LaTeX2HTML

Piotr Bolek

Streszczenie

Pakiet LaTeX2HTML oferuje dosyć specyficzny, choć nie pozbawiony pewnych zalet sposób tworzenia dokumentów hipertekstowych umożliwiając konwersję dokumentów LaTeX-owych na format HTML. Omówione zostaną instalacja i konfiguracja pakietu, pokazane będą przykłady konwersji różnych klas dokumentów LaTeX-owych (dokumenty „czysto tekstowe”, dokumenty z ilustracjami, dokumenty ze wzorami matematycznymi). Omówione zostaną także możliwości zmiany domyślnych ustawień pakietu pozwalające wpływać na wygląd dokumentów HTML-owych generowanych przy pomocy pakietu.

Wprowadzenie

Co to jest LaTeX2HTML. LaTeX2HTML jest narzędziem służącym do konwersji dokumentów LaTeX-owych do formatu HTML. Pakiet ten pozwala wykorzystać możliwości jakie daje język makrodefinicji TeX-owych przy tworzeniu dokumentów hipertekstowych. Umożliwia także łatwe tworzenie stron WWW wszystkim użytkownikom LaTeX-a.

Dokument źródłowy w LaTeX-u jest przekształcany na zbiór połączonych dokumentów HTML--owych. LaTeX2HTML automatycznie łączy poszczególne dokumenty między sobą generując odpowiednie odnośniki hipertekstowe. Wszystkie odnośniki wewnątrz dokumentu, takie jak odwołania do sekcji, ilustracji, tabel, wzorów, numerów stron pozycji bibliograficznych, przypisy itp. są przekształcane na odnośniki hipertekstowe.

Elementy typograficzne definiowane za pomocą poleceń LaTeX-a, które mają swoje odpowiedniki w HTML-u (np. akapity, wyróżnienia tekstu, wyliczenia) są zamieniane na odpowiadające im znaczniki HTML-owe. Elementy dokumentu LaTeX-owego, które nie mają bezpośredniej reprezentacji w języku HTML (np. wzory matematyczne, rysunki, tabele) mogą być automatycznie przekształcone na mapy bitowe i umieszczone w odpowiednich miejscach w dokumencie HTML.

Kiedy może się przydać LaTeX2HTML. Tworzenie dokumentów hipertekstowych przy pomocy systemu pierwotnie przeznaczonego do składu tekstu wydaje się rozwiązaniem nieco dyskusyjnym. Bez wątpienia jest to metoda idealna dla kogoś kto zna i używa LaTeX-a i musi szybko przygotować jakiś dokument nie tylko do druku, ale także do wykorzystania w serwisie WWW. Teoretycznie LaTeX2HTML umożliwia tworzenie stron WWW bez potrzeby znajomości języka HTML. Efektywne wykorzystanie pakietu może jednak wymagać od użytkownika znajomości specyfiki dokumentów hipertekstowych, minimalnej chociaż znajomości języka HTML oraz, w przypadku potrzeby zmodyfikowania domyślnego sposobu działania pakietu, także języka Perl, w którym jest napisany pakiet LaTeX2HTML. Znajomość Perla będzie także konieczna w przypadku potrzeby dodania obsługi niestandardowych makr używanych w przetwarzanym dokumencie.

LaTeX2HTML może być także niezastąpionym narzędziem dla osób tworzących dokumentację do programów, która ma być dostępna w formie drukowanej w wersji hipertekstowej w formacie HTML. Użycie pakietu LaTeX2HTML może być w takim przypadku sposobem na osiągnięcie celu stosunkowo niewielkim kosztem. „Ideologicznie” lepszą metodą byłoby wtedy tworzenie dokumentacji w SGML-u, zastosowanie pakietu LaTeX2HTML może być jednak prostsze i w wielu przypadkach wystarczające.

LaTeX2HTML może być także najlepszym rozwiązaniem problemu publikacji dokumentów wcześniej złożonych w LaTeX-u, które trzeba opublikować w WWW. Oczywiście w takim przypadku innym rozwiązaniem może być także umieszczenie dokumentu w formacie PDF. Często jednak dobrze jest udostępnić dokumenty w różnych postaciach tak, żeby odbiorca sam mógł wybrać najbardziej odpowiadającą mu formę prezentacji. Ocena potrzeb i wybór odpowiednich technik prezentacji i narzędzi do przygotowania dokumentów zależy od każdego przedsięwzięcia związanego z publikacją dokumentów w wielu postaciach.

Możliwości pakietu. Pakiet LaTeX2HTML udostępnia następujące możliwości:

- Podział dokumentu źródłowego na oddzielne strony HTML-owe połączone odnośnikami hipertekstowymi. Stopień „fragmentacji” może być określony przez użytkownika.
- Realizacja paneli nawigacyjnych, które mogą być w prosty sposób modyfikowane przez użytkownika. Panele takie są automatycznie umieszczane na każdej generowanej stronie.

- Przetwarzanie dowolnych wzorów matematycznych, automatyczna numeracja wzorów, tabele, rysunki oraz możliwość używania dowolnych środowisk.
- Możliwość dowolnego skalowania i obracania rysunków.
- Automatyczne rozwijanie definicji nowych poleceń, nawet jeśli są umieszczone w oddzielnym pliku – daje to możliwość tworzenia makr HTML-owych w LaTeX-u.
- Obsługa przypisów, spisów tabel i rysunków, bibliografii i indeksów.
- Automatyczne przekształcanie odnośników i odwołań do literatury w odnośniki hipertekstowe. Rozszerzenie możliwości definiowania odwołań o obsługę odwołań między dokumentami.
- Konstrukcje warunkowe umożliwiające definiowanie oddzielnych fragmentów dokumentu umieszczanych w wersji hipertekstowej albo papierowej.
- Możliwość umieszczania wstawek bezpośrednio w języku HTML (np. w celu definiowania formularzy).
- Różne mechanizmy przetwarzania wzorów matematycznych: przekształcanie całych wzorów na grafikę (GIF lub PNG), etykiety HTML-owe (HTML 3.0), przekształcanie na grafikę części wzorów, których nie da się zdefiniować tekstowo.

Instalacja

Pakiet LaTeX2HTML jest przeznaczony dla systemów unixopodobnych. Został on także przeniesiony do systemu Windows NT, ale ta wersja nie jest jeszcze dołączona do „oficjalnego” pakietu dystrybucyjnego. Tutaj zajmiemy się wyłącznie wersją unixową (v. 98.1).

Instalacja pakietu LaTeX2HTML jest niezwykle prosta, zwłaszcza jeśli pracujemy w systemie, którym nie administrujemy. W takim przypadku zgłaszamy administratorowi systemu, że potrzebny jest nam program LaTeX2HTML mówiąc mu ewentualnie, że można go ściągnąć z najbliższego serwera zawierającego kopię archiwum CTAN … i czekamy na załatwienie sprawy.

Jeśli chcemy używać pakietu na naszym komputerze osobistym, gdzie mamy zainstalowanego np. Linuxa sprawa jest nieco bardziej skomplikowana. Przede wszystkim musimy wcześniej zainstalować (lub upewnić się, że mamy już zainstalowane) pewne programy, z których korzysta LaTeX2HTML:

- Perl w wersji 5.002 lub wyższej (dostępną w systemie wersję Perla można sprawdzić wywołując: `perl -v`);
- oczywiście instalacja TeX-a z LaTeX-em w wersji LaTeX2ε i programem dvips w wersji 5.58 lub wyższej;
- interpreter Postsciptu Ghostscript w wersji 4.02 lub wyższej;
- pakiet programów do konwersji grafiki `netpbm` wersja z 1 marca 1994 (`pnmfile -version`).

Procedura instalacyjna. W celu zainstalowania pakietu należy wykonać następujące czynności:

1. Rozpakować pakiet dystrybucyjny. W systemie Linux (i wszędzie tam gdzie jest dostępny program `tar` w wersji GNU) należy wywołać:

   ```
   tar zxvf latex2html-⟨xx⟩.tar.gz
   ```

   W innych systemach należy rozpakować pakiet poleceniem:

   ```
   gzip -cd latex2html-⟨xx⟩.gz \
     | tar xvf -
   ```

   Po rozpakowaniu, w bieżącym katalogu pojawi się podkatalog `latex2html`, w którym znajdują się wszystkie pliki należące do pakietu. Pakiet może być rozpakowany w dowolnym katalogu. W czasie instalacji w plikach konfiguracyjnych i w samym skrypcie `latex2html` zostanie zapisana ścieżka do tego katalogu.
2. Ustalić położenie programu Perl w systemie (`which perl`), i jeśli różni się ono od tego co jest podane w pierwszym wierszu plików `latex2html`, `texexpand`, `pstoimg`, `install-test` i `makemap`, to we wszystkich tych plikach należy odpowiednio wiersz ten zmienić.
3. Skopiować pliki, które będą używane przez LaTeX-a (czyli zawartość katalogu `texinputs`) w miejsce, które jest przeszukiwane przez LaTeX-a w czasie działania.
4. Uruchomić skrypt `install-test`. Skrypt ten wykonuje właściwą procedurę instalacyjną: zapamiętuje miejsce instalacji pakietu, sprawdza czy są dostępne wszystkie programy potrzebne do działania pakietu LaTeX2HTML i zapisuje w pliku konfiguracyjnym `local.pm` ścieżki do wszystkich zewnętrznych programów wykorzystywanych przez pakiet. Skrypt ten zadaje także pytanie o format, w jakim mają być generowane fragmenty tekstu reprezentowane

jako grafika. Do wyboru są dwa formaty: GIF albo PNG. Obecnie w większości przypadków należy wybrać format GIF, ponieważ jeszcze nie wszystkie przeglądarki WWW obsługują format PNG (Netscape już tak).

5. Teraz można skopiować plik `latex2html` do katalogu, który należy do ścieżki poszukiwań programów wykonywalnych (np. `/usr/local/bin`).
6. Można teraz także zmodyfikować plik `latex2html.config` dostosowując go do swoich potrzeb. W przypadku instalowania pakietu dla wielu użytkowników, w tym pliku powinny być ustawione globalne parametry domyślne, wspólne dla wszystkich użytkowników. Każdy z użytkowników może później zastąpić je ustawieniami w prywatnych plikach konfiguracyjnych `latex2html.init` w katalogach roboczych, albo w katalogach domowych.

