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 hi-

pertekstowymi. 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.

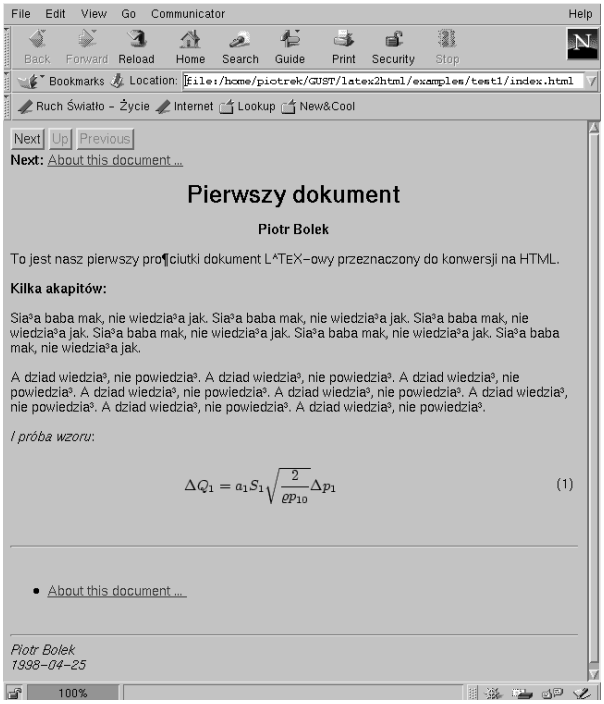
Pierwszy dokument

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}
```

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



Rys. 1.

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.

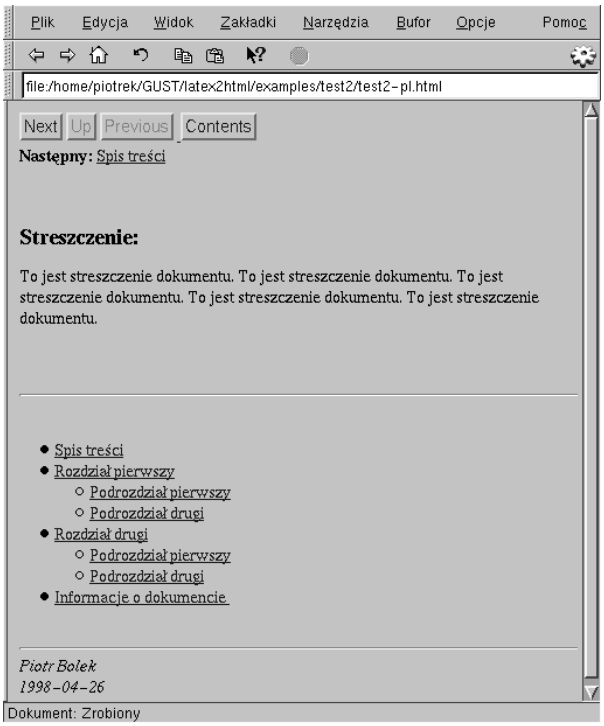
Skonfigurowanie pakietu do obsługi języka polskiego

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



Rys. 2.

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.
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

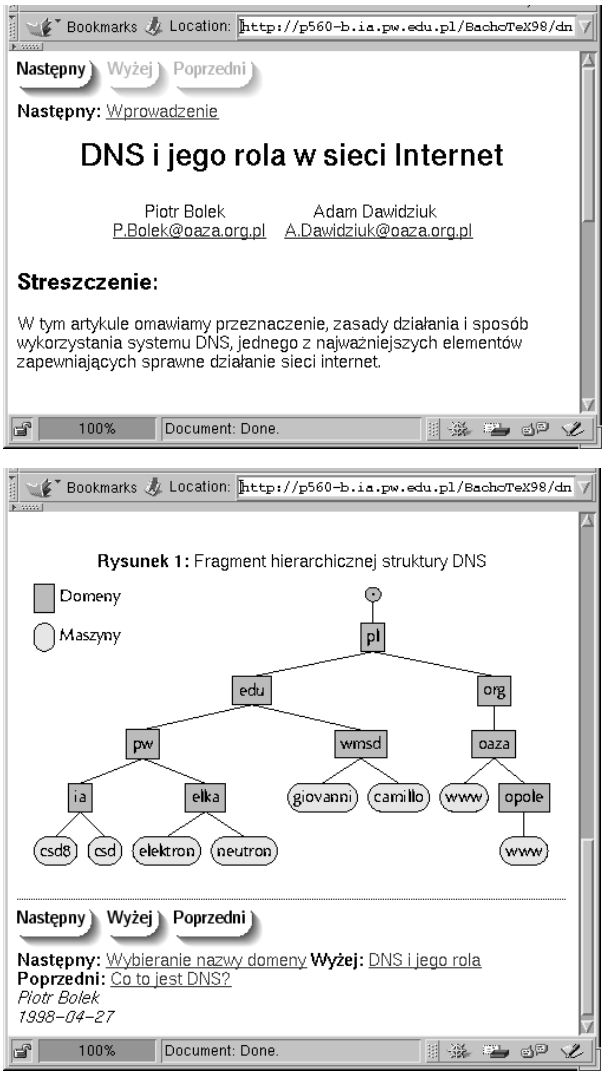
rozmiarów podmienionych ikon (tablica `%iconsizes`). Domyślne nazwy plików graficznych wykorzystywanych jako ikony nie zostały zmienione, więc nie było potrzeby zmiany zawartości tablicy `%icons`.

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
```

