszych systemach Windows. Autor używa starego, lecz wciąż doskonale sprawdzającego się interpretera GAWK-WIN w wersji 3.0.2. Ten krok jest jeszcze w zasadzie bezbolesny. Gorszy kolec sterczy przy programie METAFONT, którego nie daje się w XP uruchomić poleceniem w rodzaju

```
mf386 &plain plik.mf
```

jak dyktują ustawienia domyślne TOIL-a i konwencja przyjęta w programach D. E. Knuth'a. Znak & pełni w Windows XP specjalną rolę i powyższe wywołanie powoduje tyle samo, co uruchomienie METAFONT-a bez żadnego argumentu. Dzieje się tak dlatego, że interpreter poleceń systemu XP traktuje znak & jako separator kolejnych komend wywoływanych w jednym wierszu. Powyższe wywołanie oznacza dla systemu: „wykonaj polecenie/program `mf386` (bez argumentu), następnie wykonaj polecenie/program `plain`, którego argumentem jest `plik.mf`”. Zatem ani METAFONT, ani METAPOST, ani TEX wywołane w ten sposób, nie skompilują zadanego pliku. Jeśli konieczne jest odwołanie do formatu, można rozwiązać problem zamykając to odwołanie cudzysłowami, np:

```
mf386 "&plain" plik.mf
```

Jeżeli nie zależy nam na zgodności wstecz, w XP można także poprzedzić znak & znakiem ^:

```
mf386 ^&plain
```

W dystrybucji TEX Live dodanie argumentu `&plain` nie jest konieczne, dołączeniem formatu zajmuje się sam program `mf.exe`. Użytkownicy tej dystrybucji mogą uruchamiać program METAFONT prostym poleceniem:

```
mf plik.mf
```

Nie tylko znak & niesie niespodzianki. Operację neutralizacji specjalnych funkcji będziemy musieli zastosować wszędzie tam, gdzie w wierszu poleceń pojawią się w roli normalnych znaków inne metaznaki:

```
| ( ) < > ; , ^
```

Idźmy dalej! Wiele programów narzędziowych i skryptów wsadowych do przenoszenia plików używa konstrukcji

```
echo F | xcopy plik.1 plik.2
```

Takiego rozwiązania używa również TOIL. Prosty trick polega na przekazaniu w potoku znaku F do funkcji `xcopy`, co uprzedza pytanie interpretera poleceń, czy obiekt docelowy ma być plikiem, czy kartoteką. W polskich wersjach systemów Windows 95/98 znak F (ang. *file*) musi być zmieniony na P (jak *plik*).

Jeśli natomiast staliśmy się szczęśliwymi posiadaczami systemu Windows XP, należy wrócić do znaku F. Niezależnie od wersji językowej, dla interpretera poleceń Windows XP znak F oznacza plik. Zarówno w wersji angielskojęzycznej jak i polskojęzycznej, składnia poleceń jest identyczna, różnica polega tylko na języku, w którym wyświetlane są komunikaty interpretera wiersza poleceń. Wygląda to na krok w kierunku unifikacji. Rychło w czas! Póki co, programy wykorzystujące takie konstrukcje, mogą niestety wymagać osobnej konfiguracji dla potrzeb polskiej wersji językowej Windows 95/98 i osobnej dla potrzeb dowolnej wersji językowej Windows XP.

Zdarzają się też sytuacje, w których polecenie xcopy używane jest kilka razy z rzędu w tej samej postaci. W przypadku TOIL-a taka sytuacja wystąpi wtedy, gdy instalujemy kilka fontów po sobie, w wyniku czego fontmapa jest kilkakrotnie modyfikowana i nadpisywana. Jeśli używamy Windows XP, trzeba dopisać do funkcji xcopy parametr /Y:

```
echo F | xcopy plik.1 plik.2 /Y
```

Dodatek ten umożliwia ewentualne „ciche” nadpisanie pliku przy wywołaniu polecenia xcopy. Rzecz wydaje się banalna, choć ma niebanalny kontekst. W systemach Windows 95/98 polecenie xcopy domyślnie nadpisuje każdy plik docelowy, jeśli ten istnieje. W systemach NT/2000/XP przyjęto zasadę „ostrożności nigdy za wiele”, stąd też operacja nadpisania jakiegoś pliku zostanie poprzedzona monitem o potwierdzenie lub po prostu nie zostanie wykonana z powodu odmowy dostępu.