Po zainstalowaniu można już zacząć używanie pakietu. Jeśli mamy dokument LaTeX-owy, który chcemy przetworzyć na HTML wystarczy wywołać polecenie:

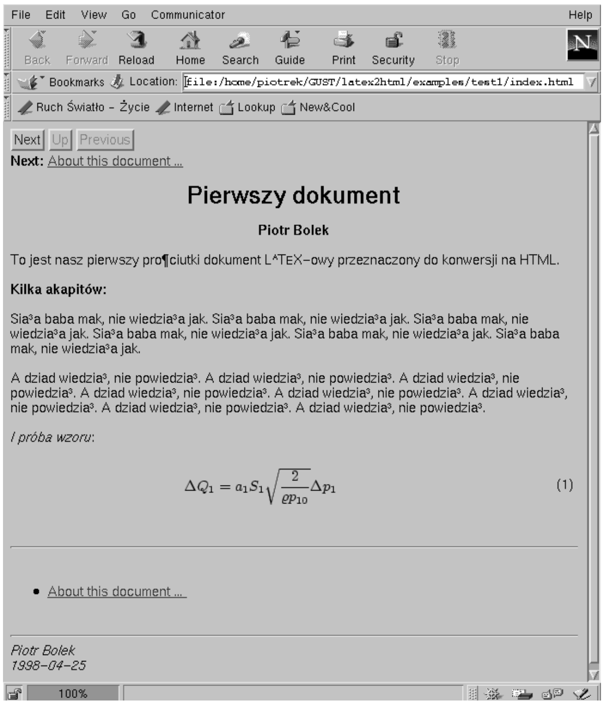
`latex2html` ⟨*nazwa-pliku*⟩

gdzie ⟨*nazwa-pliku*⟩ jest nazwą pliku źródłowego (można pominąć w niej rozszerzenie `.tex`). Po skończeniu działania programu w bieżącym katalogu powstanie podkatalog ⟨*nazwa-pliku*⟩ (bez rozszerzenia), zawierający wersję HTML-ową naszego dokumentu. Sprawa więc wygląda na niezbyt skomplikowaną. Są jednak jeszcze drobne problemy.

Pierwszy dokument

Spróbujmy więc dokonać konwersji pierwszego prostego dokumentu. Dokument źródłowy jest następujący:

```
\documentclass{article}
\usepackage{polski}

\author{Piotr Bolek}
\title{Pierwszy dokument}
\begin{document}
\maketitle
\thispagestyle{empty}
To jest nasz pierwszy prościutki dokument
\LaTeX-owy przeznaczony do
konwersji na HTML.

\textbf{Kilka akapitów:}

Siała baba mak, nie wiedziała jak.
Siała baba mak, nie wiedziała jak.
...
A dziad wiedział, nie powiedział.
A dziad wiedział, nie powiedział.
...
\emph{I próba wzoru}:
\begin{equation}
  \Delta Q_1 = a_1 S_1
  \sqrt{\frac2{\varrho p_{10}}}
  \Delta p_1
\end{equation}
\end{document}
```



Rys. 1.

Efekt działania programu nie jest niestety idealny. Co prawda program poradził sobie ze wzorem zamieniając go na grafikę, ale są problemy z językiem polskim – zamiast polskich literek pojawiły się dziwne krzaczki. Nie jest to na szczęście wielki problem, bo jak się łatwo domyślić przyczyną jest zadeklarowanie w pliku wynikowym niewłaściwego zestawu znaków. Domyślnym zestawem znaków programu jest bowiem `iso-8859-1` czyli `latin1`. Na szczęście istnieje możliwość łatwej zmiany domyślnego zestawu znaków. Można to zrobić opcją wywołania, albo (lepiej) zmianą domyślnego zestawu znaków w pliku konfiguracyjnym.

Drugim problemem, już nie tak krytycznym, są angielskie napisy pojawiające się w panelu nawigacyjnym i na przyciskach. To także można stosunkowo łatwo zmienić. Wymaga to umieszczenia odpowiednich deklaracji w pliku konfiguracyjnym i przygotowania własnych, spolszczonych wersji przycisków.

LaTeX2HTML a sprawa polska

Definitywne rozwiązanie problemu polskiego w pakiecie LaTeX2HTML jest sprawą prostą. Twórcy pakietu przewidzieli, że będą go używać także ludzie posługujący się nie tylko językiem angielskim i zaimplementowali odpowiednie mechanizmy.

Przede wszystkim dostępna jest opcja o nazwie `-html_version`, której jednym z parametrów jest zestaw znaków używany w dokumencie. Możliwe jest konwertowanie dokumentów zakodowanych z wykorzystaniem standardów ISO-Latin-[123456] oraz częściowa obsługa standardu Unicode. Dodanie do opcji `-html_version` wartości `latin2`, spowoduje że w nagłówku dokumentu HTML-owego pojawi się deklaracja:

```
<META HTTP-EQUIV="Content-Type"
 CONTENT="text/html;
 charset=iso-8859-2">
```

Możliwe jest także spolszczenie napisów generowanych, czyli uzyskanie: *Spis treści* zamiast *Contents*, *Bibliografia* zamiast *References* itd. W LaTeX-u odpowiednie definicje zmieniające te nazwy na polskie znajdują się w pakiecie polski włączanym do naszego dokumentu w nagłówku. W konfiguracji programu LaTeX2HTML dostępna jest zmienna $TITLES_LANGUAGE określająca w jakim języku mają być generowane napisy. Obecnie w standardowej wersji programu LaTeX2HTML dostępne jest wsparcie dla języka angielskiego, francuskiego i niemieckiego. Dodanie nowego języka jest jednak sprawą stosunkowo prostą.

W programie LaTeX2HTML można także definiować pakiety perlowe obsługujące używane w dokumentach pakiety LaTeX-owe. Obsługa wielu pakietów standardowych jest już zaimplementowana – pakiet polski nie jest jeszcze niestety traktowany jako standardowy. Przykład generowania dokumentów z uwzględnieniem języka polskiego będzie pokazany dalej.

Konfiguracja programu

Program LaTeX2HTML ma bardzo duże możliwości konfiguracyjne. Wiele parametrów konfiguracyjnych można podawać jako opcje wywołania konwertera. Wszystkie te opcje mają swoje odpowiedniki w parametrach konfiguracyjnych, które można umieszczać w globalnym pliku konfiguracyjnym (`latex2html.config`), w plikach konfiguracyjnych użytkowników (`.latex2html-init`) albo w plikach podawanych w opcjach `-init_file`. Wszystkie pliki konfiguracyjne są przy wywołaniu programu interpretowane jako zwyczajny kod perlowy. Zawierają one najczęściej po prostu przypisania wartości pewnym zmiennym, które wpływają na działanie konwertera. Pliki konfiguracyjne są włączane do programu głównego perlowym poleceniem `require` na początku działania konwertera. Typowe pozycje w pliku konfiguracyjnym wyglądają następująco:

```
$PARAMETR_PIERWSZY = 1;
$PARAMETR_DRUGI = "tekst";
1;
```

Zmienne w Perlu zawsze zaczynają się znakiem dolara, a kolejne instrukcje rozdzielane są znakiem średnika. Wartość parametru może być liczbą, albo ciągiem znaków. W ostatnim wierszu pliku konfiguracyjnego musi znajdować się liczba 1 (jeden) i średnik.

Ponieważ pliki konfiguracyjne są zwyczajnymi fragmentami kodu w Perlu, to można w nich także definiować procedury. Jest to wykorzystywane np. przy definiowaniu nowej wersji językowej oraz przy definiowaniu własnych nagłówków i stopek każdej strony.

Skonfigurowanie pakietu do obsługi języka polskiego

Aby LaTeX2HTML poprawnie obsługiwał dokumenty w języku polskim trzeba nieco pogrzebać w konfiguracji. Trzeba będzie zdefiniować kilka zmiennych i co najmniej jedną procedurę.

Oto plik konfiguracyjny modyfikujący podstawowe zachowanie konwertera tak aby generował on dokumenty w języku polskim:

```
$CHARSET="iso_8859_2";
$HTML_VERSION="3.2,latin2";
$TITLES_LANGUAGE='polish';

sub polish_titles {
 $toc_title = "Spis treści";
```

```
    $lof_title = "Spis rysunków";
    $lot_title = "Spis tabel";
    $idx_title = "Skorowidz";
    $ref_title = "Bibliografia";
    $bib_title = "Bibliografia";
    $abs_title = "Streszczenie";
    $app_title = "Dodatek";
    $pre_title = "Przedmowa";
    $fig_name = "Rysunek";
    $tab_name = "Tabela";
    $part_name = "Część";
    $prf_name = "Dowód";
    $child_name = "Podrozdziały";
    $info_title="Informacje o dokumencie";
    @Month = ('', 'styczeń', 'luty',
    'marzec', 'kwiecień', 'maj',
    'czerwiec', 'lipiec', 'sierpień',
    'wrzesień', 'październik',
    'listopad', 'grudzień');
   }

   sub top_navigation_panel {
   ...
   }
```

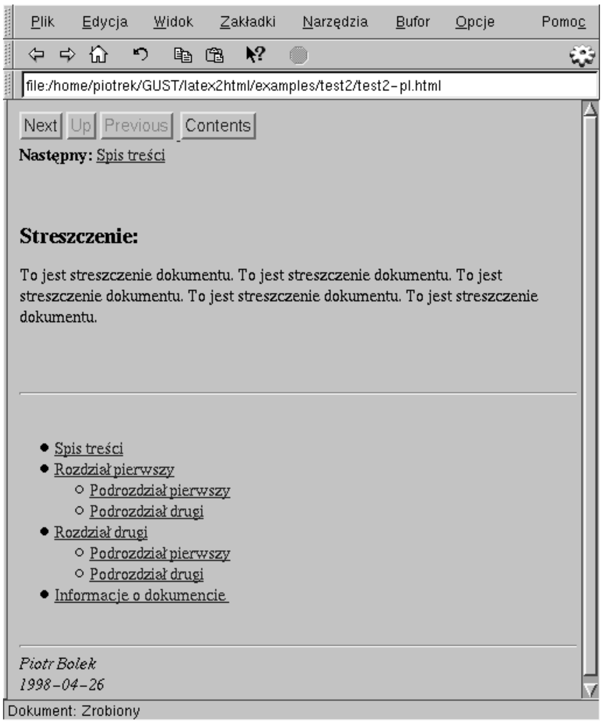
W tym przykładzie na początku definiujemy kodowanie używanego zestawu znaków. Dla poprawnego ustawienia używanego w dokumencie kodowania wystarczy użycie jednej z podanych konstrukcji. W trzecim wierszu podajemy, że językiem w jakim będą pisane tytuły generowane przez konwerter jest polski. Ponieważ domyślnie w programie LaTeX2HTML nie są zdefiniowane polskie tytuły, to musimy zrobić to sami. W tym celu definiujemy procedurę o nazwie `polish_titles`. Procedura ta zostanie automatycznie wywołana przez konwerter na początku działania, dzięki czemu zmienne określające generowane przez konwerter teksty będą miały jako wartości odpowiednie słowa polskie. Dodatkowo na końcu procedury jest definiowana tablica z polskimi nazwami miesięcy.

Na samym końcu naszego pliku konfiguracyjnego jest umieszczona procedura służąca do definiowania wyglądu panelu sterującego. Nie podaję tutaj jej definicji, ponieważ jest to ona skopiowana z oryginalnego pliku konfiguracyjnego `latex2html.config`, ze zmienionymi napisami: *Next* na *Następny* itd.

