PLAIN

Tam gdzie minus oznacza dzielenie

Bogusław Jackowski
i Marek Ryćko

Ten nieco przewrotny tytuł dotyczy oczywiście przenoszenia, czyli inaczej dzielenia wyrazów, zwyczajowo zaznaczanego w tekstach kreską przypominającą minus.

Automatyczne dzielenie wyrazów należy do podstawowych zadań wykonywanych przez programy komputerowe służące do składania tekstów.

Jednym z najciekawszych algorytmów dzielenia wyrazów, a przy tym dokładnie opisanych i udokumentowanych, posługuje się system TEX. Jego „ciekawość” polega na tym, że jest to algorytm uniwersalny, tzn. dający się przystosować do dzielenia wyrazów w różnych językach, a jednocześnie zdumiewająco prosty i efektywny w implementacji. Warto się temu algorytmowi przyjrzeć.

Reguły dzielenia wyrazów w języku polskim

Żeby zrozumieć, jak trudne zadanie postawił przed sobą Knuth, zajmijmy się przez chwilę regułami dzielenia wyrazów w języku polskim. M. Szymczak („Słownik ortograficzny języka polskiego”, PWN, Warszawa, 1986) pisze:

> „(...) Dzielenie wyrazów przy przenoszeniu wyrazów z jednego wiersza do następnego opiera się w ortografii polskiej na dwóch kryteriach: fonetycznym i morfologicznym. Kryterium fonetyczne nakazuje przenoszenie części wyrazu z jednego wiersza do drugiego zgodnie z podzielnością na sylaby, np. *bu-rza*, *nie-wy-trzy-ma-nie*, *li-tość*, *po-na-pi-sy-wa-li-by*, *za-nie-po-ko-jo-ny*, *wy-so-ko*, *le-piej*, *te-go*, *prze-ciw*. Kryterium morfologiczne nakazuje przenoszenie części wyrazu z jednego wiersza do drugiego zgodnie z podzielnością na przedrostek i rdzeń, a w wyrazach złożonych — w miejscu złożenia, np. *przed-murze*, *pod-punkt*, *wy-brzeże*, *roz-łąka*, *od-dźwięk*, *przy-kład*, *za-płata*, *na-trafić*, *wy-ścigi*, *o-brona*, *bez-wstyd*, *u-kładowy*, *po-dróż*, *pod-oficer*, *pod-oddział*, *na-przód*, *pod-o-pieczny*, *po-dmuch*, *noc-leg*, *konio-krad*. (...)”

przy czym

> „(...) kryterium morfologiczne jest nadrzędne w stosunku do kryterium fonetycznego (...)”

Zasady ogólne wyglądają prosto i przejrzyście. Omówienie zasad szczegółowych zajmuje Szymczakowi pięć stron i o ile człowiekowi mogłoby to wystarczyć, to jednak zalecenia takie jak:

> „(...) Jeżeli przejrzystość podziału na przedrostek i rdzeń jest w ***dzisiejszej świadomości językowej zatarta*** (podkr. aut.), grupy spółgłoskowe znajdujące się na granicy cząstek dzielimy dowolnie, np. *o-błok* a. *ob-łok*, *o-tręby* a. *ot-ręby*, (...)”

bardzo trudno zalgorytmizować.

Spróbujmy się zastanowić, czy nie ma takich zasad, od których nie byłoby wyjątków.

Dobrym kandydatem wydawać by się mogła reguła niedzielenia głosek *cz*, *dz*, *rz* oraz *sz*. Niestety, są wyjątki od tej reguły: *tysiąc-złotowy*, *pod-zwrotnikowy*, *mar-znąć* i *em-es-zet*. Podobnie miękkie *ni* może ulec rozbiciu, na przykład w słowie *in-iekcja*.

