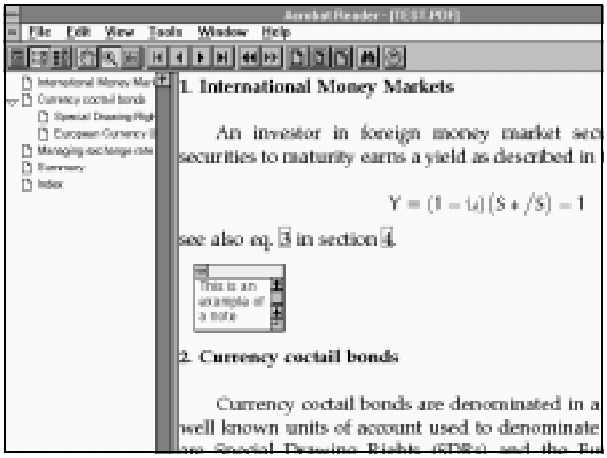
Narzędzia

Publikacje elektroniczne: TEX i PDF

Tomasz Przechlewski

TEX-owcy w odróżnieniu od różnorakiej maści wielbicieli WYSIWYG-ów brzydzą się czytania z ekranu komputerowego. Co najwyżej, na etapie tworzenia dokumentu, można rzucić nań okiem, wykorzystując tę lub inną podglądarkę ekranową, ale nic więcej. To zapewne tłumaczy małą wśród nich popularność książek elektronicznych, hipertekstu, itp. pomysłów. Ale uwaga! Ten tekst będzie o tworzeniu dokumentów i to w systemie TEX, które *należy* czytać na ekranie komputera.



Rys. 1. Przykładowy dokument oglądany w programie ACROBAT READER

Co to jest Acrobat?

ACROBAT oznacza całą grupę programów opracowanych w firmie Adobe Systems Inc., służących do tworzenia, modyfikowania, drukowania i oglądania plików zapisanych w formacie PDF (*Portable Document Format*). PDF jest formatem hipertekstowym, podobnie jak powszechnie znany i używany HTML.

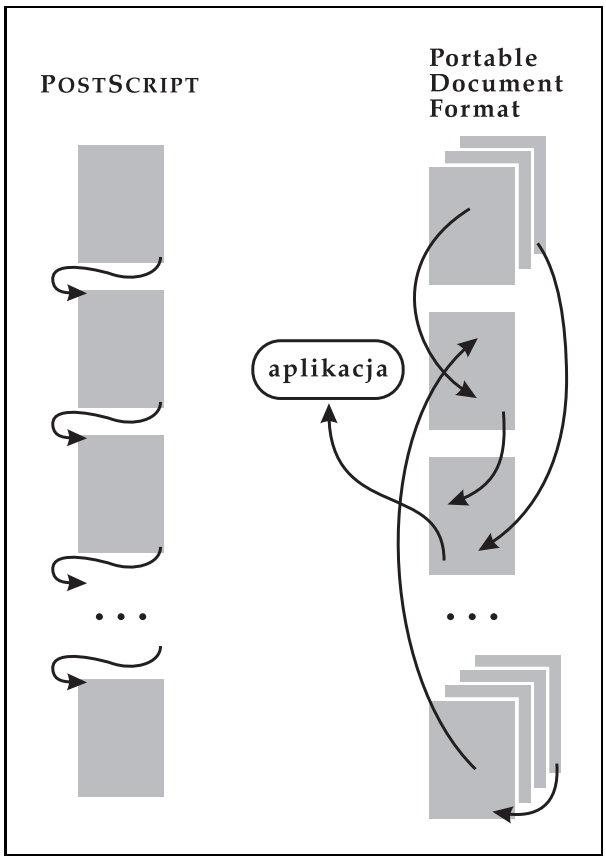
HTML opisuje dokument za pomocą jego struktury logicznej, tj. wskazuje na znaczenie poszczególnych jego elementów (np. tytuł rozdziału, tytuł punktu, itd.). Ostateczny wygląd tak zdefiniowanych elementów jest uzależniony od programu, który używamy do przeglądania (czytania). PDF nie opisuje struktury dokumentu, ale jest językiem opisu strony, podobnie jak POSTSCRIPT.

Okazało się, że HTML, jako format opisujący tylko strukturę, nie za bardzo się sprawdza w praktyce. Zapewne z tego powodu HTML komplikuje się w ostatnim czasie, a większość modyfikacji służy do definiowania (atrakcyjnego) wyglądu dokumentu. Tradycyjnie pliki HTML-owe były dużo mniejsze od odpowiadających im plików PDF, można je było przeglądać za pomocą wielu programów i łatwo tworzyć. Obecnie wszystko to jest w dużym stopniu przeszłością. Pliki HTML-owe są duże, bo przeładowane grafiką, często dużo większe od odpowiadających im plików PDF (bo te są silnie kompresowane), przeglądać je można za pomocą programu *xxx* w wersji *yyy*, w każdym innym przypadku czegoś tam nie zobaczymy. Pomimo tego HTML ciągle nie oferuje takich możliwości prezentacji jak PDF. Okazuje się, że nie jest łatwo z latawca zrobić jumbo-jeta, szczególnie metodą doklejania kolejnych łatek. Matematyka, tabele, dokumenty składane z wykorzystaniem znaków z alfabetów niełacińskich są, póki co, poza zasięgiem HTML-a. Kiedy publikacja jest skomplikowana, praktycznie nie mamy wyboru, pozostaje tylko PDF. TEX umie składać najtrudniejsze dokumenty, PDF umożliwia ich przedstawienie w postaci elektronicznej, pozostaje tylko połączyć te technologie.

Dokument zapisany w formacie PDF zawiera niezależny od konkretnego urządzenia drukującego opis każdej strony oraz dodatkowe elementy takie jak: hipertekstowe odsyłacze (*links*), notki[1] (*annotations*), zakładki hipertekstowe (*bookmarks*), miniatury stron (*thumbnails*) i inne. Oczywiście, elementy hipertekstowe mogą być użyteczne tylko w przypadku korzystania z elektronicznej postaci dokumentu, czyli w przypadku przeglądania go na ekranie komputera. Rysunek 1 pokazuje wygląd zakładek, notek i odsyłaczy w oknie programu ACROBAT READER.