Jeśli zapiszemy tę konfigurację w pliku `pl1.cnf` i wywołamy konwerter:

```
latex2html -init_file pl1.cnf test2
```

to możemy otrzymać stronę jak na rys. 2. Widać tam, że zamiast *Next* pojawiło się słowo *Następny*, zamiast *Abstract* – *Streszczenie* i zamiast *About this document...* *Informacje o dokumencie*. Niestety w języku angielskim są jeszcze napisy na ikonach nawigacyjnych. Ale na to rada jest tylko jedna – trzeba zastąpić domyślne ikony konwertera własnymi, co niebawem uczynimy.



Rys. 2.

Można oczywiście zapisać naszą konfigurację do prywatnego pliku `~/.latex2html.init`. Wtedy wszystkie nasze dokumenty będą generowane w wersji „polskiej”.

Definiowanie własnego układu stron

Każda strona generowana przez LaTeX2HTML może składać się z:

- górnego panelu nawigacyjnego,
- treści,
- dolnego panelu nawigacyjnego,
- podpisu.

Wszystkie elementy strony poza treścią są opcjonalne. Wygląd każdego z tych elementów można także dowolnie modyfikować. W panelu można umieszczać dowolne teksty i grafikę oraz elementy nawigacyjne w postaci ikon lub odnośników tekstowych.

Konfiguracja układu strony jest dosyć skomplikowana. Poniżej wymienione są parametry, które służą do definiowania paneli nawigacyjnych i podpisu:

- Struktury paneli są definiowane jako procedury perlowe `top_navigation_panel` i `bot_navigation_panel`. W procedurach tych mogą być realizowane dowolne działania. Każda z tych procedur powinna zwrócić ciąg znaków (można używać zmiennych, w których są zapamiętane nazwy plików graficznych używanych w nagłówku i stopce – patrz niżej). Napisy zwrócone przez te procedury będą, po rozwinięciu wszystkich zmiennych, wstawione jako kod HTML-owy odpowiednio w nagłówkach i stopkach stron.
- Wszystkie ikony używane do nawigacji są zapamiętane w tablicy asocjacyjnej `%icons`. Tablica ta jest domyślnie zdefiniowana w pliku `latex2html.config`, ale można ją przedefiniować w prywatnych plikach konfiguracyjnych. Indeksami tej tablicy są nazwy zmiennych, w których będą przechowywane nazwy ikon. Wartościami są nazwy plików, w których są zapisane ikony. W czasie działania konwerter definiuje zmienne o nazwach takich jak indeksy tablicy. Zmiennych tych można używać do konstrukcji struktury paneli w procedurach `top_navigation_panel` i `bot_navigation_panel`. Domyślne nazwy plików wykorzystywanych jako ikony tworzone są wg. schematu: ⟨*funkcja*⟩`_motif.`⟨*format*⟩ i ⟨*funkcja*⟩`_motif_gr.`⟨*format*⟩, przy czym ⟨*funkcja*⟩ to np. `up`, `next`, `contents` itd., a ⟨*format*⟩, to `gif` lub `png` w zależności od domyślnego formatu graficznego jaki wybraliśmy w czasie konfiguracji pakietu. Każda ikona występuje w dwóch wersjach: aktywnej i nieaktywnej – z przyrostkiem `_gr` umieszczonym na końcu nazwy właściwej pliku. W oryginalnych ikonach nieaktywne wersje mają „wyblakłe” szare napisy i stąd konwencja ich oznaczania.
- Rozmiary ikon zdefiniowanych w tablicy `%icons` powinny być podane w tablicy `%iconsizes`.

Użycie własnych ikon w panelu nawigacyjnym. Najprostsza zmiana w wyglądzie strony jaką można zrobić, to zastąpienie ikon domyślnych własnymi. Aby to zrobić trzeba przygotować własne wersje plików graficznych, nazywając je tak jak oryginalne ikony. Aby nasze nowe ikony zostały użyte musimy jeszcze podać konwerterowi miejsce, gdzie znajdują się nasze ikony oraz rozmiary naszych wersji ikon. Wszystko to można zrobić w pliku konfiguracyjnym:

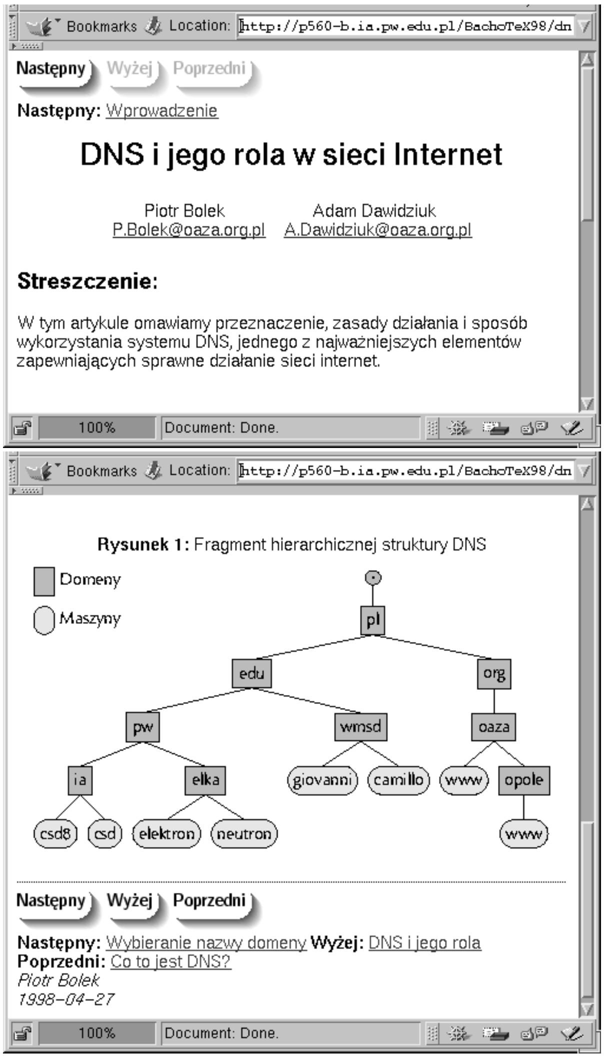
```
$BODYTEXT='bgcolor="white"';
$LATEX_COLOR="\\pagecolor[gray]{1}";
$ICONSERVER="../moje_ikony/";

%iconsizes =
(( 'up', 'WIDTH="53" HEIGHT="31"',
 'next','WIDTH="72" HEIGHT="32"',
 'previous','WIDTH="75" HEIGHT="31"',
 'contents','WIDTH="79" HEIGHT="33"',
 'index','WIDTH="77" HEIGHT="31"'
) ;

1;
```

W powyższym pliku konfiguracyjnym użyta jest zmienna `$ICONSERVER`, która zawiera adres (URL), pod którym można znaleźć ikony. Ponieważ przygotowane zostały własne wersje ikon, o innych niż oryginalne rozmiarach, to konieczne było podanie rozmiarów podmienionych ikon (tablica `%iconsizes`). Domyślne nazwy plików graficznych wykorzystywanych jako ikony nie zostały zmienione, więc nie było potrzeby zmiany zawartości tablicy `%icons`.

Oprócz zmiany ikon w tym przykładzie dodaliśmy do elementu `<BODY>` dokumentu HTML-owego atrybut `BGCOLOR=WHITE`, więc wszystkie strony będą miały białe tło. Parametr `$LATEX_COLOR` służy do podania koloru „papieru” (tła strony) jaki będzie użyty w czasie przetwarzania rysunków i wzorów. Domyślnie kolor ten jest szary (taki jak kolor zwyczajnego tła przeglądarki WWW). Jeśli w generowanych przez konwerter rysunkach (np. ze wzorów matematycznych) używany jest antyaliasing, to kolor używany jako tło w docelowym dokumencie HTML-owym i kolor używany jako tło przy przetwarzaniu powinny być takie same.

Na rys. 3 pokazano fragmenty przykładowych stron dokumentu przetworzonego z wykorzystaniem obu zaprezentowanych wyżej plików konfiguracyjnych – tego, który przedefiniowuje tytuły elementów stałych i tego podmieniającego ikony. Konwerter został wywołany następująco:

```
latex2html -init_file moje_ikony.cnf \
  -init_file pl1.cnf dns
```



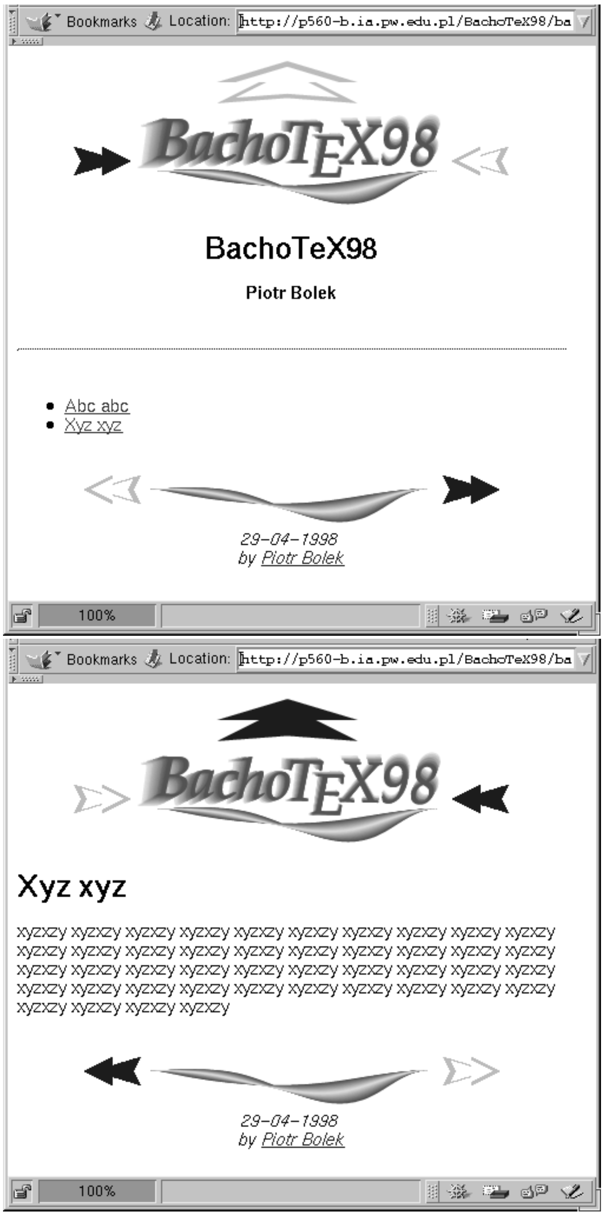
Rys. 3.