To może chociaż da się ustalić, że sylaba nie może składać się z samych spółgłosek? Jest to w zasadzie dobra reguła, ale ma (co najmniej) jeden wyjątek: szkocki przedrostek *Mc* wolno oddzielić. Na przykład nazwisko *McMillan* wolno podzielić *Mc-Millan*. Oczywiście implikuje to kłopoty z rzymskim zapisem liczb — rok *MCMXCII* komputer mógłby uznać za szkockie nazwisko.

W takim razie może istnieją nie zakazy, a nakazy dzielenia, od których nie ma wyjątków? Np. nakaz rozdzielania par identycznych spółgłosek (*świec-cy*, *Jagieł-ło*, *łam-my*, *pan-na*, *ar-ras*, *las-so*, *get-to* itp.). Odpowiedź, jak łatwo było przewidzieć, brzmi: „nie”. Np. kryterium morfologiczne nakazuje podział *ode-ssie*, a nie *odes-sie*, chyba że chodzi o *Odessę*. A jeśli się zgodzić, że przedrostek rodzimy musi być oddzielony, to będzie kłopot ze słowem *zeppelin*, bo jak komputer miałby się domyślić, że w tym przypadku nie chodzi o przedrostek *ze*? Mógłby wprawdzie wiedzieć, że żadne słowo w języku polskim nie zaczyna się od *pp* (chociaż jest słowo zaczynające się od trzech spółgłosek, z czego dwie pierwsze to *ww* — proszę zgadnąć jakie to słowo). Ale jeśli jakiś autor chciałby oddać mowę jąkającego się bohatera pisząc np. *zeppsuuty*, to co wtedy?

Niejednoznaczności typu „*odessie*” stanowią dla komputera trudność praktycznie nie do pokonania.

Choć jest ich niewiele, to jednak zdarzają się i utrudniają życie, np.: *podróżować* — być w podróży lub pokryć różem lica; *odziewać* — ubierać lub odpowiedzieć w taki sam sposób komuś, kto ziewnął; *podrobić* — np. karmę ptactwu lub podpis; *Tarzanie* — słowo to może oznaczać wzywanie znanego bohatera komiksów lub czynność tarzania się; *narwali* — to słowo z kolei można rozumieć jako czas przeszły dokonany albo dopełniacz liczby mnogiej.

W sytuacjach tego typu jedynie kontekst semantyczny decyduje o podziale słowa. Rozsądnie jest nie żądać zbyt wiele od programów składających teksty i zostawić człowiekowi możliwość ingerencji w bardziej skomplikowanych przypadkach.

Sporo trudności nastręcza dzielenie nazwisk i nazw własnych. O nazwiskach szkockich już była mowa. Ale i polskie nazwy potrafią sprawić kłopot. Na przykład czy w nazwisku *Utnik* głoskę *u* należy traktować jako przedrostek, tak jak w słowie *u-tnie*? Szczególnie trudne do podziału są słowa będące zbitką dwóch lub więcej słów. Zalecany jest oczywiście podział w miejscu złożenia: *noc-leg*, *trzech-set-letni* itp. Zdarzają się jednak czasem bardzo osobliwe nazwy własne, np. *Wierzch-las*. Chociaż taki właśnie podział wydaje się uzasadniony, to przecież trudno dać głowę, że nazwy tej nie da się wywieść od *wierzenia* i *chlastania*. Podobnie gdyby jakaś miejscowość nazywała się *Szynkwas*, to nazwa mogłaby się wywodzić ostatecznie — choć jest to mało prawdopodobne — od *szyn* i *kwasu*.

Zeppelin i *szynkwas* to przykłady słów obcych bądź obcego pochodzenia, które zadomowiły się w naszym języku. Słowa tej kategorii, takie jak np. *er-zac*, *jazz-man* czy *soft-ware*, to źródło nowych utrapień dla kogoś, kto chciałby sformalizować reguły dzielenia wyrazów w języku polskim.