Opis obiektów na stronie jest bardzo podobny do tego, który jest wykorzystywany w języku POSTSCRIPT, ale struktura dokumentu jest zupełnie inna. Dokument POSTSCRIPT-owy ma strukturę zwykłego pliku (jest opisywany strona po stronie), natomiast PDF opisuje strony, operatory hipertek-

1: Odpowiednikiem tego elementu w książce „papierowej” i zapewne natchnieniem dla autorów tego pomysłu może być żółta karteczka – „sticker” przylepiona do jakiejś strony z dodatkową informacją.



Rys. 2. Schematyczne porównanie struktury plików POSTSCRIPT-owych i plików w formacie PDF

stowe i inne elementy jako obiekty o dostępie swobodnym. Ponadto technologia PDF wykorzystuje różne rodzaje kompresji, co daje w rezultacie dużo mniejsze pliki.

ACROBAT jest technologią „w rozwoju", w tej chwili (początek grudnia 1996 r.) firma Adobe rozpoczęła dystrybucję wersji 3.0. Artykuł pisano opierając się na wersji poprzedniej (o numerze 2.1).

Tworzenie dokumentów w formacie PDF. Z reguły dokumenty przygotowywane są w takich aplikacjach, jak PageMaker, FrameMaker, Word, czy TEX, a następnie zamieniane są na format PDF. Zamiana ta może się odbywać na dwa sposoby. Prostszym jest tworzenie plików PDF bezpośrednio z aplikacji wyjściowej w sposób przypominający drukowanie. Do tego służy program PDFWRITER.

PDFWRITER działa podobnie do zwykłego sterownika drukarki, ale zamiast generować komendy specyficzne dla „jakiejś tam" drukarki, generuje plik w formacie PDF. PDFWRITER, zainstalowany np. w systemie Windows, umożliwia oczywiście w ten sposób wygenerowanie pliku PDF-owego z każdej okienkowej aplikacji. Ta droga generowania PDF-ów jest póki co dla TEX-owców nieprzydatna.

Drugi sposób generowania plików PDF polega na utworzeniu najpierw pliku POSTSCRIPT-owego. Praktycznie każdy edytor tekstu czy system składu potrafi wygenerować dokument w formacie języka POSTSCRIPT. Użytkownicy TEX-a zwykle wykorzystują do tego program DVIPS. Następnie plik POSTSCRIPT-owy może zostać zamieniony do formatu PDF za pomocą opracowanego przez firmę Adobe programu DISTILLER, który zamienia plik POSTSCRIPT-owy na plik w formacie PDF. DISTILLER jest programem droższym, ale ma większe możliwości: produkuje na ogół mniejsze pliki, lepiej sobie radzi z grafiką, umożliwia sprawniejsze wstawienie do dokumentu elementów hipertekstowych.

Zamiast DISTILLER-a można wykorzystać do destylacji GHOSTSCRIPT. Ten doskonały interpreter języka POSTSCRIPT obsługuje w wersji 4.00 lub nowszej także format PDF. Zaletą GHOSTSCRIPT-a jest to, że jest oprogramowaniem *public domain*, wadą, że na razie uzyskiwane efekty nie są najlepsze. Szerzej o wykorzystaniu GHOSTSCRIPT-a piszę dalej.

Przeglądanie dokumentów zapisanych w formacie PDF. Dokumenty zapisane w formacie PDF można przeglądać za pomocą wielu programów, dostarczonych przez różnych producentów. Niektóre z nich są darmowe, inne trzeba kupić. Najpopularniejsze programy to: ACROBAT READER oraz ACROBAT EXCHANGE, opracowane przez Adobe Systems, oraz wspomniany już GHOSTSCRIPT. READER jest nieodpłatnie udostępniany przez producenta i jest dostępny dla wielu platform systemowo-sprzętowych, GHOSTSCRIPT jest oprogramowaniem *public domain*.

ACROBAT READER umożliwia oglądanie i drukowanie plików PDF, przy czym przy przeglądaniu możliwe jest wykorzystanie wszystkich *standardowych* możliwości hipertekstowych, tzn. odsyłaczy, zakładek hipertekstowych, notek, itp.).

ACROBAT EXCHANGE jest programem komercyjnym. Umie to co READER, ale także umożliwia

edycję plików PDF: wstawianie/usuwanie odsyłaczy, zakładek, notek itp, usuwanie/dodawanie stron, łączenie dokumentów PDF-owych, obsługę wtyczek[2] (*plug-ins*).

GHOSTSCRIPT umożliwia tylko obejrzenie oraz wydrukowanie pliku PDF. Nie obsługuje żadnych elementów hipertekstowych.

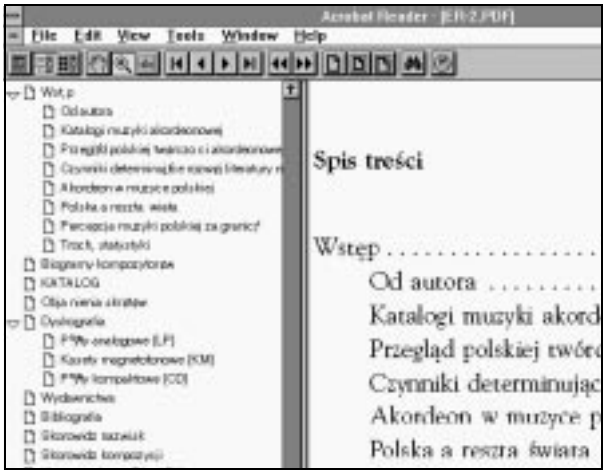
Problemy z przenośnością PDF-a. Z punktu widzenia polskiego użytkownika technologia PDF nie jest niestety idealna. Papierowa wersja dokumentu jest OK, ale jej elektroniczna wersja ma znacznie ograniczoną funkcjonalność. Ograniczenia wynikają z niedoskonałości technologii PDF, z naszego – użytkowników języka polskiego – punktu widzenia.