Przedefiniowanie struktury stron. Kolejnym przykładem będzie całkowite przedefiniowanie struktury stron generowanych przez LaTeX2HTML. Oto plik konfiguracyjny:

```
1. $INFO='';
2. $BODYTEXT="bgcolor=white";
3. $LATEX_COLOR="\\pagecolor[gray]{1}";
4. $ICONSERVER=
5.  "http://www/BachoTeX98/arrows";
6. $ALTERNATIVE_ICONS=
7. "/home/piotrek/GUST/latex2html/arrows";
8. $BOTTOM_NAVIGATION=1;
9.
10. $ADDRESS='<DIV ALIGN="center">'.
11.  `date +%d-%m-%Y`.
12.  '<br> by
13.  <A href="mailto:P.Bolek@ia.pw.edu.pl">
14.  Piotr Bolek</A></DIV>';
15.
16. %iconsizes =
17.  (( 'up', 'WIDTH="120" HEIGHT="36"',
18.  'next','WIDTH="48" HEIGHT="24"',
19.  'previous','WIDTH="48" HEIGHT="24"',
20. ) ;
21.
22. sub top_navigation_panel {
23. qq|
24. <!--Mój panel nawigacyjny-->
25. <table align="center">
26.  <tr><td><td align="center">$UP
27.  <tr><td>$NEXT<td align="center">
28.  <img src="/BachoTeX98/bachotex98.gif">
29.  <td>$PREVIOUS
30. </table>
31. <BR>|
32. }
33.
34. sub bot_navigation_panel {
35. "<BR>
36. <table border='0' align='center'>
37.  <tr>
38.  <td align='left'>$PREVIOUS
39.  <td align='center'>
40.  <img src='/BachoTeX98/img2.gif'>
41.  <img width=1 height=40
42.  src='/BachoTeX98/null.gif'>
43.  <td align='right'>$NEXT
44. </table>
45. ";
46. }
47.
48. 1;
```

Pierwszy wiersz sprawia, że nie jest generowana strona z informacją o dokumencie. Drugi i trzeci znamy już z poprzedniego przykładu. Wiersze 4, 5, 6 i 7 określają położenie naszych wersji ikon. Parametr `$ALTERNATIVE_ICONS` (wiersze 6 i 7) jest używany jeśli wywołamy konwerter z opcją `-local_icons` – ikony zostaną wtedy skopiowane do katalogu, w którym jest zapisywany dokument wynikowy. Po wywołaniu bez tej opcji ikony będą w czasie oglądania dokumentu w przeglądarce WWW ściągane spod adresu (URL) podanego jako wartość parametru `$ICONSERVER`.

Nadanie parametrowi `$BOTTOM_NAVIGATION` wartości 1 powoduje, że na wszystkich stronach



Rys. 4.

umieszczane są panele nawigacyjne na górze i dole strony.

Parametr $ADDRESS, służy do definiowania adresu (oczywiście może tam być cokolwiek – wcale nie musi być to adres), który pojawia się na dole każdej strony. Wartością tego parametru powinien być ciąg znaków. Domyślną wartością jest data (rok-miesiąc-dzień) i pełna nazwa użytkownika, który dokonał konwersji. W naszym pliku konfiguracyjnym jako adres podana jest data w innej konwencji (korzystamy z możliwości unixowego polecenia date) i nazwisko twórcy strony z adresem e-mail jako odnośnikiem. Perlowy operator „.” (kropka), został użyty do sklejenia wartości parametru w jedną całość.

Ponieważ w panelach używamy własnych wersji ikon nawigacyjnych, to musimy podać ich rozmiary (tablica %iconsizes).

Na końcu pliku konfiguracyjnego podane są dwie procedury, które definiują wygląd i działanie panelu. W procedurach tych używamy zmiennych $NEXT, $PREVIOUS i $UP, które zawierają kod HTML-owy, tworzący z naszych ikon przyciski nawigacyjne, do strony następnej, poprzedniej i wyższej w hierarchii dokumentu. LATEX2HTML wstawia różne ikony jako przyciski nawigacyjne w zależności od położenia strony w hierarchii dokumentu. Oba panele zawierają oprócz przycisków nawigacyjnych także dodatkowe pliki graficzne, stanowiące oprawę graficzną każdej strony dokumentu.

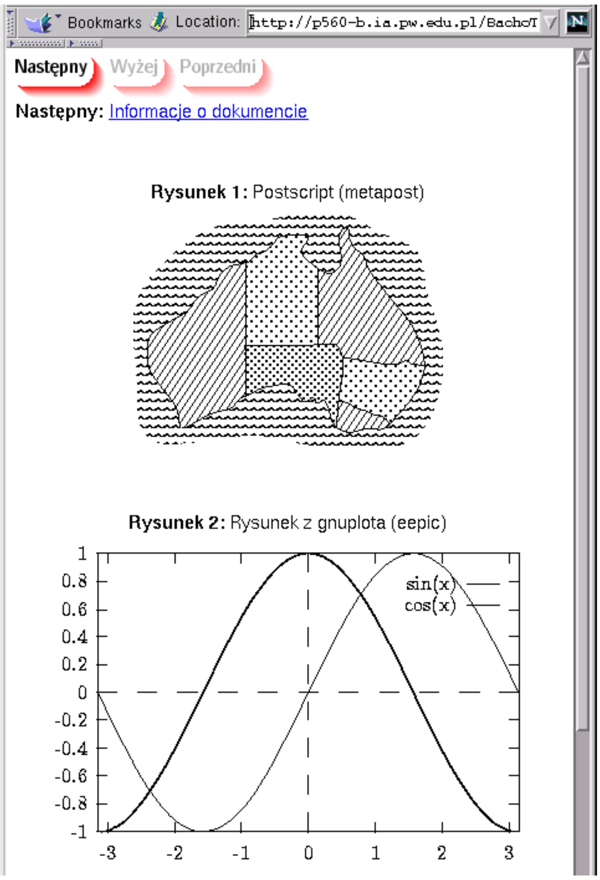
Dwie przykładowe strony wygenerowane z wykorzystaniem podanej konfiguracji pokazane są na rysunku 4.

LATEX2HTML, matematyka, tabele i rysunki

W pokazanych wcześniej przykładach widać już było, że program LATEX2HTML radzi sobie z wzorami i rysunkami wstawianymi do dokumentu LATEX-owego. W przypadku rysunków konwerter robi jedyną sensowną rzecz jaką można wymyślić. Do przekształcenia rysunków z formatów używanych w LATEX-u na mapę bitową (gif lub png) wykorzystywany jest LATEX, dvips, ghostscipt i filtry do konwersji grafiki z pakietu netpbm. Dzięki temu wszystkie rysunki we wszystkich akceptowanych przez LATEX-a i dvipsa formatach będą przekształcone na mapy bitowe w odpowiednich rozdzielczościach. Na rys. 5 pokazany jest przykład dokumentu z dwoma rysunkami: jeden z nich zrobiony był programem metapost, a drugi programem gnuplot i zapisany w formacie eepic.

Sposób przetwarzania tabel zależy od generowanej wersji HTML-a. Jeśli generujemy dokument do HTML-a w wersji 2.x, to tabele będą potraktowane dokładnie tak samo jak rysunki i zamienione na grafikę. W przypadku generowania dokumentu w formacie HTML 3.x tabele będą zrobione z wykorzystaniem poleceń HTML-wych.

Wzory są w zasadzie zawsze przetwarzane na grafikę. LATEX2HTML może co prawda generować dokumenty w HTML-u 3.0 (w opcji wywołania -html_version trzeba dodać wtedy parametr



Rys. 5.

math), w którym zdefiniowane są elementy do składania matematyki, ale wzory są poprawnie obsługiwane tylko przez przeglądarkę Arena, która jest programem raczej mało używanym.

Na sposób generowania grafiki reprezentującej wzory można wpływać używając opcji lub ustawień w plikach konfiguracyjnych. Normalnie każdy wzór jest w całości przekształcany na obrazek i wstawiany do dokumentu. Jeśli użyjemy wspomnianego wyżej rozszerzenia programu LaTeX2HTML o nazwie math, to wzory będą generowane zgodnie ze standardem HTML 3.0. Użycie tego rozszerzenia jednocześnie z opcją -no_math spowoduje, że konwerter będzie próbował części wzorów składać tekstem (np. zmienne nie w ułamkach, funkcje typu sin itp.), a z tego co się nie da potraktować w ten sposób zrobi grafikę. Taka hybrydowa metoda ma niestety tę wadę, że symbole pojawiające się w częściach wzorów składanych „tekstowo” mogą się znacznie różnić od symboli w częściach graficznych. Popatrzmy na rysunek 6.

Symbole, które są wyświetlane tekstowo różnią się od tych samych symboli w częściach graficznych.

$$\theta N_1 = \left(\frac{G_1(j\omega)N_1}{1+G_1(j\omega)N_1} \right) \tag{1}$$

$$\sin(x) = \sum_{x=0}^{\infty} \frac{\cos(xy)}{\sin(xz)} \tag{2}$$

$$\int_{i=0}^{20} a_x b_y c_z \frac{a_x b_y c_z}{x_a y_b z_c} \tag{3}$$

$$\begin{matrix} a & b & c \\ \alpha & \beta & \gamma \\ x & y & t \end{matrix} \tag{4}$$

Rys. 6.

Fontem, który służył do wyświetlania tekstu w przeglądarce była Helvetica, różnicę widać zwłaszcza w literach „a” oraz „y”. Problemy są także z pozycjonowaniem znaków (wzór 4). Wszystkie obrazki reprezentujące symbole są wyrównywane do dołu, co jak widać daje nie najlepszy efekt.

Technika dzielenia wzorów na części i wyświetlanie czego się da tekstowo, mogłaby dać lepsze efekty jeśli wykorzystane byłyby możliwości jakie dają „szablony stylów” (ang. *style sheets*). Najpopularniejsze przeglądarki obsługują już style i dają możliwość wyświetlania wybranych fragmentów tekstu wybranym krojem. Do wyświetlania składanych tekstowo fragmentów wzorów można byłoby wtedy użyć np. kroju Schoolbook, który ma podobne kształty liter do krojów cmr[1] (zwłaszcza w niskiej, ekranowej rozdzielczości). Konwerter LaTeX2HTML dla każdego dokumentu generuje co prawda styl CSS, ale obsługa szablonów nie jest jeszcze doskonała. Można co prawda nawet podać swój plik z szablonem, który będzie wykorzystywany zamiast szablonu generowanego przez konwerter, ale kod HTML-owy jest generowany w taki sposób, że odróżnienie kursywy matematycznej od tekstowej

1: Ewentualni oponenci niech wezmą pod uwagę, że jeśli w przeglądarce jako font podstawowy jest ustawiona Helvetica (co nie jest rzadkością), to jest to niewątpliwie prawda.

nie zawsze jest możliwe. Po zajrzeniu w kod konwertera widać, że twórcy są w trakcie pracy nad bardziej kompletną implementacją obsługi szablonów.