W tej sytuacji nie dziwią wyniki testów porównawczych W. Puzy (W. Puza, praca magisterska pt. „Analiza porównawcza wybranych programów Desktop Publishing (DTP)", Instytut Poligrafii Politechniki Warszawskiej), który stwierdził, co następuje:

> „(...) Test sprawdzający poprawność dzielenia polskich słów wyodrębnił 3 różne poziomy skuteczności testowanych programów:
>
> - Cyfroset, TEX – ok. 10 % błędów
> - Dywiz (dedykowany dla Quark'a) – powyżej 20 % błędów
> - Bizon (dedykowany Calamusowi), VP Commander (dedykowany Venturze) – powyżej 50 % błędów. (...)"

Tak więc pozornie trywialny problem okazuje się w praktyce trudnym orzechem do zgryzienia.

Można oczywiście polemizować z wytycznymi językoznawców. Zalecany przez Szymczaka podział *za-jrzeć* lub *po-jmać* wydaje się dziś anachroniczny. Pójście dalej tym tropem, przy równoczesnym zasłanianiu się „wymaganiami komputerów", powoduje, że zbyt kuszące staje się maksymalne upraszczanie reguł, oczywiście z uszczerbkiem dla języka polskiego.

Autor TEX-a obrał zupełnie inną drogę: szukał takiego algorytmu, który byłby w zasadzie niezależny od reguł dzielenia wyrazów; reguły dzielenia miałyby być określone za pomocą danych do tego algorytmu. Algorytm spełniający te wymagania skonstruował F. M. Liang (rozprawa doktorska pt. „Word Hy-phen-a-tion by Com-put-er", Uniwersytet Stanforda, USA, 1983). Udało mu się nawet coś więcej: dane są stosunkowo łatwe do utworzenia, nie jest do tego bynajmniej potrzebna wiedza informatyczna.

Jak TEX dzieli wyrazy

Algorytm dzielenia wyrazów jest niezwykle prosty. Oczywiście przy założeniu, że odpowiednie dane zostały utworzone. Załóżmy więc, że te dane mamy. Składa się na nie zbiór ciągów utworzonych na przemian z liter i cyfr, np.: `2s0z1z0`, `2s0z0l0n0`, `2t1ć0` itp. Zamiast cyfry skrajnej może wystąpić kropka, np.: `.w0e3s2`, `.w0w8`, `8r0s0z.`, `8r0z0ł.` itp. Ciągi te nazywać będziemy wzorcami.

Przypuśćmy teraz, że mamy dane jakieś słowo, np. `odkaszlnąć`. Najpierw zamieniamy je na postać taką, jaką mają wzorce, wstawiając pomiędzy literami cyfrę `0`, a na brzegach kropki, otrzymując `.o0d0k0a0s0z0l0n0ą0ć.` (kropka — jak stąd widać — oznacza skraj słowa). Następnie wyszukujemy w naszym zbiorze danych wszystkie wzorce, które na to słowo dają się nałożyć w taki sposób, że zgadzają się położenia liter i kropek, ale niekoniecznie cyfr; w szczególności cyfra może się nakładać na kropkę, ale nie na literę. Powiedzmy, że znaleźliśmy następujące wzorce: `.o2d2`, `.o0d3k2`, `0ą1`, `2d1k0`, `2l1n0`, `2s0z1l0`, `2s0z0l0n0`, 8ć., 0a1, 0o1, 0s4z0. Jeśli je nałożymy na słowo .o0d0k0a0s0z0l0n0ą0ć. i spośród cyfr nakładających się na siebie weźmiemy największe, to otrzymamy .o2d3k2a2s4z2l1n0ą8ć. — i to już wszystko. Zgodnie z algorytmem Lianga słowo wolno podzielić jedynie w takim miejscu, w którym pojawia się cyfra nieparzysta. Oznacza to, że algorytm dopuszcza w tym wypadku dwa punkty podziału: *od-kaszl-nąć*.