Przede wszystkim bardzo użyteczna funkcja wyszukiwanie w programie ACROBAT READER jest zależna od sposobu kodowania fontu (*encoding vector*). Obecnie w Polsce obowiązują przynajmniej trzy „standardy" kodowania, odpowiadające trzem platformom sprzętowym: MS Windows, Macintosh oraz systemy Unixowe. W rezultacie dokument przygotowany na Macu będzie miał polskie literki w innych miejscach niż oczekuje tego użytkownik Windowsów na pececie. Klikając w przycisk lornetki i wpisując np. „Łódź" możemy ze zdumieniem stwierdzić, że tekst nie zawiera takiego słówka, tylko dlatego, że był przygotowany innymi fontami niż ekranowe w naszym systemie.

Co więcej, nic nie stoi na przeszkodzie, aby dokument składać fontami o różnym wektorze kodowania (np. kupiliśmy dobre fonty u różnych producentów i wykorzystujemy je w jednym dokumencie). Efekt jest taki, że aczkolwiek na ekranie wyświetlane są poprawnie, to z punktu widzenia funkcji wyszukiwania tak nie jest.

Ponadto hipertekstowy spis treści jest wyświetlany fontem *wbudowanym w przeglądarkę*. Dostępny powszechnie ACROBAT READER używa do tego celu fontu, który nie ma kompletu polskich znaków diakrytycznych. Efekt można obejrzeć na rysunku 3.

Szkoda, że firma Adobe poszła na takie uproszczenia, stojące w sprzeczności z nazwą produktu. *Portable* znaczy przenośny. Mam nadzieję, że wraz z upowszechnianiem się technologii ACROBAT powyższe mankamenty zostaną usunięte. Zresztą nawet anglojęzyczni użytkownicy mogą mieć kłopoty – przeciętny zjadacz Windowsów nie musi wiedzieć, że „ffi" np. w słowie *difficult* to jeden znak a nie trzy. A jak wprowadzić ligaturę „ffi" z klawiatury?



Rys. 3. Polskie znaki diakrytyczne zniknęły w spisie treści, ale w dokumencie (prawe okno) wszystko jest w porządku

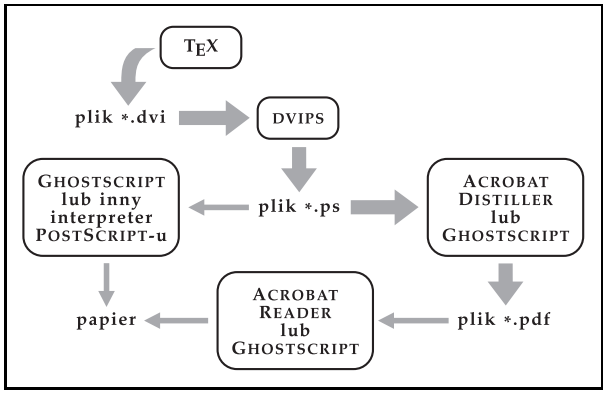
Tworzenie dokumentów w formacie PDF w systemie TEX – uwagi wstępne

Proces konwersji dokumentów TEX-owych do formatu PDF przedstawiono schematycznie na rysunku 4. Wszystkie elementy systemu mają status *public domain*, za wyjątkiem oczywiście DISTILLER-a. W chwili obecnej DISTILLER jest programem generującym pliki PDF o najlepszej jakości, ale można mieć nadzieję, że już niedługo równie dobre rezultaty będzie można osiągnąć za pomocą GHOSTSCRIPT-a[3].

Pierwszy etap zamiany pliku TEX-owego na PDF to zwykła kompilacja z wykorzystaniem odpowiednich makrodefinicji. Jeżeli używamy LATEX-a, to można użyć do tego pakietu `hyperref` Sebastiana Rahtza, lub pakietu `hyper` Michaela Mehlicha. Oba są dostępne w węzłach CTAN-u (patrz literatura). Jeżeli *nie* używamy LATEX-a, to prawdopodobnie poradzimy sobie sami z przygotowaniem odpowienich makr, za punkt wyjścia mogą służyć makrodefinicje pakietu `tp-pdf`, oparte

2: Niektóre wtyczki są obsługiwane także przez ACROBAT READER-a.

3: Nadzieja ta wynika z historii rozwoju GHOSTSCRIPT-a. Jeszcze kilka lat temu był to program oględnie mówiąc średni.



Rys. 4. Schemat generowania plików PDF przygotowanych w systemie TEX

o pomysły S. Rahtza, opisane na stronach 7–9 tego opracowania.

Pakiety `hyper` i `hyperref` różnią się implementacją. Pakiet `hyper` jest zgodny ze specyfikacją projektu HyperTEX, który – mówiąc w dużym skrócie – proponuje pewien standard dotyczący komend `\special` związanych z hipertekstem. Takie komendy `\special` mają się rozpoczynać od `html:`. Wykonanie komendy `\special` przez TEX-a polega na rozwinięciu jej zawartości i przesłaniu do pliku `dvi`. Zawartość ta winna być zrozumiała dla urządzenia drukującego. Stare wersje DVIPS nie rozumiały komend `\special` zaczynających się od `html:`, trzeba było używać mutacji nazwanej DVIHPS. Obecnie nie ma takiej potrzeby – DVIPS uruchomiony z opcją `-z` działa tak, jak DVIHPS.

Pakiet `hyperref` także jest zgodny ze specyfikacją projektu HyperTEX, ale po uaktywnieniu opcji `nativepdf`, pakiet wstawiać może instrukcje `\special`, zawierające „od razu" operatory `pdfmark`. Takie `special`-e są oczywiście zrozumiałe dla „normalnego" DVIPS, nie ma potrzeby używania DVIHPS, czy uruchamiać DVIPS-a z opcją `-z`. Piszę o tym wszystkim z uwagi na bałagan i zaszłości jakie można znaleźć w dokumentacjach.