Małe i wielkie. Proponuję mały teścik! Skopiujmy w XP ze dwa fonty Type1 wraz z metrykami do plików aaa.pfb i aaa.afm oraz BBB.PFB i BBB.AFM i umieśćmy je w bieżącej kartotece. Utwórzymy prosty dokument TEX-owy postaci

```
\nopagenumbers
\font\a=aaa at 20pt
\font\B=BBB at 20pt
\a Pchnąć w~tę Łódź jeża\par
\B Pchnąć w~tę Łódź jeża
\end
```

Poddajmy TOIL-a próbie. Aby uruchomić program w klasyczny sposób, wystarczy wywołać główny plik wsadowy pakietu – a2tqx.bat – z argumentem, którym jest nazwa pliku metrycznego instalowanego fontu. Aby jednak zainstalować wszystkie fonty w bieżącej kartotece od razu, utwórzmy prosty skrypt wsadowy

```
FOR %%x IN (*.afm) DO CALL a2tqx.bat %%x
```

Kiedy TOIL zrobi wszystko co do niego należy, skompilujmy testowy plik TEX-owy. Proszę nie liczyć na moc wrażeń, w XP wszystko zadziała.

Jeżeli jednak tę samą próbę instalacji, dla tych samych plików, przeprowadzimy w Windows 98, poprawnie zadziała TOIL, dobrze spisze się TEX, niemiłą niespodziankę szykuje natomiast DVIPS, który nie znajdzie w fontmapie informacji o foncie aaa. Wszystko natomiast przebiegnie poprawnie dla fontu BBB. Dlaczego tak się stanie?

To bynajmniej nie jest wina TOIL-a! Problem leży w różnej interpretacji nazw plików w systemach 98 i XP. Jeżeli w środowisku XP dowolny plik nazwiemy małymi literami, w Windows 98 ten sam plik figurował będzie pod tą samą nazwą, tylko pisaną wielkimi literami. Mowa tu nie o tym, co wyświetlają aplikacje okienkowe lub konsole typu FAR czy Windows Commander. Chodzi o nazwę pliku, jaką zwróci intepreter wiersza poleceń w przytoczonej konstrukcji z zastosowaniem polecenia FOR lub po prostu przy poleceniu DIR. W naszym przypadku plik aaa.afm został w XP wykorzystany przez TOIL-a do wygenerowania pliku aaa.tfm, a do fontmapy wpisana została nazwa aaa. W Windows 98 metryka fontu ma nazwę AAA.AFM, TOIL tworzy metrykę TEX-ową o nazwie AAA.tfm (rozszerzenie nie ma żadnego znaczenia), a w fontmapie zostaje umieszczona nazwa AAA. W pliku TEX-owym zadeklarowaliśmy przecież font o nazwie aaa i ta sama nazwa znalazła się z pliku DVI. DVIPS rozróżnia małe i wielkie znaki, toteż szuka w fontmapie nazwy identycznej jak ta, która znajduje się w przetwarzanym pliku DVI. Nie znajduje i stąd cały ambaras!

Należy podkreślić, że w tym wypadku istotna jest zgodność nazwy fontu zadeklarowanego w źródle TEX-owym z nazwą fontu znajdującą się w fontmapie, a nie same nazwy plików. Interpreter poleceń, który pośredniczy w tym całym zamieszaniu, wykona przecież tę samą operację dla pliku o nazwie aaa.afm i dla pliku o nazwie AAA.AFM. W XP wywoła plik wsadowy a2tqx.bat z argumentem aaa.afm, a w Windows 98 z argumentem AAA.AFM. TOIL natomiast małe i wielkie znaki rozróżnia doskonale i zakłada, że w argumencie wywołania pliku wsadowego podaliśmy właśnie taką nazwę, jaką chcemy mieć w fontmapie. Dlatego konsekwentnie się jej trzyma robiąc DVIPS-owi psikusa.

Niespodzianek związanych z nazwami plików może być więcej, więc warto prześledzić ten wątek dokładniej. Lewa kolumna poniższej tabelki przedstawia pliki, którym nazwy nadano w systemie XP. Nieważne, czy stanie się to przy użyciu wiersza poleceń, czy za pomocą klikania. Ważne, że nazwanie pliku odbywa się w XP. Prawa kolumna tabelki pokazuje, w jaki sposób te same pliki zostaną odczytane w Windows 98.

XP	98
mmmm.mmm	MMMM.MMM
DDDD.DDD	DDDD.DDD
mmmm.DDD	MMMM.DDD
DDDD.mmm	DDDD.MMM

No i jak to ugryźć? Zły czar pryśnie, jeśli będziemy stosować nazwy plików dłuższe niż 8 znaków lub rozszerzenia dłuższe niż 3 znaki. To raczej kiepska rada. Problem znika również wtedy, gdy w nazwie umieścimy polskie znaki, albo gdy nazwiemy plik zarówno dużymi, jak i małymi literami. To też dość pokrętny sposób. Nie ma żadnych różnic w wielkości znaków między nazwami plików utworzonych w Windows 98, a tym, jak je „widzi" Windows XP – magia działa tylko w jedną stronę i tylko dla plików nazwanych małymi literami. To też nie powód do radości, bo walczymy o pełną zgodność środowiska TEX-owego z każdym pokoleniem Windows.