Wzorce przechowywane są w pamięci komputera jako struktura drzewiasta (dokładniej *trie*; p. D. E. Knuth, „The Art of Computer Programming", tom 3, „Sorting and Searching"). Pozwala to na znaczne upakowanie danych przy zachowaniu szybkiego dostępu do informacji, a równocześnie struktura danych i algorytm są stosunkowo łatwo implementowalne.

Warto podkreślić, że dzięki efektywności zastosowanej metody TEX może przechowywać w pamięci kilka różnych zestawów wzorców i dzielić wyrazy w kilku językach w obrębie jednego dokumentu (a nawet akapitu).

Problem jedynie w tym skąd wziąć wzorce?

Automatyczne tworzenie danych dla algorytmu dzielenia wyrazów

Główna część pracy Lianga dotyczyła metody tworzenia wzorców na podstawie słownika zawierającego słowa poprawnie podzielone. Przygotowanie takiego słownika jest oczywiście czaso- i pracochłonne, ale za to dalsza część zadania jest już względnie łatwa. W największym uproszczeniu algorytm Lianga można zapisać następująco:

```
m := maksymalna długość wzorca
for c := 1 to 9 do
begin
  ustal kryteria akceptowalności dla wzorców
  for l := 1 to m do
  begin
    dla każdego słowa w słowniku sprawdź,
    czy nie dostarcza ono informacji o moż-
    liwości wstawienia cyfry c we wzor-
    cach długości l przy zadanych kryteriach
    akceptowalności
  end
end
```

Innymi słowy program uzupełnia zestaw wzorców kolejno dla coraz większych cyfr i coraz dłuższych wzorców, zostawiając użytkownikowi na każdym etapie możliwość ingerencji. Kryteria akceptowalności określa się przez podanie trzech liczb: wagi dla podziałów poprawnych w_p, wagi dla podziałów niepoprawnych w_n i poziomu akceptowalności p. Wzorzec jest akceptowany, gdy $k_p w_p - k_n w_n \geq p$, gdzie k_p i k_n oznaczają odpowiednio liczbę poprawnych i niepoprawnych podziałów generowanych przez dany wzorzec.

W niektórych przypadkach może się zdarzyć, że program nie znajdzie zestawu wzorców dzielącego poprawnie wszystkie słowa. Powodem mogą być błędne dane (np. w słowniku może się znaleźć dwa razy to samo słowo, ale inaczej podzielone) bądź niewłaściwy dobór liczb w_p, w_n oraz p w kolejnych iteracjach. Różne strategie dobierania wartości tych współczynników wpływają na objętość wynikowego zestawu wzorców. Minimalizacja liczby wzorców w wygenerowanym automatycznie zestawie wymaga pewnej wprawy i — oczywiście — przestudiowania pracy doktorskiej Lianga, gdzie zagadnienie to jest szczegółowo omówione.

Jak widać „nauczenie" TEX-a zasad dzielenia wyrazów w danym języku jest właściwie kwestią odpowiednio zasobnego słownika podzielonych poprawnie wyrazów. Dostarczenie takiego słownika to zadanie dla językoznawców. Natomiast dalszy etap jest ideowo prosty, nie wymaga znajomości technik programowania, ani żadnych innych specjalnych umiejętności. Potrzebna jest jedynie cierpliwość.

Ręczne tworzenie danych dla algorytmu dzielenia wyrazów

Możliwość automatycznego tworzenia zestawu wzorców nie wyklucza możliwości stworzenia takiego zestawu ręcznie. Początkowo wydawało się, że język polski ma na tyle regularne zasady dzielenia wyrazów, że łatwiej będzie utworzyć zestaw wzorców w ten właśnie sposób.