Po utworzeniu pliku `dvi` uruchamiamy DVIPS i tworzymy plik POSTSCRIPT-owy. Jeżeli chcemy aby dokument PDF-owy zawierał hipertekstowe zakładki, (*bookmarks*) to: jeżeli używamy pakietu `hyperref`, należy przetworzyć otrzymany plik POSTSCRIPT-owy za pomocą PERL-a (odpowiedni skrypt jest dołączony do pakietu); pakiet `hyper` nie ma możliwości dodania zakładek; w pakiecie `tp-pdf` zakładki obsługiwane są bez potrzeby korzystania z programu zewnętrznego. Tak przygotowany plik POSTSCRIPT-owy jest gotowy do „destylacji".

Destylacja. Najlepsze jakościowo pliki PDF tworzy obecnie DISTILLER. Jest to jednak program komercyjny, i to dość drogi. Firma Adobe oferuje go w zestawie programów o nazwie Acrobat Pro za ok. 300 dolarów.

Obsługa DISTILLER-a jest bardzo prosta, ustawienia domyślne parametrów programu w zasadzie wystarczają. Tym niemniej warto przestudiować opis parametrów [2] – odpowiednia zmiana ustawień domyślnych może poprawić jakość plików i/lub zmniejszyć ich rozmiar.

Na przykład DISTILLER domyślnie dołącza do dokumentu cały font, jeżeli więcej niż 10% znaków tego fontu jest wykorzystanych w dokumencie. Umieszczenie w pliku konfiguracyjnym o nazwie `example.ps` następujących informacji

```
<< /SubsetFonts true /MaxSubsetPct 99 >>
setdistillerparams
```

spowoduje, że dopiero po wykorzystaniu 99% znaków cały font będzie dołączany do dokumentu, a w pozostałych przypadkach jedynie jego podzbiór znajdzie się w pliku PDF. Powyższa opcja przestanie być istotna z chwilą gdy w powszechnym użytku znajdzie się DVIPS posiadający możliwość dołączania podzbioru znaków fontu.

Począwszy od wersji 4.00 GHOSTSCRIPT także potrafi „destylować" pliki POSTSCRIPT-owe. W tym celu podczas kompilacji musi zostać dołączone urządzenie `pdfwrite` (co jest standardem dla wszystkich platform 32 i 64 bitowych).

Najważniejszym ograniczeniem obecnej wersji `pdfwrite` jest to, że za wyjątkiem 13 „podstawowych" fontów POSTSCRIPT-owych (tj. Courier, Times, Helvetica oraz Symbol) wszystkie inne są konwertowane na reprezentację bitmapową. To samo dzieje się z fontami, które wprawdzie nazywają się Courier, Times, Helvetica czy Symbol ale używają niestandardowego wektora kodowania (przy czym za standard przyjmuje się zapewne *ADOBE standard encoding*). W obecnej wersji GHOSTSCRIPT-a nie zaimplementowano żadnej z opcji DISTILLER-a dotyczących *image resampling* oraz kompresji.

W celu konwersji pliku *plik*`.ps` do pliku *plik*`.pdf` można uruchomić GHOSTSCRIPT-a (`gs` oznacza nazwę programu) w następujący sposób:

```
gs -q -dNOPAUSE -sDEVICE=pdfwrite
 -sOutputFile=plik.pdf plik.ps -c quit
```

Oczywiście można to wszystko umieścić w odpowiednim skrypcie/batchu; w dystrybucji GHOSTSCRIPT-a (system DOS) znajduje się gotowy skrypt, `ps2pdf.bat`. Użycie `ps2pdf` jest bardzo proste:

```
ps2pdf plik.ps plik.pdf
```

Problem fontów. Fonty Computern Modern (czy ich polskie odpowiedniki fonty `pl`), ciągle najczęściej używane fonty w składzie systemem TEX, oryginalnie są fontami bitmapowymi (w formacie `pk`), wygenerowanymi na określoną rozdzielczość. Urządzenie drukujące zamienia format `pk` na format specyficzny dla konkretnej drukarki. DVIPS zamienia fonty `pk` na POSTSCRIPT-owe fonty typu trzeciego (bitmapowe). Powstały w ten sposób plik POSTSCRIPT-owy jest zależny od urządzenia drukującego (rozdzielczości).

Kiedy produktem końcowym jest wydruk na papierze, nie jest to wielkim problemem, bo zawsze możemy przygotować odpowiednie do używanej drukarki fonty (na podstawie ich postaci METAFONT-owej). Ale gdy przygotowujemy dokument elektroniczny to musimy zakładać, że będzie on oglądany i drukowany na urządzeniach o najrozmaitszych parametrach.

Aczkolwiek plik POSTSCRIPT-owy zawierający fonty bitmapowe *zostanie* przekształcony do formatu PDF, to mając na uwadze to wszystko co powiedziano wyżej należy *unikać* ich stosowania w dokumentach w formacie PDF. Ponadto fonty bitmapowe są wyświetlane dużo wolniej na ekranie, oraz co chyba najważniejsze, ich czytelność jest dużo gorsza (są po prostu nieczytelne).

Fonty Computern Modern (komplet plus fonty AMS Euler) są także dostępne jako obwiedniowe fonty typu pierwszego w pakiecie BaKoMa (autor Basil K. Małyszew). Pakiet GHOSTSCRIPT od wersji 4.00 zawiera kilkadziesiąt bardzo dobrej jakości fontów (w tym odpowiedniki wszystkich 35 fontów, w które standardowo jest wyposażony interpreter POSTSCRIPT-a). Są to fonty *public domain*, dostępne w wielu archiwach, nie zawierają one jednak polskich znaków diakrytycznych.

I tu dochodzimy do nieciekawej konkluzji, że nie za bardzo jest czym tworzyć *polskie* dokumenty PDF-owe. Darmowych fontów obwiedniowych (czyli mówiąc inaczej fontów typu pierwszego *alias* POSTSCRIPT-owych) zawierających polskie diakrytyki jest tyle, co kot napłakał. Jeżeli ktoś kupił jakiś font, to też wcale nie musi oznaczać, że może go dołączyć do pliku PDF ponieważ wielu producentów fontów na to się nie zgadza lub żąda za to ekstra pieniędzy.