Rozszerzanie konwertera

Standardowa obsługa nowych poleceń. Pakiet LaTeX2HTML udostępnia mechanizmy standardowej obsługi poleceń. Przede wszystkim możemy zupełnie nie przejmować się tym, że używamy jakichś poleceń nieznanych dla konwertera. Domyślnym sposobem ich przetwarzania jest skopiowanie ich argumentów do pliku HTML-owego a samych poleceń do LaTeX-a w celu ewentualnego utworzenia obrazka. Może to prowadzić do błędów w czasie wywołania LaTeX-a.

Oprócz tego dostępnych jest kilka mechanizmów postępowania z nieznanymi poleceniami i ich argumentami:

- Ignorowanie (procedura `&ignore_commands`).
- Przekazanie do LaTeX-a w celu przerobienia na grafikę (`&process_commands_in_tex` albo `&process_commands_inline_in_tex`).
- Przekazanie do LaTeX-a w celu późniejszego wykorzystania przez inne polecenia (`&process_commands_nowrap_in_tex`).
- Przetworzenie jako środowiska (nie do końca jasne – `&process_commands_wrap_deferred`).

Wszystkich wymienionych powyżej procedur używa się w podobny sposób. Podaje im się listę poleceń (oraz ich argumentów opcjonalnych i wymaganych), po jednym poleceniu w wierszu. Argumenty wymagane oznacza się jako `{}` a opcjonalne – `[]`. Nazwę polecenia od argumentów i argumenty między sobą rozdzielamy znakiem krzyżyka (`#`). Użycie procedury `&ignore_commands` może wyglądać następująco:

```
&ignore_commands(<<_IGNORED_);
linebreak # []
mbox
_IGNORED_
```

W przykładzie tym ignorowane jest polecenie `\linebrak` wraz z opcjonalnym argumentem, a argument polecenia `\mbox` jest przekazywany na wyjście do pliku HTML.

Pozostałych procedur standardowej obsługi używa się analogicznie jak `&ignore_commands`.

Definiowanie obsługi poleceń LaTeX-owych. Podobnie jak w LaTeX-u definiuje się klasy i pakiety rozszerzające jego możliwości, albo zmieniające jego parametry, w programie LaTeX2HTML można definiować rozszerzenia realizujące konwersję poleceń definiowanych w pakietach LaTeX-owych. Rozszerzenia te to oczywiście moduły perlowe włączane automatycznie po napotkaniu przez program LaTeX2HTML pakietu LaTeX-owego w preambule dokumentu. W pakiecie dystrybucyjnym LaTeX2HTML znajduje się katalog `styles`, gdzie znajdują się perlowe odpowiedniki niektórych klas i pakietów LaTeX-owych. Są tam m.in. moduły:

- `article.perl`, `report.perl`, `book.perl` – odpowiedniki głównych klas LaTeX-owych;
- `graphics.perl`, `epsbox.perl`, `epsfig.perl`, `floatfig.perl`, `floatflt.perl`, `color.perl`, `colordvi.perl`, `wrapfig.perl` – moduły realizujące przetwarzanie grafiki i kolorów w dokumentach LaTeX-owych;
- `supertabular.perl`, `longtable.perl` – moduły przekształcające różne rodzaje tabel w sposób analogiczny do tabel definiowanych z wykorzystaniem środowiska `tabular`;
- `amsart.perl`, `amsbook.perl`, `amsfonts.perl`, `amsmath.perl`, `amssymb.perl`, `amstex.perl` – moduły obsługujące pakiety AMS-TeX-owe.

Nic oczywiście nie stoi na przeszkodzie aby pisać własne moduły, dla pakietów i klas tworzonych na własny użytek. Używając tego mechanizmu można także łatwo dodać obsługę języka polskiego za pomocą modułu `polski.perl`, który będzie ładowany automatycznie zawsze gdy w LaTeX-u użyjemy pakietu `polski`.

W modułach obsługujących style możemy definiować procedury, które będą automatycznie konwertowały zdefiniowane w tych stylach polecenia LaTeX-owe, na odpowiedni kod HTML-owy. Konwencję tworzenia takich procedur pokażemy na przykładzie.

Załóżmy, że w naszym pakiecie zdefiniowaliśmy trzy makra:

1. `\nbhyp` – bezparametrowe makro rozwijające się w niełamliwy dywiz.
2. `\email` – jednoparametrowe makro, którego argumentem jest adres poczty elektronicznej – w dokumencie HTML-owym chcemy je zastąpić odnośnikiem pocztowym.
3. `\href` – dwuargumentowe makro, definiujące odnośnik hipertekstowy – pierwszy argument to adres odnośnika, a drugi tekst aktywny.

Makro to chcemy zamienić na odnośnik (`<A HREF="...">`).

W programie LaTeX2HTML jest zdefiniowany ogólny mechanizm obsługi nieznanych poleceń LaTeX-owych. Polecenia są obsługiwane przez procedury o nazwach: do_cmd_⟨*polecenie*⟩. Procedura taka zostanie automatycznie wywołana przez konwerter w chwili napotkania polecenia. Procedury obsługujące polecenia LaTeX-owe muszą być skonstruowane w specjalny sposób. Dostają one jako argument fragment tekstu zaczynający się w miejscu napotkania polecenia. Schemat procedury jest następujący:

```
sub do_cmd_<name> {
# Wczytanie danych wejściowych
 local($_) = @_;
# Deklaracja lokalnych zmiennych
 local($arg1, $arg2, $result);

# Przeczytanie argumentów polecenia
# i~wczytanie ich do zmiennych lokalnych
 s/$next_pair_pr_rx/$arg1 = $2;'' /eo;
 s/$next_pair_pr_rx/$arg2 = $2;'' /eo;
# ...
# Obróbka parametrów
# ...

# Zwrócenie wyniku
 $result. $_;
}
```

Czytanie parametrów realizowane jest w sposób dosyć zawiły. W zmiennej `$next_pair_pr_rx` przechowywane jest wyrażenie regularne, dzięki któremu pobiera się kolejne parametry. Taką instrukcję zamiany (`s/.../.../eo`) trzeba powtórzyć tyle razy ile argumentów ma nasze polecenie. W naszym przykładzie obsługiwaliśmy polecenie dwuargumentowe.

Popatrzmy teraz na procedury obsługujące trzy wyżej wymienione polecenia. Pierwsza jest bardzo prosta. Ponieważ polecenie to nie ma argumentów, więc nic nie czytamy. Dopisujemy tylko na wyjście znak „–":

```
sub do_cmd_nbhyp {
 local($_) = @_;
 "-". $_;
}
```

W drugiej procedurze musimy przeczytać parametr i użyć go do utworzenia konstrukcji `<A HREF="mailto:arg">arg</A>`. Poza przeczytaniem argumentu i skonstruowaniem pożądanej wynikowej konstrukcji HTML-owej nie ma tutaj także nic trudnego.

```
sub do_cmd_email {
 local($_) = @_;
 local($a, $result);

 s/$next_pair_pr_rx/$text = $2;'' /eo;
 qq|<A HREF="mailto:$at">$a</A>|. $_;
}
```

Trzecia procedura różni się od poprzedniej właściwie tylko tym, że pobieramy tutaj dwa argumenty. Poza tym wygląda tak samo.

```
sub do_cmd_href {
 local($_) = @_;
 local($url, $text, $result);

 s/$next_pair_pr_rx/$url = $2;'' /eo;
 s/$next_pair_pr_rx/$text = $2;'' /eo;
 $result = qq|<A href="$url">$text</A>|;
 $result. $_;
}
```

Rzeczywiste procedury mogą oczywiście zawierać bardzie wymyślne przetwarzanie przeczytanych argumentów, najczęściej jednak procedury służące do konwersji poleceń LaTeX-owych na HTML nie będą bardziej skomplikowane od powyższych.

Wszystkie procedury obsługujące polecenia z jakiegoś pakietu można umieścić w pliku o nazwie takiej samej jak nazwa pakietu i rozszerzeniu `.perl`. Jeśli umieścimy taki plik w bieżącym katalogu albo w podkatalogu `styles` katalogu zainstalowania pakietu LaTeX2HTML to będzie on automatycznie wczytywany przez konwerter, gdy napotka on w preambule dokumentu wywołanie danego pakietu.

Podsumowanie

Pakiet LaTeX2HTML oferuje dosyć ciekawe podejście do tworzenia dokumentów hipertekstowych w formacie HTML. Konwersja prostych dokumentów, bez potrzeby modyfikowania domyślnych ustawień pakietu jest bardzo łatwa. Efektywne wykorzystanie pakietu może jednak wymagać dobrej znajomości nie tylko LaTeX-a, ale także formatu HTML i języka Perl, w którym napisany jest konwerter i realizowana jest jego konfiguracja.

Fakt, że konwerter jest napisany i konfigurowany w języku Perl jest dużą zaletą, ponieważ daje możliwości łatwego modyfikowania i rozszerzania działania.

W niniejszym tekście pominięte zostało wiele możliwości, jakie daje pakiet LaTeX2HTML np. rozszerzenia hipertekstowe standardowych poleceń LaTeX-owych, możliwość przetwarzania warunkowego, wstawki w języku HTML, obsługa indeksów i bibliografii. Możliwości te mogą być bardzo przydatne przy konwersji dokumentów istniejących, a zwłaszcza przy tworzeniu nowych dokumentów od podstaw. Dokładne omówienie wszystkich cech pakietu musiałoby zająć dużo więcej miejsca i czasu. Może kiedy indziej…

◇ Piotr Bolek
P.Bolek@ia.pw.edu.pl

Od Redakcji: Omawiany pakiet dostępny jest na serwerach GUST i CTAN, a także na CD TeXLive3 w katalogu:

`support/latex2html/`

Erik Frambach (też członek GUST ;-) prezentuje obok inne rozwiązanie łącza TeX→HTML – program TeX4ht.

Conversion of anyTeX to HTML using TeX4ht*

Erik Frambach

Abstract

Gurari's TeX4ht program makes it possible to convert TeX files (plain TeX, LaTeX or whatever) to HTML. *This article is a case-study of the conversion of the 4allTeX manual (written in LaTeX) to* HTML.
Keywords: conversion, html, www, TeX4ht.

Introduction

I guess that by now almost everyone knows what HTML is. In a nutshell: it's a standardized format for publishing text on the World Wide Web. The structure of well-written HTML documents is very similar to well-written (La)TeX documents, so conversion to HTML shouldn't be very hard.

In TeX you would write, e.g., '`{\it italic text}`', in HTML you would write '`<it>italic text</it>`'.

In LaTeX '`\chapter{Chapter title}`' is how you start a new chapter, in HTML it should convert to '`<h1>Chapter title</h1>`'.[1]

Just like in TeX, in HTML line breaks generally have no meaning, so you can spread text over any number of lines. But in HTML the start of a new paragraph has to be indicated explicitly, whereas in TeX an empty line is sufficient. In HTML an empty line has no special meaning.