W 1984 J. Désarménienowi udało się zamknąć reguły dzielenia wyrazów dla języka francuskiego w zestawie liczącym 804 wzorce i dla języka włoskiego w zestawie liczącym zaledwie 88 wzorców, podczas gdy standardowy zestaw wzorców dla amerykańskiego TEX-a zawiera ich 4447 (J. Désarménien, „The use of TEX in French: hyphenation and typography", w: D. Lucarella (ed.), „TEX for Scientific Documentation, Proc. of the First European

Conference, Addison-Wesley, 1984). Sukces podejścia Désarméniena zainspirował H. Kołodziejską, która w roku 1987 opublikowała listę 2168 wzorców dla języka polskiego (H. Kołodziejska, „Dzielenie wyrazów w systemie TEX”, Sprawozdania Instytutu Informatyki Uniwersytetu Warszawskiego, nr 165, 1987).

Wynik Kołodziejskiej zdawał się potwierdzać wstępne założenia. Niestety. Po kilku latach intensywnego testowania (maczali w tym palce m. in. autorzy niniejszej pracy) wzorców znacznie przybyło. Po gruntownej przeróbce liczebność zestawu wzorców wzrosła do 4053 i wolno przypuszczać, że to jeszcze nie koniec.

W tych warunkach zasadne staje się pytanie, czy nie należałoby zrewidować założenia o regularności zasad podziału słów w języku polskim. Jest prawdopodobne, że program Lianga mógłby wygenerować mniej liczny zestaw wzorców.

Tym niemniej, jak wynikało z porównań poczynionych przez Puzę (p. wyżej), zestaw wzorców dla języka polskiego w swojej obecnej postaci daje wyniki zadowalające. Zważywszy, że Puza w istocie dysponował jedną z wersji pośrednich zestawu i że zestaw aktualny jest znacznie udoskonalony w stosunku do tamtej wersji, można oczekiwać jeszcze lepszych rezultatów.

Najaktualniejsza wersja tego zestawu wchodzi w skład pakietu MEX (polskiej adaptacji TEX-a). Pakiet ten jest dystrubuowany w postaci źródłowej jako produkt *public domain* przez GUST.

Inne aspekty automatycznego dzielenia wyrazów

Automatyczne znajdowanie miejsc podziału słów nie wyczerpuje spraw związanych z dzieleniem wyrazów do celów składu komputerowego. Jak zostało już wspomniane, czasami ingerencja człowieka staje się niezbędna. Poprawnie zaprojektowany system składu taką ingerencję powinien umożliwiać. Użytkownik powinien mieć w szczególności możliwość wprowadzenia zakazu dzielenia niektórych słów.

W przypadku TEX-a nierozsądne byłoby manipulowanie za każdym razem przy zestawie wzorców. Użytkownik TEX-a może podzielić słowo według własnego uznania na dwa sposoby.

Pierwszy sposób to umieszczenie go na liście wyjątków, nadrzędnej w stosunku do wzorców. Jest to, niestety, metoda wygodna jedynie w językach niefleksyjnych. W języku polskim umieszczenie jakiegoś słowa na liście wyjątków wiąże się (w przypadku TEX-a) z podaniem od razu wszystkich jego odmian, co jest raczej nieporęczne. Na szczęście problem taki nie pojawia się zbyt często. Głównym powodem tego typu zabiegów bywają słowa złożone lub obcego pochodzenia (bądź i jedno, i drugie).

Drugi sposób polega na wskazaniu w tekście dodatkowych miejsc podziału (ang. *discretionary hyphens*). Większość systemów DTP oferuje taką możliwość użytkownikom. W przypadku TEX-a należy ostrożnie posługiwać się tą metodą, bo można przeszkodzić TEX-owi w automatycznym wstawianiu drobnych odstępów między znakami (inaczej podcięć, ang. *implicit kerns*).

TEX oferuje użytkownikowi jeszcze parę innych możliwości, pozwalających na panowanie nad algorytmem podziału wyrazów.

Zabronienie podziału wyrazu jest w TEX-u sprawą trywialną, wystarczy wyraz „zamknąć” w tzw. pudełko (TEX-owa komenda `\hbox`).