Niniejsze opracowanie udało się złożyć *wyłącznie* fontami obwiedniowymi, o statusie *public domain*. Tekst zasadniczy złożony jest fontem PLATAN, do składu formuł matematycznych najlepsze wydają się fonty EULER, które w formacie obwiedniowym znaleźć można w pakiecie BaKoMa; w tym pakiecie znajdują się także fonty `logo*` potrzebne do składania nazwy METAFONT. Zamiast fontu `pltt*` użyto fontu NimbusMonoL-Regular z dystrybucji GHOSTSCRIPT-a. KAPITALIKI są składane fontem wirtualnym, wygenerowanym za pomocą programu `afm2tfm` (z opcją `-V`).

Operatory pdfmark

Jak już wspomniano, elementy hipertekstowe są realizowane za pomocą operatorów pdfmark. Plik POSTSCRIPT-owy „wzbogacony" o takie operatory, jest następnie zamieniany, np. DISTILLER-em na plik w formacie PDF. W tym punkcie omówimy dokładniej operatory pdfmark, za pomocą których realizowane są podstawowe konstrukcje hipertekstowe jak odsyłacz, zakładka i notka.

Odsyłacze hipertekstowe. Działanie odsyłacza hipertekstowego, w dominujących w chwili obecnej systemach „myszoidalnych" wygląda tak: naciśnięcie klawisza myszy, wtedy gdy wskaźnik myszy znajduje się w wyróżnionym miejscu dokumentu powoduje wyświetlenie innej części dokumentu (lub jakiejś części innego dokumentu). Wszyscy to znamy, bo używamy Netscape'a! W przypadku dokumentu zapisanego w HTML-u, polecenie `<a NAME = `*label*` </a>` określa miejsce, *do którego* nastąpi przeskok, zaś `<a href=`*label*`>`, miejsce, *z którego* odwołanie nastąpi. W przypadku PDF-a mechanizm jest identyczny, tylko szczegóły inne: miejsce, do którego nastąpi odwołanie definiowane jest za pomocą operatora `DEST`, którego składnia wygląda następująco[4]:

`[/Dest /`*label*` /View [`*spec*`] /DEST pdfmark`

Etykieta */label* jest odtąd identyfikatorem pewnego miejsca w dokumencie PDF-owym. Mówiąc precyzyjniej wskazuje na stronę[5], *na której* ją

4: Operator `DEST` może być zdefiniowany także w inny sposób. Przedstawiamy, zapis najwygodniejszy z punktu widzenia TEX-a. Pełną specyfikację operatorów pdfmark można znaleźć w [3]. (Uwaga ta dotyczy także innych omawianych w tekście operatorów pdfmark)

5: Bo dokument w formacie PDF, w przeciwieństwie do HTML-a, jest podzielony na strony.

zdefiniowano (za pomocą odpowiedniego operatora DEST). Analogia z LaTeX-ową komendą \label{*etykieta*} jest tutaj zupełnie na miejscu.

Ponadto pole /View [*spec*] określa jaka część tej strony i w jaki sposób ma być wyświetlona. Wartością *spec* może być m.in:

- Fit – wyświetlenie na ekranie całej strony,
- FitR *x1 y1 x2 y2* – wyświetlenie na ekranie prostokąta, zdefiniowanego za pomocą współrzędnych *x1*, *y1* oraz *x2* i *y2*,
- XYZ *left top zoom* – wyświetlenie fragmentu strony, gdzie *left* określa przesunięcie od lewego marginesu a *top* od góry strony. Wartość *zoom* określa powiększenie, przy czym 1 oznacza 100% (normalną wielkość).

Składnia operatora pdfmark, definiującego odsyłacz wygląda następująco:

```
[ /Rect [llx lly urx ury]
  /Border [bx by c [d] ]
  /Color [r g b]
  /Dest /label
  /Subtype /Link /ANN pdfmark
```

Pola /Rect [*llx lly urx ury*] oraz /Subtype[6] są wymagalne, pozostałe są opcjonalne. Wartością pola /Rect jest wektor czterech liczb określających lewy-dolny (*llx*, *lly*) i prawy-górny (*urx*, *ury*) wierzchołek prostokąta (w jednostkach używanych w języku PostScript). Kliknięcie myszą na ten prostokąt powoduje wyświetlenie części dokumentu związanej z etykietą */label*.

Wspomniany prostokąt jest normalnie obwiedziony ramką (widoczną tylko przy wyświetlaniu dokumentu – nie jest ona drukowana), której własności określa czteroelementowy wektor liczb, będący wartością pola /Border. Trzeci element tego wektora (oznaczony jako *c*) określa grubość ramki. Jeżeli c = 0 ramka będzie niewidoczna. Pozostałe trzy elementy są na tyle rzadko używane, że zamieszczenie ich opisu tutaj jest zbędne.

Kolor ramki jest zdefiniowany, za pomocą trzyelementowego wektora będącego wartością pola /Color. Elementy wektora są liczbami z przedziału $[0, 1]$. Każda taka trójka definiuje jakiś kolor, zgodnie z modelem RGB, np. /Color [1 0 0] określa, że ramka będzie czerwona.

6: Jedyną poprawną wartością tego pola jest /Link. Być może w przyszłości pojawią się inne możliwości.

Zakładki hipertekstowe. Zakładki hipertekstowe (*bookmarks*) wyświetlane w lewym oknie Acrobat Reader-a znakomicie ułatwiają przeglądanie dokumentu elektronicznego. Są one definiowane za pomocą operatora pdfmark, którego składnia wygląda następująco:

```
[ /Count num /Dest /label
  /Title (string) /OUT pdfmark
```

Klucz /Count może się pojawić *tylko wówczas* gdy zakładka ma „podzakładki". Wartość bezwzględna *num* określa wtedy liczbę zakładek bezpośrednio odnoszących się do zakładki nadrzędnej (tj. bez „podpodzakładek"). Znak liczby *num* określa sposób wyświetlania zakładek według następującej reguły: jeżeli *num* jest liczbą dodatnią zakładki podrzędne są wyświetlane, jeżeli jest ujemna są one ukryte.