Of course there are more differences, small ones and big ones, and we will run into several of them in this article.

Principles

TeX4ht consists of a set of macros (`tex4ht.sty`) and a DVI postprocessor (`tex4ht.exe`). TeX4ht is able to cooperate with arbitrary TeX formats because it doesn't interpret any TeX code itself, like LaTeX2HTML does. Almost all the work is left to TeX. The trick is that TeX4ht adds `\specials` to

---

*: Artykuł, który otrzymaliśmy bezpośrednio od Autora, jednocześnie ukazuje się w biuletynie grupy holenderskiej MAPS. Dziękujemy (*Red.*).

1: Likewise conversion of HTML to LaTeX is also easy. HTML can be interpreted by TeX directly, e.g. using Carlisle's LaTeX-package 'typehtml'.

the DVI file. These \specials will guide TeX4ht's DVI postprocessor in generating HTML. You can think of it like this. Whenever TeX reads a \section command and writes the title of the section to the DVI file, a \special{<h2>} and a \special{</h2>} are written too. The TeX4ht DVI postprocessor will find these \specials and it will know what to do. In reality it's much more complex than that because the table of contents needs an entry, and this entry should on its turn link back to that section. A lot of bookkeeping has to be done.

A significant advantage of this method is that numbering, references and macros have already been processed by TeX. TeX4ht only needs TeX's output.

Necessary adaptations

In order to use TeX4ht a few adaptations to your TeX file are necessary. First of all the TeX file tex4ht.sty has to be loaded:

```
\documentclass{rapport1}
...
\input{tex4ht.sty}
\Preamble{htm,3,fonts}
\begin{document}
\EndPreamble
...
\end{document}
```

Note that the style file tex4ht.sty is not loaded using the \usepackage command. The reason is that it's not strictly a LaTeX style file but a generic set of macros.

The \Preamble statement determines that the file extension of the HTML output shall be 'htm' ('html' is also possible but plain MS-DOS users will have trouble with that). The option '3' specifies that I don't want one long HTML document but rather a sectioning in three levels: chapters, sections and subsections. So, I will get more smaller HTML files, which is more convenient for Internet publishing. TeX4ht takes care automatically of the necessary structure and referencing to and from all parts.

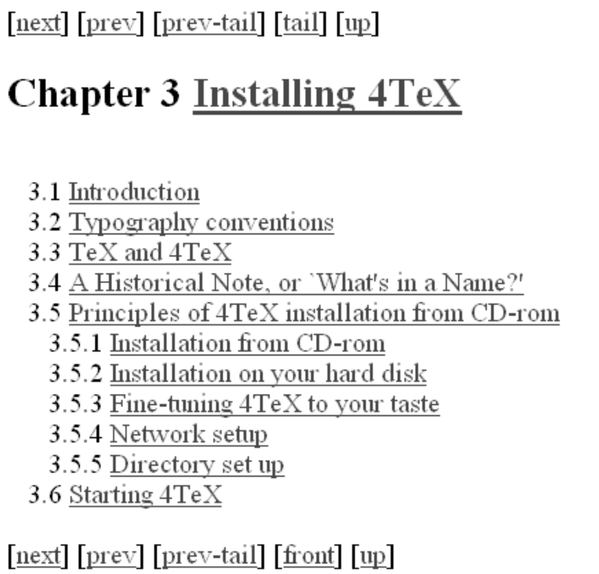
The option 'fonts' indicates that I want TeX4ht to respect (if possible) font switches such as italic, bold, and sansserif in the HTML version.

After this simple adaptation we can already make HTML because TeX4ht is preconfigured to work with LaTeX.

Running TeX4ht

After compiling the TeX file with these adaptations the resulting DVI file is no longer usable for "normal" previewing or printing. Therefore it's wise to copy all input files to a new directory before you start adapting the files.

The new DVI file can now be processed by the DVI postprocessor. It will generate HTML and some more files, the most important of which is the IVD file. This file contains all parts of the DVI file that TeX4ht can't convert to plain HTML. Some constructions are simply not possible in HTML (e.g. a picture environment), or insufficiently supported (e.g. mathematics and accented characters). The IVD file contains all such elements, one per page. TeX4ht generates its own bookkeeping files: XRE for cross-references and COT for the table of contents. These files don't require any user intervention.


[next] [prev] [prev-tail] [tail] [up]

Chapter 3 Installing 4TeX

3.1 Introduction
3.2 Typography conventions
3.3 TeX and 4TeX
3.4 A Historical Note, or 'What's in a Name?'
3.5 Principles of 4TeX installation from CD-rom
3.5.1 Installation from CD-rom
3.5.2 Installation on your hard disk
3.5.3 Fine-tuning 4TeX to your taste
3.5.4 Network setup
3.5.5 Directory set up
3.6 Starting 4TeX

[next] [prev] [prev-tail] [front] [up]

A partial table of contents

The IVD file can be converted to PostScript with DVIPS. The resulting PostScript file can then be converted to multiple PCX files by Ghostscript. These PCX files are actually full bitmapped pages, but at a low resolution, say, 110 dpi, because they will be displayed on computer screens.

The PCX files need to be "cropped", i.e., surrounding white space has to be removed. The program 'Display' can do that for you automatically. At the same time Display can convert PCX to GIF, the bitmap format that all WWW browsers support. The last step of the conversion is making the GIF files "transparent". The program 'GifTrans' will do just that.

Naturally TEX4ht did its bookkeeping well, so it knows where all the GIF files should be referenced.

All in all this procedure is not really simple but it can be automated quite easily. In 4TEX version 4 the whole procedure is a matter of pushing a few buttons. The HTML version of the 4allTEX manual on the 4allTEX cd-roms was produced directly by 4TEX without additional manipulation.

The whole procedure is rather time-consuming. The TEX4ht DVI postprocessor takes a lot of time to process a DVI file, but generating GIF files can also take a lot of time. But then again, generating an HTML version of a whole book is not a daily task for most of us.

Fine-tuning

Although the first results may already be acceptable there is a lot that can be improved by fine-tuning.

In the case of the 4TEX manual I made the following adaptations:

After the \Preamble I added:

```
\Configure{centerline}%
  {\HCode{<center>}}{\HCode{</center>}}

\let\Xincludegraphics=%
  \includegraphics
\def\includegraphics[#1]#2{%
  \hbox{\Picture+[pict]{}%
  \Xincludegraphics[#1]{#2}%
  \EndPicture}}
```

This code will make the \centerline command translate to its HTML equivalent <center> and </center>. The \includegraphics command is extended in such a way that TEX4ht knows now that how to handle it: generate a GIF picture. Note that the simple implementation I chose forces me to supply options in square brackets with each \includegraphics command. Of course a more flexible solution is possible but I regard that as syntactic sugar. A better solution is left to the reader as an exercise.

Another necessary adaptation is the removal of the \tableofcontents command. TEX4ht will automatically generates one, and two tables is really too much. Even if a book starts with a preface, the table of contents will always come first. That's why the preface now also gets an entry in the table of contents. It makes sense in this context.

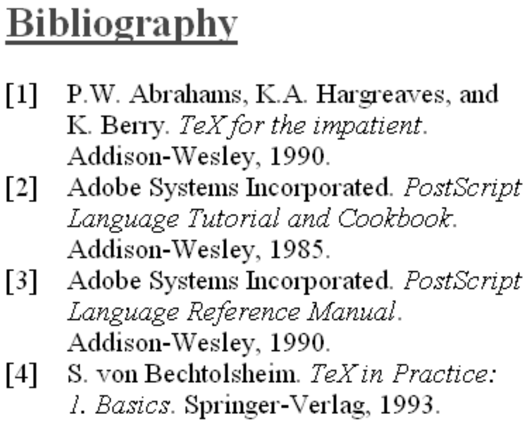
Bibliographies have to be handled differently too. Instead of

```
\bibliograpystyle{plain}
\bibliograpy{mybib}
```

you now have to specify:

```
\input{myfile.bbl}
```

Of course the bbl file has to be available from another previous TEX and BibTEX run.



Bibliography

[1] P.W. Abrahams, K.A. Hargreaves, and K. Berry. *TeX for the impatient*. Addison-Wesley, 1990.
[2] Adobe Systems Incorporated. *PostScript Language Tutorial and Cookbook*. Addison-Wesley, 1985.
[3] Adobe Systems Incorporated. *PostScript Language Reference Manual*. Addison-Wesley, 1990.
[4] S. von Bechtolsheim. *TeX in Practice: 1. Basics*. Springer-Verlag, 1993.

A part of the bibliography

Strictly speaking the low E from the TEX logo can be represented in HTML but exact positioning (kerning) is not possible, so the result will always be ugly. So it's better to write simple 'TeX'. The LATEX logo is even more difficult. TEX4ht is clever enough to recognize this problem. It doesn't even attempt to typeset the logos. For those who are not convinced: this is how the LATEX logo could be represented in HTML (on one line without spaces!):

```
L<SUP><FONT SIZE=-2>A</FONT></SUP>
T<SUB><FONT SIZE=+0>E</FONT></SUB>X
```

Note that it depends on your browser and the font it uses how good or bad this will look.

The LATEX logo in html: good or bad?

Because we are making an interactive document now it would be very useful to translate email and other Internet addresses ('URL's) into active links:

```
\def\email#1{%
  \HCode{<A HREF="mailto:#1">}%
  \texttt{#1}\HCode{</A>}}
\def\url#1{%
  \HCode{<A HREF="#1">}%
  \texttt{#1}\HCode{</A>}}
```

Of course this requires a little more knowledge of HTML, but it also shows how easy it is to add literal HTML code using the \HCode command. Another interesting example is a new definition of the key icons that we used in the printed version of the manual (e.g. $\boxed{\text{Esc}}$). I could easily have TeX4ht convert them to GIF pictures but that turned out to be a bad idea. TeX4ht regards every instance of a picture as unique so it will generate hundreds of pictures although there are no more than 30 unique pictures (by the way, this also applies to accented characters). Therefore the following solution is much better:

```
\def\KeyFont{\Large\sf}
\def\Key#1{%
  \HCode{<FONT COLOR="\#FF0000">}%
  {\KeyFont[#1]}%
  \HCode{</FONT>}}
```

In stead of a character in a black frame you now get a key in square brackets, all in red. On the screen this is very effective and of course it's much faster to download and render than GIF pictures.

Another peculiarity that needs fixing is the interpretation of LaTeX's `\\` command. TeX4ht simply ignores it completely. No problem. The HTML 'break' command can easily be added:

```
\let\CarriageReturn=\\
\def\\{\CarriageReturn\HCode{<BR>}}
```

Note that the command '`\\*`' now doesn't work anymore, but that option is irrelevant for the HTML version anyway. For '`\\[xx]`' the same applies though we sometimes abuse it to add some white space between 'paragraphs'. In those cases I replaced it with \par.