Użytkownik może też określić minimalną długość oddzielonego początku i końca słowa. Domyślnie TEX nie podzieli słowa krótszego niż pięcioliterowe: początek nie może być krótszy niż dwie litery, koniec — nie krótszy niż trzy litery. Przy składaniu w wąskiej szpalcie może się pojawić konieczność złagodzenia tych rygorów, zwłaszcza że w języku polskim za dopuszczalne uznaje się oddzielanie pojedynczej samogłoski na początku słowa i dwuliterowej sylaby na końcu. Z drugiej strony w składzie wysokiej jakości powinno się unikać dzielenia wyrazów w ogóle, a tym bardziej na zbyt krótkie fragmenty. Możliwość sterowania wielkością dzielonego słowa jest więc bardzo przydatna w praktyce.

Zasady dobrego składu wymagają, aby wyrazy dzielone nie pojawiały się w sąsiednich wierszach i by ostatni wiersz na prawej stronie nie zawierał słowa dzielonego. Składy nie spełniające tych wymogów uważa się (i słusznie) za brzydkie. TEX dostarcza parametrów pozwalających skutecznie utrudniać brzydkie składanie. W szczególności można wręcz zabronić umieszczania słowa dzielonego w ostatnim wierszu na stronie. Może to sprawić pewne kłopoty algorytmowi łamiącemu wiersze na strony, ale to już zupełnie inna historia.

Zostaje jeszcze problem słów zawierających łącznik, np. *biało-czerwony*. W języku angielskim słowa takie jak *machine-oriented* przenosi się

machine-
oriented

podczas gdy w języku polskim stanowczo zaleca się dostawienie dodatkowo łącznika na początku następnego wiersza:

biało-
-czerwony

TEX pozwala dołączyć do zestawu komend standardowych komendy definiowane przez użytkownika. W MEX-u została zdefiniowana komenda `\=`, którą należy podczas przygotowywania tekstu umieścić w miejscu łącznika. Kontynuując przykład, *biało-czerwony* należałoby podać TEX-owi jako `biało\=czerwony`. Użytkownik musi pamiętać o konsekwentnym przestrzeganiu tej zasady, a o resztę zadba już TEX. Pozostawienie zwykłego łącznika oznacza słowo niepodzielne. Jest to przydatne w przypadku takich słów jak np. *K-202*, których oczywiście dzielić nie należy.

Uwagi końcowe

Jak widać zagadnienie automatycznego dzielenia wyrazów to problem bardzo obszerny. Wchodzą tu w grę zagadnienia językoznawcze, typograficzne i informatyczne.

Praktyka pokazuje, że bardzo efektywny algorytm automatycznego podziału słów, w połączeniu z niekoniecznie bardzo wygodnymi możliwościami ingerowania ręcznego, stwarza użytkownikowi możliwość panowania w pełni nad procesem dzielenia słów przez system komputerowego składu tekstów.

Oprogramowanie

Z notatnika oblatywacza TEX-owego

Stanisław Wawrykiewicz

Witam w nowym dziale GUST-ownego biuletynu, poświęconym nowinkom dotyczącym szeroko pojętego oprogramowania TEX-owego oraz praktyce jego konfigurowania i obsługi. Nie roszczę sobie wyłączności do pisania w tym dziale i wobec tego dołączam się do apelu PT Redaktorów o przesyłanie artykułów i notatek, poświęconych szczególnie programom działającym na innych niż IBM PC/MS DOS platformach sprzętowo-systemowych.

Na początek oprogramowanie podstawowe, czyli sam program TEX. Większość z nas używa (na komputerach PC) bardzo popularnego emTEX-a Eberharda Mattesa. Zaprezentuję tutaj krótko SB38TEX, będący implementacją TEX-a w wersji 3.141, autorstwa Wayne G. Sullivana i Petera Breitenlohnera. Program charakteryzuje się zwartym kodem, uruchamiany może być nawet na komputerach klasy XT, jest bardzo szybki i sprawniej niż standardowy emTEX radzi sobie z większą ilością fontów na stronie składu.