Pole /Title określa tekst (treść) zakładki. Długość tekstu nie może przekraczać 65535 znaków. W praktyce napisy te powinny być dużo krótsze bowiem okno, w którym będą wyświetlane nie będzie zbyt szerokie.

Etykieta zdefiniowana w polu /Dest */label* wskazuje na jakieś miejsce w dokumencie, analogicznie jak ma to miejsce w przypadku odsyłaczy hipertekstowych. Operatory /OUT mogą się pojawić w dowolnym miejscu pliku PostScript-owego, jedynie ich kolejność jest istotna – bowiem według niej będą wyświetlane.

Notka. Operator /ANN umożliwia także tworzenie notek, oraz tzw. *custom annotations*, które nazwijmy notkami specjalnymi. Składnia operatora /ANN, w przypadku definiowania notki jest następująca:

```
[ /Contents (string)
  /Rect [llx lly urx ury]
  /Open b /Color [r g b]
  /Title (string) /ANN pdfmark
```

Powyższa konstrukcja powoduje pojawienie się prostokątnego okna, o współrzędnych określonych za pomocą liczb *llx*, *lly*, *urx*, *ury*, zawierającego tekst z pola /Contents. Tytuł okna określa pole /Title, kolor tła okna – pole /Color. Jeżeli wartością pola /Open jest true okno jest otwarte, i widoczna jest jego zawartość, jeżeli /Open ma wartość false, lub nie jest wyspecyfikowane, zamiast okna wyświetlana jest ikona. Pola /Title, /Color i /Open są opcjonalne.

Forma notki nie może być zdeterminowana na etapie tworzenia PDF-a. Tekst notki – w odróżnieniu od tekstu podstawowego – jest wyświetlany przez przeglądarkę fontem określanym przez jej użytkownika.

Notki specjalne mogą zawierać video, dźwięki i inne nietekstowe informacje. Nie będziemy, na razie, omawiać tego tematu.

Inne operatory pdfmark. Operator `/DOCINFO` umożliwia określenie podstawowych informacji o dokumencie, takich jak: autor, tytuł, temat, słowa kluczowe, i kilka innych (szczegóły [3] s. 18, lub pakiet `tp-pdf`). Informacje te są wyświetlane po wybraniu `general info` z menu `file` w programie ACROBAT READER. Z kolei `/DOCVIEW` umożliwia m.in. określenie jak ma być wyświetlany dokument w momencie jego otwarcia. Przykładowo, w pakiecie `tp-pdf` komenda `\UseOutlines` powoduje, za pomocą odpowiedniego operatora `/DOCVIEW`, wyświetlenie dokumentu z widocznymi zakładkami (standardem jest, że zakładki są na początku niewidoczne).

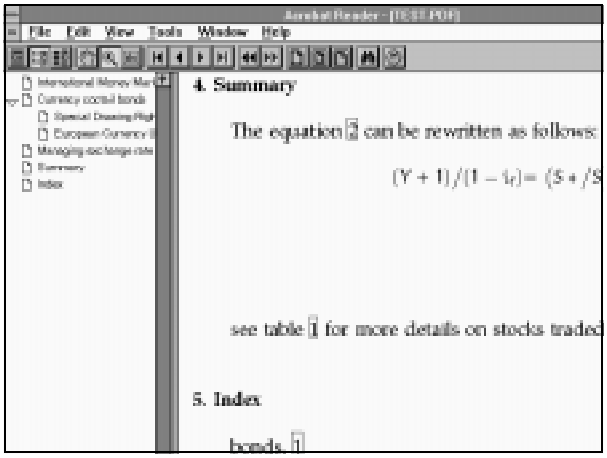
Pakiet tp-pdf

Pakiet `tp-pdf` jest rozwinięciem plain-owego pakietu makr `tp-crf` (por. [7]), służącego do oprogramowywania odsyłaczy. Pakiet ten włącza się do dokumentu w następujący sposób:

```
\input tp-pdf.tex
deklaracje pakietu
\PDFbegin
```

W odróżnieniu od LaTeX-a, który definiuje za użytkownika wszystko od początku do końca, pakiet `tp-pdf` wymaga zdefiniowania niektórych instrukcji, do których odwołują się podstawowe komendy pakietu. W szczególności każda instrukcja TEX-owa definiująca element dokumentu, do którego ma nastąpić odesłanie (punkt, podpunkt, tabela, rysunek, równanie, przypis), powinna zostać odpowiednio zdefiniowana czy zmodyfikowana.

Weźmy na przykład testowy dokument dołączony do pliku `tp-pdf`. Ma on bardzo prostą strukturę, gdyż jest tylko podzielony na punkty za pomocą instrukcji `\sec`. Rysunki 1 (str. 1) i 5 są ilustracją działania odsyłacza hipertekstowego. Ustawiając wskaźnik myszy na prostokącie obrysowanym wokół cyfry 4 w zdaniu „see also eq. 3 in section 4.” (rys. 1), na ekranie powinna zostać



Rys. 5. Przykładowy dokument (patrz rys. 1) po skorzystaniu z odsyłacza „section 4”.

wyświetlona części dokumentu przedstawiona na rysunku 5.