It's remarkably easy to typeset tables in HTML with TeX4ht.

If you simply use plain 'tabular' environments everywhere all goes well. TeX4ht will generate an HTML <TABLE> that any WWW browser can render quickly and reliably. Some other HTML convertors don't even try: they simply generate a large (ugly) picture.

The rest of the conversion came down to removing (if necessary) of spurious statements: `\clear(double)page`,\parskip,\parindent,\textheight and other typographic commands are meaningless in this context.

TeX4ht is very careful in typesetting mathematics. If possible it will generate HTML, if not it will include GIF pictures for (parts of) a formula.

For instance, to typeset $\sin x^2 \sum_1^\infty \delta x$ we would write

```
$\sin x^2 \sum_1^\infty \delta x$
```

and TeX4ht will generate the following HTML code:

```
sin x<SUP >2</SUP>
<FONT FACE="SYMBOL">a</FONT>
<SUB >1</SUB>
<SUP ><FONT FACE="SYMBOL">Y</FONT>
</SUP><FONT FACE="SYMBOL">d</FONT>x
```

Very decent. If we put the same formula in an equation environment then TeX4ht will generate a picture, in which of course the formula number will be included.

Conclusion

Using TeX4ht to convert (La)TeX to HTML can give very good results. Still there are a few weak points that could be improved:

- You have no control over placement and design of the table of contents.
- TeX4ht generates [up], [front], [tail], [prev-tail], [next] and [prev] links, both at the top and the bottom of each 'page'. On the bottom of the page they sometimes follow immediately after running text, without any white space.
- The HTML code could be improved. WWW browsers have no trouble with it, but if you want to tweak something it may be a little annoying.
- Pictures are not reused even if they are completely identical.
- The font definition files that come with TeX4ht are very usable but may still give too little information to make the conversion run perfectly. For the time being it seems wise to stick to the Computer Modern fonts or the 'mathptm' fonts.
- MakeIndex output is not (yet) supported.

But let's not forget that this program is still very young and in alpha testing phase. Compared to it's most prominent competitor LaTeX2HTML it has some clear advantages:

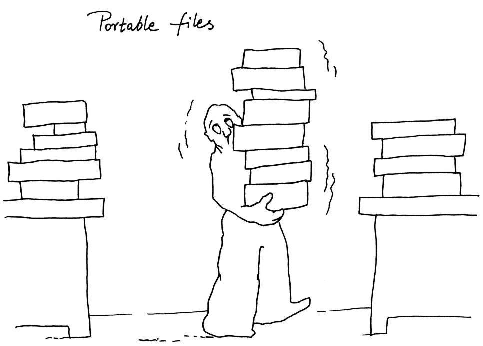
- The system is LaTeX independent: it can cooperate with any TeX system.
- The system is not Unix dependent. Nowadays LaTeX2HTML is not so Unix centered anymore either, but still…
- Many inherently difficult things (macros!) are left to TeX to resolve. Now who will argue with TeX?
- Numbering of chapters, sections, formulas, tables, figures etc. remains intact.
- Tables are decently converted to HTML.
- Input encoding is irrelevant, because TeX takes care of that.

The results of the conversion of the 4TeX manual are available on the 4allTeX cd-roms (see `\4texdoc\4texdoc.htm`) and (somewhat adapted) on the NTG WWW server: `http://www.ntg.nl/4allcd/4texdoc/4texdoc.htm`

Naturally the whole TeX4ht package (version alpha 3) is on the 4allTeX cd-roms (see disk2, `distrib\tex4ht`) and it's built into 4TeX.

From Gurari's WWW page `http://www.cis.ohio-state.edu/~gurari/` you can always download the newest version of TeX4ht. On Gurari's pages you can also find several nice examples of conversions. Take a good look at `http://www.cis.ohio-state.edu/~gurari/TeX4ht/mn162.html` in which Gurari demonstrates in detail how well known public domain TeX and LaTeX documents can be converted to HTML. It should provide lots of insight.

◇ Erik Frambach
E.H.M.Frambach@eco.rug.nl



Zastosowania

SYBISLAW – TeXtowa bibliograficzna baza danych

Piotr Chrząstowski

Wprowadzenie

System SYBISLAW (SYstem BIbliografii SLAWistycznej) jest systemem informacyjnym opracowywanym na zlecenie Instytutu Slawistyki Polskiej Akademii Nauk przez Piotra Chrząstowskiego i Marcina Engela. Celem jego jest stworzenie środowiska umożliwiającego wpisywanie i redakcję danych potrzebnych do wydania Słownika Slawistycznego. System został zrealizowany w oparciu o system składania tekstów LaTeX.

Niniejszy dokument stanowi opis procesu wprowadzania danych do systemu i ich segregacji. W projekcie założono wpisywanie danych przez bibliografów przeszkolonych w używaniu programu, w edytorze udostępnionym przez aplikację.

Model danych

Dane są przechowywane w postaci tekstowej. Nietypowe fonty i znaki diakrytyczne są reprezentowane w konwencjach LaTeX-owych.

Każdy rekord jest podzielony na pola. Ogólny format opisu pola jest następujący:

```
\nazwa-pola{treść pola}
```

Szczegółowy opis rekordu wraz z polami wewnątrz rekordu opisane są w rozdziale 5.

Sposób działania aplikacji

Aplikacja wczytywania danych służy wprowadzaniu, opracowywaniu i segregacji danych bibliograficznych.

Wprowadzanie tekstu odbywa się w specjalnych oknach edytora. Zadaniem osoby opracowującej tekst jest zaklasyfikowanie notki do odpowiedniego podrozdziału ze struktury zadanej drzewem (przez wybór odpowiedniego liścia drzewa; wyboru dokonuje się w sposób stosowany w programach typu Norton Commander). Opracowanie tekstu polega na wpisaniu do odpowiednich pól rekordu (autor, tytuł, ...) stosownych danych.

Fonty. Fonty trzeba podawać według konwencji LaTeX-owej.

Znaki diakrytyczne i cyrylica. Znaki diakrytyczne oraz litery nietypowych alfabetów należy podawać wprost w formacie LaTeX-a. Baza zadba o właściwe posortowanie danych.

Redagowanie tekstu. Redagowanie polega na wprowadzaniu tekstu (ewentualnie dokonywaniu poprawek) wraz z określeniem struktury pól i fontów. Prosty edytor będący częścią aplikacji umożliwia typowe funkcje edycyjne, jak wyszukiwanie, zamiana, przeglądanie tekstu.

Pliki w LaTeX-u

Rozdział ten omawia postać plików wykorzystywanych przez aplikację. Ogólnie pliki, z których korzysta aplikacja można podzielić na 3 rodzaje:

1. pliki z danymi;
2. pliki konfiguracyjne;
3. pliki dodatkowe.

Pliki z danymi. Baza danych znajduje się w plikach zapisanych w standardzie LaTeX-a. Wszystkie te pliki są umieszczane w podkatalogu `BAZA` głównego podkatalogu. W celu uniknięcia ich przypadkowego usunięcia są one opatrzone DOS-owym atrybutem *read-only*. Użytkownik nie powinien ingerować w zawartość tych plików, gdyż może to spowodować utratę spójności danych. Wszelkie zmiany redakcyjne i edytorskie powinny być wykonywane na *kopii* tychże plików utworzonych przez aplikację menedżera bazy danych, znajdujących się w podkatalogu `DODRUKU`. Pliki te stanowią także podstawę do przygotowania wersji gotowej do druku.

Każdy rozdział bazy pamiętany jest w osobnym pliku. Wszystkie one posiadają rozszerzenie `.tex`. Ich nazwy są tworzone w następujący sposób: pierwszą literą nazwy jest '*b*'; po niej następuje liczba liter określająca stopień zagłębienia rozdziału. Pliki znajdujące się zatem na poziomie 3 będą miały nazwę złożone z 4 liter (łącznie z wiodącą literą '*b*'). Litery występujące po pierwszej literze określają numer rozdziału w obrębie jednostki wyższej (a=1, b=2, ...). Zatem plik o nazwie `bacb.tex` oznacza drugi podpodrozdział trzeciego podrozdziału pierwszego rozdziału.

Plik główny o nazwie `rocznik.tex` składa się z określenia stylu i nazw rozdziałów głównych ułożonych w odpowiedniej kolejności.

Tu też znajdują się pliki zawierające słowniki skrótów (osobny dla cyrylicy).

Pliki konfiguracyjne. Pliki te znajdują się w podkatalogu `DODRUKU` i zawierają definicje i redefinicje standardowych makr LaTeX-owych służących do tworzenia indeksu, słownika oraz określających sposób drukowania poszczególnych typów pozycji bibliograficznych oraz ich pól.

Pakiet dostarcza pliki konfiguracyjne. Są to m.in.:

- Plik `bibdef.sty`. Jest to plik zawierający zaawansowane definicje LaTeX-owe, które powinny być redefiniowane przez użytkownika tylko w szczególnych przypadkach. Tutaj znajduje się między innymi środowisko do produkcji indeksu, odnośników oraz definicje makr odpowiadających typom pozycji bibliograficznych, które nie wpływają na ich wygląd w drukowanej pracy. Plik posiada dosowy atrybut *read-only*.
- Plik `format.sty` jest plikiem przeznaczonym dla użytkownika. Poprzez prostą modyfikację zawartych w nim definicji sposobu składania poszczególnych pól może on wpływać na końcową postać rocznika.

Pliki dodatkowe. Pliki te znajdują się również w katalogu `DODRUKU`. Można w nich umieścić dodatkowe informacje, które w końcowej wersji rocznika pojawią się przed i po właściwej jego części. Pliki te to:

- `wstep.tex` – początkowo zawiera jedynie polecenie utworzenia spisu treści, ale można dopisać do niego wszelkie elementy rocznika, które mają się pojawić przed właściwą bibliografią (np. uwagi redakcyjne).
- `zakon.tex` – początkowo pusty, zawiera elementy rocznika mające się ukazać po części bibliograficznej.

Format plików z danymi

Rocznik slawistyczny dzieli się na rozdziały. Każdy rozdział może zawierać rekordy, podrozdziały i odsyłacze. Znajduje to swoje odzwierciedlenie w formacie plików z danymi. Pliki te są produkowane i przetwarzane przez menedżera bazy danych.

Pliki z rozdziałami. Pliki zawierające poszczególne rozdziały rozpoczynają się komendą: \rozdzial{numer}{naglowek}; numer określa zagłębienie rozdziału i może być liczbą naturalną między 1 a 7 (dopuszczamy tylko 7 możliwych zagłębień), naglowek to tytuł rozdziału.