W przypadku instalacji fontów Type1 jednym z sensownych wyjść jest skorzystanie z opcji, którą umożliwia TOIL, czyli wpisywanie do fontmapy nazw wszystkich TFM-ów wielkimi literami w połączeniu z konsekwentnym stosowaniem wielkich liter w nazwach fontów deklarowanych w dokumencie TEX-owym. Druga możliwość uniezależnienia się od kaprysów interpreterów wiersza poleceń to stosowanie w fontmapach podwójnych wpisów – małymi i wielkimi literami – dla każdego instalowanego fontu.

Im dalej w las... tym więcej kaktusów! Do listy kłopotów, które może spotkać miłośnik TEX-a pracujący w nowszych systemach Windows dopisać można jeszcze wiele punktów. Z ważniejszych na pewno wymienić należy brak funkcji CHOICE, wykorzystywanej w starszych systemach do zczytywania znaków z klawiatury podczas działania programu wsadowego. Interpreter wiersza poleceń cmd.exe nie ma tej funkcji. Chwila grozy może spotkać tych, którzy w XP chcieli użyć funkcji CONVERT programu ImageMagick – doskonałego skądinąd narzędzia do wsadowej obróbki plików graficznych. Polecenie CONVERT zarezerwowane jest przez system do konwertowania systemu plików FAT na NTFS. Konwersja taka teoretycznie nie jest groźna dla danych, sprawia jednak, że dysk staje się zupełnie nie widoczny dla systemów Windows 95/98. Może ta lista ma swój koniec, ale obawiam się, że nie w tym miejscu.

Po co gryźć ten kaktus. To istotne pytanie, skoro nowe techonologie firmy Microsoft szykują użytkownikom TEX-a tyle niespodzianek. Warto jednak wspomnieć też o pozytywnych stronach zagadnienia.

Krokiem naprzód jest na pewno pełna możliwość korzystania w XP z fontów Type1 w aplikacjach okienkowych, bez pomocy dodatkowych narzędzi. Również nowalijki interpretera poleceń mogą być kuszące. Składnia poleceń została wzbogacona o wiele wygodnych rozwiązań. Polecenie FOR rozbudowane zostało o iteracje na zmiennych liczbowych. Umożliwia to wygodne stosowanie konstrukcji pętli w skryptach wsadowych. Potrafi także analizować i przetwarzać pliki tekstowe lub dane z potoku wg określonych przez użytkownika separatorów pól i rekordów. Zostało także ulepszone przetwarzanie parametrów wywołania pliku wsadowego. Oto kilka przykładów:

%*	oznacza wszystkie argumenty
%~f1	rozwija %1 do pełnej ścieżki
%~d1	rozwija %1 do litery dysku
%~n1	rozwija %1 do nazwy pliku
%~x1	rozwija %1 do rozszerzenia
%~s1	stosuje krótkie nazwy w~ścieżce
%~a1	rozwija %1 do atrybutów pliku
%~t1	rozwija %1 do daty
%~z1	rozwija %1 do rozmiaru pliku

Jeśli ktoś nie pokochał starego i solidnego GREP-a rozpowszechnionego w świecie Unix-owym, w XP jest dla niego alternatywa w postaci działającego z wiersza poleceń programu FINDSTR przeszukującego pliki w oparciu o wyrażenia regularne. Trudno to nazwać wynalazkiem teraz, kiedy GREP obchodzi swoje 30 urodziny, jednak zawsze lepiej późno, niż wcale. Dorzucić można dzwonki i gwizdki, takie jak chroniona pamięć, wbudowany generator liczb pseudolosowych, możliwość uruchomienia skryptu wsadowego o określonej godzinie i dacie, możliwość warunkowego uruchomienia skryptu czy polecenia w zależności od tego, czy zadziałał poprawnie skrypt poprzedni, a dla estetów także możliwość ustawienia kolorków tekstu i tła konsoli.

W praktyce interpreter XP jest zgodny ze starszymi wersjami – szansa natrafienia na kolec jest mimo wszystko nieduża. Dysponuje natomiast znacznie większą gamą możliwości. Zasadniczo został więc ulepszony i rozbudowany. Jeśli dodać do tego niezłe możliwości konfiguracji podsystemu DOS, okazuje się, że jest to środowisko całkiem wygodne dla zastosowań TEX-owych. Oczywiście tylko wtedy możemy stosować wszystkie dzwonki i gwizdki, jeżeli nie zależy nam na zgodności z zabytkowymi systemami Windows z końca XX wieku. Jeżeli jednak potrzebujemy takiej zgodności, najlepsze, co można powiedzieć o systemach wieku XXI, to to, że wciąż daje się je pożenić ze środowiskiem TEX-owym. Czasem nawet warto – nie taki kaktus straszny. . .