Fragment pliku TEX-owego odpowiadający temu co przedstawia rys. 5, wygląda następująco:

```
\sec{Summary}
\label{sec:summary}
The equation~\ref{eq:yield}...
```

Jak widać, notacja jest dokładnie taka sama jak w przypadku „normalnych” odsyłaczy co oznacza, że jeżeli dokument został prawidłowo przygotowany (nawet wtedy, kiedy jeszcze nikt nie słyszał o PDF-ie) to do zdefiniowania odsyłaczy hipertekstowych wystarczy prosta modyfikacja komend określających strukturę dokumentu. W naszym przykładzie instrukcja `\sec` jest zdefiniowana następująco:

```
\def\sec#1{\presecskip
 \global \advance\secC 1
 \def\CurrentCounter{\the\secC}
 \def\CurrentPDFLabel{sec.\the\secC}%
 \leftline{%
  \PDFdestination{\CurrentPDFLabel}%
  \bf\CurrentCounter.~#1}
 \writetorep{\CurrentPDFLabel}{#1}%
 \par \postsecskip }
```

W powyższym fragmencie kodu wykorzystane zostały dwie instrukcje pakietu `tp-pdf`, mianowicie `\PDFdestination` (związana z odsyłaczami) oraz `\writetorep` (związana z zakładkami). Ponadto zdefiniowane zostały makrodefinicje `\CurrentCounter` oraz `\CurrentPDFLabel` wykorzystywane później w instrukcji `\label`.

Zacznijmy od odsyłaczy. W tym przypadku zasadniczą rolę odgrywa określenie miejsca na stronie, na które wskazuje odsyłacz. Ponieważ TEX nie pozwala uzyskać odpowiedniej informacji (por. artykuł B. Jackowskiego „Ach, te kształty" w niniejszym numerze biuletynu), pozostaje wykorzystać do tego celu POSTSCRIPT.

Standardowym sposobem do zorientowania się „a gdzież to się znajdujemy na stronie?" jest użycie POSTSCRIPT-owego operatora `currentpoint` zwracającego współrzędne bieżącego punktu. Jeżeli zatem przed rozpoczęciem składania, np. tytułu rozdziału ustalimy bieżące współrzędne na stronie, to jest to bardzo dobry punkt, od którego może być wyświetlana strona. Nie wchodząc w szczegóły komend POSTSCRIPT-owych, przyjmijmy, że makro `\PDFdestination{`*label*`}` definiuje współrzędne bieżącego punktu, a następnie wysyła odpowiednią komendę `\special`, nadając polu `/Dest` wartość *label*.

Zauważmy, że w związku z tą techniką `\PDFdestination` jest wykonywane *przed* rozpoczęciem składania elementu, na który potem wskazuje, a co za tym idzie, TEX nie przeczytał jeszcze komendy `\label`, nie może zatem wykorzystać jej argumentu jako wartości pola `/Dest`. Rozwiązanie tego jest następujące: każda instrukcja definiująca element dokumentu, do którego może nastąpić odesłanie, generuje etykietę automatycznie, np. w przypadku punktu jest to `sec.`*kolejny-numer-punktu* (definicja `\CurrentPDFLabel`). Ta etykieta jest wstawiana do pliku `dvi` i jest wartością pola `/Dest`.

W taki sposób należy zmodyfikować wszystkie instrukcje określające te elementy dokumentu, do których chcemy się odwoływać. Natomiast opisane poniżej instrukcje `\label` i `\ref` są częścią, pakietu `tp-pdf` i z reguły nie ma potrzeby ich modyfikowania.

Jeżeli po wykonaniu `\sec` TEX napotka makro `\label{`*label*`}`, to do pliku są wysyłane, zamiast trzech, cztery elementy:

```
\newlabel{label}{{odnośnik}{str}{TEX-label}}
```

gdzie *label*, to etykieta zdefiniowana przez użytkownika, *TEX-label* to etykieta zapamiętana jako wartość makra `\CurrentPDFLabel` i wreszcie *odnośnik* to wartość makra `\CurrentCounter`, określającego odnośnik. Etykiety *label* i *TEX-label* muszą wskazywać na to samo miejsce dokumentu i ten sam odnośnik, ale to już załatwiają makra pakietu `tp-pdf`.

Do określenia miejsca, z którego ma nastąpić odesłanie służy komenda `\ref`. Makro to powinno wstawiać odsyłacz i przesyłać do pliku `dvi` odpowiedni operator `/ANN`. W tym celu modyfikujemy, komendę `\ref` z pakietu `tp-crf`.

```
\def\ref#1{%
 \@ifundefined{#1}{\@markmissingcr
 \@crwrn{undefined cr-> \string#1}}%
 {\def\CurrPDFAnchor{\expandafter
 \expandafter\expandafter
 \@CDR \csname #1\endcsname }%
 \PDFanchorstart{\CurrPDFAnchor}%
 \xref{#1}\PDFanchorend }}
```

Komenda `\PDFanchorstart{\CurrPDFAnchor}` definiuje punkty *llx*, *lly*. Następnie TEX składa zwykły odnośnik (za pomocą `\xref`, która jest identyczna z komendą `\ref` z pakietu `tp-crf`) – z reguły jakiś numer, a potem komenda `\PDFanchorend` definiuje punkty *urx*, *ury* i wysyła odpowiedni operator `/ANN`. W ten prosty sposób, wyznaczony zostaje aktywny prostokąt, naokoło tradycyjnego odsyłacza.

W wyniku wykonania komendy `\CurrPDFAnchor` zawierającej instrukcje `\@CDR \csname #1\endcsname`, poprzedzone trzema komendami `\expandafter` (por. wiersze 4–6), otrzymujemy po prostu *TEX-label*. Potrójne użycie operatora `\expandafter` gwarantuje właściwą kolejność rozwijania poszczególnych komend. Definicja `\@CDR` jest następująca:

```
\def\@CDR#1#2#3{#3}
```

Oczywiście makra `\@car` i `\@cdr`, trzeba tak zmodyfikować aby wybierały pierwszy i drugi z *trzech* argumentów (por. [7], s. 36)

Przygotowanie zakładek hipertekstowych jest równie proste, jednakże kilka rzeczy wartych jest wyjaśnień. Makro definiujące element dokumentu, z którym ma być związana zakładka powinno zawierać komendę `\writetorep{`*TEX-label*`}{`*string*`}` (p. wyżej). W wyniku działania instrukcji `\writetorep`, powstaje plik, z rozszerzeniem `.rep`. Do pliku `rep` wysyłane są wiersze, z których każdy zawiera dwie informacje: `{`*TEX-label*`}{`*string*`}`, np. fragment pliku `.rep` wygenerowany po kompilacji tego artykułu wygląda następująco:

```
{sec.6}{Operatory pdfmark}
{ssec.6.1}{Odsyłacze hipertekstowe}
```

Postać etykiety *TEX-label* może być w dużym stopniu dowolna. Jedyne na co należy zwrócić uwagę, to liczba kropek w niej się znajdujących, ponieważ na podstawie ich liczby ustalany jest „poziom" zakładki, tj. jedna kropka – zakładka, dwie – podzakładka, itd.