SB38TEX dostępny jest jako dobro wspólne (*public domain*) w pakiecie `sb38tex.zip` spakowanym programem `pkzip`. Zawiera on 31 plików zajmujących po dekompresji ok. 0.5MB, w tym: `tex.exe` — właściwy program datowany na 18.05.92 (143552 bajtów), `initex.exe` — program do generowania *formatu*, czyli pliku binarnego zawierającego definicje makr, wzorce przenoszenia itp., `plain.tex` — plik źródłowy podstawowego formatu, `tex.poo` i `nohelp.poo` — pliki unikalne dla każdej implementacji TEX-a, zawierające komunikaty generowane przez program, niezbędne przy tworzeniu formatu (`tex.poo` jest pełnym zestawem komunikatów i podpowiedzi TEX-a), `sb38tex.doc` — dokumentacja pakietu przygotowana do złożenia przy użyciu standardowego formatu `plain.fmt`, 16 plików metrycznych (z rozszerzeniem `.tfm`) podstawowych fontów TEX-a. SB38TEX pozwala na zdefiniowanie zamiany kodów znaków czytanych przez TEX-a na kody wewnętrzne, zgodnie z którymi znaki umieszczone są w plikach fontów. Definicje takie powinny być zawarte w pliku `codep.age` obecnym w bieżący katalogu podczas generowania formatu. Przykład takiego pliku znajduje się w pakiecie, zaś poprawność notacji można sprawdzić dołączonym programem `cdpgtest.exe`. Z kolei program `sb38set.exe` służy do modyfikacji pakietu w przypadku gdy będzie on instalowany na dysku innym niż `C:`. Wymagana jest wtedy obecność w bieżącym katalogu plików: `tex.exe`, `initex.exe`, `tex.poo` i `nohelp.poo`.

Dodatkowo dołączono do pakietu źródła Pascalowe `sbmenu.pas` prostego programu typu *Shell* uruchamiającego TEX-a, edytor i sterowniki (*drivers*) ekranu i drukowania. Program wymaga modyfikacji zgodnie z lokalną konfiguracją TEX-a i używanymi programami sterowników, po czym kompilacji Turbo Pascalem.

Przygotowanie formatu Plain

Przygotowanie formatu `plain.fmt` wymaga następujących plików z pakietu sb38tex: `initex.exe`, `tex.poo`, `plain.tex`, `hyphen.tex` i ewentualnie zmodyfikowanego pliku `sbcodep.age` Należy określić zmienne środowiskowe DOS-a używane przez program `tex.exe`, proponuję tutaj utworzenie katalogów o krótkich nazwach:

```
SET FMTSB=c:\tex\sbtex
SET SBFTMP=C:\;-r04
SET FONTTFMS=.;c:\tex\tfms
SET TEXINPUTS=c:\tex\lib;c:\tex\input
```

Zmienne oznaczają kolejno (w nawiasach podano wartości domyślne): `FMTSB` (`c:\tex\formats`) — katalog zawierający format(y), czyli plik(i) `*.FMT`, `SBFTMP` (`c:\;-r06`) — katalog na tworzone podczas pracy TeX-a pliki tymczasowe i sposób gospodarki pamięcią (tu maksymalne wykorzystanie pamięci na pamięć podstawową programu, parametry opisane są bardziej szczegółowo w dokumentacji), `FONTTFMS` (`c:\tex\fonttfms`) — katalog(i) z plikami metrycznymi `.TFM`, wreszcie `TEXINPUTS` (`c:\tex\inputs`) — katalog(i) przeszukiwane przez TeX-a po użyciu komendy `\input`. Katalogi standardowo oddzielamy średnikiem.