Kolejnym elementem pliku są opisy poszczególnych pozycji bibliograficznych. Są one postaci:

```
\praca{
  \autor{}
  \tytul{}
  \odpow{}
  \odpow2{}
  \wydan{}
  \stron{}
  \inne{}
  \seria{}
  \trac{}
  \abstrakt{}
  \haslo{}
  \sciezkag{}
  \sciezkap{}
}
```

w przypadku pracy książkowej lub:

```
\recenzja{
  \autor{}
  \tytul{}
  \odpow{}
  \odpow2{}
  \wydan{}
  \stron{}
  \inne{}
  \seria{}
  \trac{}
  \rocz{}
  \haslo{}
  \sciezkag{}
  \sciezkap{}
}{
  \recenzje{
    \autorrec{}
    \tytulrec{}
    \rok{}
    \innerec{}
    \haslorec{}
}
```

w przypadku recenzji. Znaczenie poszczególnych pól i standardowy sposób wyświetlania zestawiono w tabeli obok.

komenda	sposób definicji	znaczenie
\autor{}	{\bf #1}	imię i nazwisko autora (ew. nazwiska oddzielone przecinkami)
\tytul{}	{#1}	pełny tytuł pracy (z ewentualnymi podtytułami, częściami, itp.)
\odpow{}	{#1}	pole odpowiedzialności
\odpow2{}	{#1}	pomocnicze pole odpowiedzialności
\wydan{}	{#1}	opis wydania (numer, wydawnictwo, rok)
\stron{}	{#1}	liczba stron
\inne{}	{#1}	inne elementy opisu
\seria{}	{#1}	seria
\trac{}	{#1}	nazwisko autora w oryginalnym brzmieniu
\rocz{}	{#1}	w przypadku recenzji numer rocznika slawistycznego, w którym ukazała się recenzowana praca
\abstrakt{}	{#1}	w przypadku pozycji książkowej
\haslo{}	{}	klucz do sortowania
\sciezkag{}	{}	ścieżka do rozdziału, w którym umieszczona została praca
\sciezkap{}	{}	ścieżki do rozdziałow w których zostanie umieszczona referencja do pracy
\recenzje{	{\item #1}	recenzje
\autorrec{}	{\bf #1}	autor recenzji
\tytulrec{}	{#1}	tytuł recenzji, tytuł czasopisma
\rok{}	{#1}	rok wydania czasopisma
\innerec	{#1}	inne elementy opisu
\haslorec	{}	klucz do sortowania

Tabela 1. Znaczenie i standardowy sposób wyświetlania pól

Pola \recenzje i \sciezkap mogą wystąpić wielokrotnie. Pola z pustym argumentem nie są umieszczane w pliku.

Ponadto, został dodany typ pracy *artykuł* z polami takimi jak w przypadku pracy książkowej.

Po ciągu opisów pozycji książkowych w pliku może wystąpić ciąg komend \input <nazwa> w celu dołączenia podrozdziałów danego rozdziału.

Ostatnim elementem pliku jest (opcjonalna) komenda: \referencje, która dołącza ewentualne odnośniki do prac z innych rozdziałów. Jej parametrem jest ciąg nazw rekordów oddzielony przecinkami i zakończony kropką.

Odsyłacze. Ponieważ jedna praca może wystąpić w kilku rozdziałach, konieczne jest opcjonalne nadawanie nazw pozycjom bibliograficznym na wewnętrzny użytek TEX-a. W tym celu każda z komend `\praca` i `\recenzja` mają opcjonalny pierwszy argument, którym jest jej nazwa. Nazwy tworzone są automatycznie przez menedżera bazy danych poprzez dołączenie do litery *r* litery oznaczającej numer kopii oraz kolejnego numeru rekordu. **Uwaga:** Każda kopia programu powinna mieć inny numer, w przeciwnym razie wystąpić mogą kłopoty przy łączeniu baz danych za pomocą `merge`.

Indeks. Każde nazwisko może być parametrem komendy `\indeks`. Powoduje ona oprócz normalnego pojawienia się nazwiska autora, utworzenie wejścia do indeksu w jednej z trzech form:

- jeśli nie posiada opcjonalnego argumentu, to w indeksie po nazwisku autora pojawi się numer pozycji bibliograficznej;
- jeśli posiada opcjonalny argument `rec`, to w indeksie po nazwisku autora ukaże się tekst *rec.* i numer pozycji bibliograficznej;
- jeśli posiada opcjonalny parametr `patrz{}`, to w indeksie po nazwisku autora pojawi się odsyłacz *zob.* po czym pojawi się inne nazwisko.

Komendy indeksowania są automatycznie dodawane przez menedżera w następujący sposób:

- do każdego autora w polu `haslo` bez argumentu opcjonalnego;
- do każdego nazwiska w polu `autor` różnego od nazwiska z pola `haslo` z odsyłaczem do nazwiska z pola `haslo`;
- do każdego nazwiska w polu `haslorec` z argumentem `rec`;
- do każdego nazwiska z pola `autorrec` różnego od nazwiska z pola `haslorec` z odsyłaczem do nazwiska z pola `haslorec`.

Rozdziały. Dodatkowe definicje rozdziałów znajdują się w pliku `bibdef.sty`. (LATEX dopuszcza niestety tylko 5 poziomów zagłębienia).

Pakiet bazy danych

Pakiet zarządzający bazą danych składa się z następujących części:

- pełna instalacja TEX-a i LATEX-a z fontami standardowymi i dostarczonymi przez Instytut Slawistyki fontami zawierającymi znaki ze wszystkich alfabetów słowiańskich;
- program `tex386` wersja beta (zwykły `tex286` nie dawał sobie rady z ilością materiału);
- pakiet archiwizatora;
- pliki skryptowe `.bat` umożliwiające szybkie tworzenie archiwów i zarządzanie nimi;
- program `merge.exe` w dwóch wersjach umożliwiający łączenie dwóch *rozłącznych* fragmentów baz danych znajdujących się na różnych komputerach;
- dodatkowy program `makeindx` do tworzenia indeksów;
- główna aplikacja menedżera programów.

Archiwizator. Jest to jedna z nowszych wersji programu `arj` wraz z pomocniczymi plikami wsadowymi. Umożliwia ona wydajne pakowanie informacji stanowiących treść bazy.

Merge. Program `Merge` umożliwia połączenie dwóch rozłącznych fragmentów baz danych. Argumentami programu `Merge` są nazwy dwóch plików. Program automatycznie dokona segregacji danych.

Makeindx. Jest to prostsza wersja programu `makeindx` dostarczanego z pakietem LATEX. Oryginalny program nie potrafił wykorzystywać wysokiej pamięci a przez to nie radził sobie z wielotysięcznymi indeksami.

Menedżer bazy danych. Jest to główna aplikacja pakietu.

- Część pierwsza umożliwia wprowadzanie i przetwarzanie danych oraz proste funkcje edycyjne.
- Część druga umożliwia oglądanie wybranych rekordów i pojedynczych rozdziałów w trybie graficznym, realizowanie funkcji wyszukiwania, przygotowywanie rocznika do druku, obsługę słownika skrótów i konwersję danych z dyskietki.

Wewnętrzna reprezentacja danych

Zostały użyte kolekcje TVision w TP7. Baza jest czytana w całości do pamięci i tam przetwarzana.

Interfejs graficzny

Ekran aplikacji jest podzielony na cztery obszary. W górnej części znajduje się menu główne (patrz niżej). W części dolnej znajdują się informacje o przypisaniu klawiszy funkcyjnych. Główny ekran jest podzielony na dwie części. W każdej z nich wyświetlane jest jedno z trzech możliwych okien.

Aplikacja przypomina w wyglądzie i obsłudze popularny program `Norton Commander`.

OKNO EDYTORA. Okno to zawiera pełną informację o przetwarzanym rekordzie i umożliwia jej przetwarzanie.

OKNO DRZEWA. Okno to zawiera strukturę drzewiastą rozdziałów. Umożliwia wybór rozdziału za pomocą myszy. Nawigację usprawnia system skrótów.

OKNO LISTY POZYCJI. Okno to zawiera listę pozycji znajdujących się w aktualnie edytowanym rozdziale. Wyświetlone są też podrozdziały i liczba odsyłaczy.

MENU GŁÓWNE. Wygląd menu jest następujący (w nawiasach podane są aktywne klawisze):

- Lewy
  - Drzewo
  - Lista
  - Edytor
- Baza
  - Konwersja z dyskietki (F1)
  - Zapisz bazę (F2)
  - Koniec pracy (F10)
- Komendy
  - Przenieś rekord (F6)
  - Usuń rekord (F8)
  - Utwórz podrozdział (F7)
  - Obejrzyj rekord (F3)
  - Edytuj rekord (F4)
  - Uporządkuj podrozdziały (F5)
- Szukaj
- Słownik
  - Wprowadź
  - Znajdź
  - Usuń
- Konfiguracja
- Prawy
  - Drzewo
  - Lista
  - Edytor

Wymagania sprzętowe

Ze względu na duży rozmiar bazy proponowana minimalna konfiguracja to: komputer 386DX40MHz, pamięć 8–16MB, dysk twardy 250 MB, monitor VGA mono.

Wykorzystanie SYBISLAW-a

Baza danych SYBISLAW od 4 lat jest używana w Instytucie Slawistyki PAN do generowania slawistycznych roczników bibliograficznych. Ze względu na to, że dane zbierane są w wielu krajach, wprowadzenie LaTeX-a, jako standardu, ułatwiło elektroniczne przekazywanie danych. Przyjazność środowiska i zaawansowane funkcje edycyjne pozwoliły na wyłapywanie niespójności w bazie danych (m.in. wykrywanie dubletów).

Z relacji pani dr Ewy Rudnik-Karwatowej, redaktorki Rocznika, wynika, że większość materiałów z zagranicy trzeba lokalnie przetwarzać na LaTeX-a, choć w jednym ośrodku (Bułgaria) udało się wdrożyć SYBISLAW i dane do Rocznika są wpisywane bezpośrednio w bazie i potem integrowane z resztą danych w Warszawie.

Od czasu wdrożenia bazy powstały 4 roczniki bibliograficzne utworzone w systemie SYBISLAW, a przygotowanie każdego z nich wymaga obecnie znacznie mniej pracy, niż w czasach, gdy prowadzono prace metodą tradycyjną. Od 3 lat nie było żadnych zastrzeżeń co do pracy bazy. Ostatnie modyfikacje były wprowadzone w roku 1995.

Efektem ubocznym zastosowania systemu SYBISLAW jest rozpropagowanie LaTeX-a w środowisku slawistów, gdyż mimo, że baza działa automatycznie, znajomość LaTeX-a jest pożądana przy wprowadzaniu tekstów. Przyjęte TeX-owe konwencje oznaczania znaków regionalnych wymuszają nauczenie się ich przez użytkowników bazy.

Baza nadaje się do bardzo szybkiego przystosowania jej do tworzenia innego typu bibliografii.

◇ Piotr Chrząstowski
`pch@mimuw.edu.pl`