Należy zadbać o to, by napis *string*, który zostanie wysłany do pliku `dvi` (po rozwinięciu przez TEX-a) nie zawierał niczego poza zwykłym tekstem, ponieważ jest on przeznaczony dla READER-a. Na przykład umieszczenie w napisie instrukcji `\TeX` spowoduje, że na ekranie READER-a zamiast spodziewanych trzech liter pojawi się rozwinięcie:

```
T\kern -.1667em\lower.5ex\hbox{E}...
```

Problemy tego rodzaju pomaga rozwiązać komenda `\Xsanitize`, którą użytkownik powinien odpowiednio dostosowywać do swoich potrzeb. Standardowo zamienia ona jedynie tyldę (~) na spację, a komendę `\TeX` – na napis `TeX`:

```
\def\Xsanitize{\def\TeX{TeX}\def~{ }}
```

Jeżeli np. w dokumencie pojawia się słowo `Fr\"{u}n\c{c}ois`, to trzeba dodać do `\Xsanitize` odpowiednie definicje komend `\"` i `\c`. Pamiętajmy tylko, że *string* jest wyświetlany fontem ekranowym, a nie TEX-owym i w związku z powyższym ma inny wektor kodowania.

Plik `rep` musi być przetworzony do postaci zgodnej ze specyfikacją operatora `/OUT` (patrz str. 7). Zadanie to wykonuje makrodefinicja `\AddBookmarks`. Wykonywana jest ona na początku kompilacji, przed rozpoczęciem pisania do pliku `rep`, o ile komendy `\PDFbegin` nie poprzedzono deklaracją `\NoBookmarks`. Takie rozwiązanie oznacza konieczność dwukrotnej kompilacji dokumentu, ale i tak w ogólnym przypadku powtórna kompilacja jest nieunikniona. Makrodefinicja `\AddBookmarks` przy okazji zamienia polskie litery na ich łacińskie odpowiedniki (w przeciwnym razie uzyskuje się rezultaty przedstawione na rys. 3). Jeżeli kiedyś ten zabieg przestanie być konieczny, lub jeżeli z jakiś innych względów zakładki mają zawierać polskie diakrytyki, należy wśród deklaracji poprzedzających `\PDFbegin` umieścić `\NoPLSanitize`.

Deklaracja `\UseOutlines` powoduje, że dokument jest otwierany z widocznymi zakładkami.

Po kompilacji, za pomocą DVIPS tworzymy plik POSTSCRIPT-owy, gotowy do destylacji.

Pakiet `tp-pdf` potrafi także dodać hipertekstowe odnośniki do numerów stron w skorowidzach generowanych we współpracy z programem PLINDEX i plikiem makr `plidxmac` (por. [5]). Wystarczy dołączyć do dokumentu plik `plidxmac.tex` *przed* dołączeniem pliku `tp-pdf.tex`, i wykonać komendę `makeindex`, tj.:

```
\input plidxmac \makeindex
\input tp-pdf.tex
```

Sposób, w jaki działa PLINDEX powoduje, że odnośniki wskazują na stronę a nie na miejsce na stronie.

Podsumowanie

Przykłady dokumentów w formacie PDF, wykonanych tak jak opisano to powyżej, znaleźć można na stronie `http://www.GUST.org.pl/PDF` i w archiwum GUST: `ftp://ftp.GUST.org.pl/TeX/GUST/bulletin/`. Są tam artykuły z archiwalnych numerów biuletynu, statut, deklaracja itd. Pakiet `tp-pdf` jest dostępny w archiwum GUST w katalogu `TeX/GUST/contrib/TeX-PDF`.

Podziękowanie

Dziękuję Bogusławowi Jackowskiemu za współpracę w przygotowaniu artykułu.

Literatura

[1] Adobe Systems Incorporated. *Portable Document Format Reference Manual*, version 1.2, December, 1996. Dokument dostępny w `www.adobe.com`.

[2] Adobe Systems Incorporated. *Acrobat Distiller Parameters*, Adobe Developer Support. Technical Note 5151, 1 September 1995. Dokument dostępny w `www.adobe.com`.

[3] Adobe Systems Incorporated. *pdfmark Reference Manual*, Adobe Developer Support. Technical Note 5150, 7 July 1995. Dokument dostępny w `www.adobe.com`.

[4] Adobe Systems Incorporated. *PostScript Language Reference Manual*, Addison-Wesley, Reading 1990.

[5] Bogusław Lichoński. `Plidxmac` makra do tworzenia skorowidzów w systemie plain-TEX. Dokumentacja dla użytkowników i programistów. Dostępna na serwerze GUST (`ftp://ftp.GUST.org.pl`).

[6] Michael Mehlich. Opis pakietu `hyper.dtx`. Dostępny na serwerach GUST i CTAN.

[7] Tomasz Przechlewski. *Odsyłacze*. GUST, nr 5, 1995. Dostępny także w `www.GUST.org.pl`.

[8] Sebastian Rahtz i Yannis Haralambous. Opis pakietu `hyperref`, wersja 4.06. Dostępny na serwerach GUST i CTAN.

[9] Tomas Rokicki. Dokumentacja do programu DVIPS, wersja 5.58. Dostępna na serwerach GUST i CTAN.

[10] Tanmoy Bhattacharya. Opis pakietu `hyper-tex.sty`. Dostępny w `ftp.xxx.lanl.gov`.

◇ Tomasz Przechlewski
`ekotp@univ.gda.pl`