Wymienione wyżej programy najlepiej umieścić w katalogu wskazywanym przez zmienną `FMTSB`. Generowanie formatu wymaga uruchomienia polecenia:

```
initex plain \dump
```

TeX wraz z utworzonym właśnie formatem najlepiej uruchamiać za pośrednictwem pliku wsadowego, np. `t.bat`:

```
c:\tex\sbtex\tex &plain %1 %2
```

Przygotowanie formatu MeX

Korzystamy z tych samych plików i katalogu co wyżej. Dodatkowo z pakietu `mex103` niezbędne są: `mex.tex`, `mex1.tex`, `mex2.tex`, `mexconf.tex`, `plhyph.tex`, oraz wygenerowane Metafont-em pliki `*.TFM` 16 podstawowych fontów MeX-a predefiniowanych w formacie: `plr5`, `plr7`, `plr10`, `plmi5`, `plmi7`, `plmi10`, `plsy5`, `plsy7`, `plsy10`, `plex10`, `plbx5`, `plbx7`, `plbx10`, `pltt10`, `plsl10`, `plti10`. Jak widać są to odpowiedniki fontów predefiniowanych formatu plain. Modyfikujemy plik `sbcodep.age` zgodnie z wybranym „standardem” kodowania polskich liter w pliku do składu (zawartość pliku dla kodów Mazovia i Latin2 przedstawiono poniżej, należy zachować dokładną notację, włącznie z końcowymi znakami `<<`).

Ponieważ format MeX zawiera zwykle wzorce przenoszenia dla dwóch języków musimy na czas jego generowania zmodyfikować gospodarkę pamięcią, zwiększając pamięć wzorców (*trie size*). Wykonajmy polecenia:

```
SET SBFTMP=C:\;-r04;-t13
initex mex \dump
```

Uruchamiamy TeX-a za pomocą pliku, np. `mex.bat` (zakładam dostęp do samego programu wyspecyfikowany zmienną `PATH`):

```
tex &mex %1 %2
```

Poniższy skrypt (np. `plain.bat`) umożliwia emulację formatu `plain`, w formacie MeX:

```
tex &mex \emulateplain\input %1 %2
```

Po wygenerowaniu formatów zostawiamy na dysku jedynie `tex.exe`, plik(i) `*.fmt` i oczywiście pliki `*.TFM`. Ludzie, nie zaśmiecajcie sobie dysków!

Po lewej zawartość pliku `sbcodep.age` dla przekodowania znaków wejściowych w kodach Mazovia, po prawej — w kodach Latin2 (oba „widziane” we własnych środowiskach edycyjnych)

Ą=^^81	Ą=^^81
Ć=^^82	Ć=^^82
Ę=^^86	Ę=^^86
Ł=^^8a	Ł=^^8a
Ń=^^8b	Ń=^^8b
Ó=^^d3	Ó=^^d3
Ś=^^91	Ś=^^91
Ź=^^99	Ź=^^99
Ż=^^9b	Ż=^^9b
ą=^^a1	ą=^^a1
ć=^^a2	ć=^^a2
ę=^^a6	ę=^^a6
ł=^^aa	ł=^^aa
ń=^^ab	ń=^^ab
ó=^^f3	ó=^^f3
ś=^^b1	ś=^^b1
ź=^^b9	ź=^^b9
ż=^^bb	ż=^^bb
^^81=^^8f	^^81=^^a4
^^82=^^95	^^82=^^8f
^^9b=^^90	^^f3=^^a8
^^8a=^^9c	^^8a=^^9d
^^8b=^^a5	^^8b=^^e3
^^d3=^^a3	^^d3=^^e0
^^b9=^^98	^^91=^^97
^^99=^^a0	^^99=^^8d
^^f3=^^8d	^^9b=^^bd
^^aa=^^92	^^a1=^^a5
^^ab=^^a4	^^a6=^^a9
^^b1=^^9e	^^aa=^^88
^^bb=^^a7	^^b9=^^e4
<<	^^b1=^^98
	^^bb=^^be
	<<