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



Rys. 3.

```
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;
```

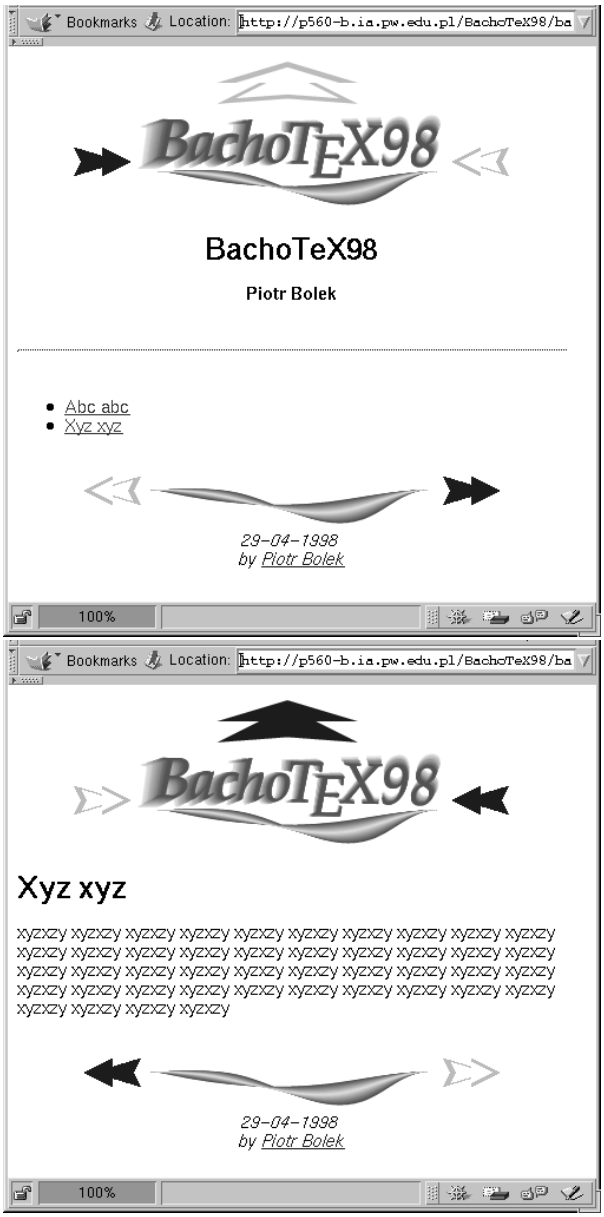
Pierwszy wiersz sprawia, że nie jest generowana strona z informacją o dokumencie. Drugi i trzeci znamy już z poprzedniego przykładu. Wiersze 4, 5, 6 i 7 określają położenie naszych wersji ikon. Parametr `$ALTERNATIVE_ICONS` (wiersze 6 i 7) jest używany jeśli wywołamy konwerter z opcją `-local_icons`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

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



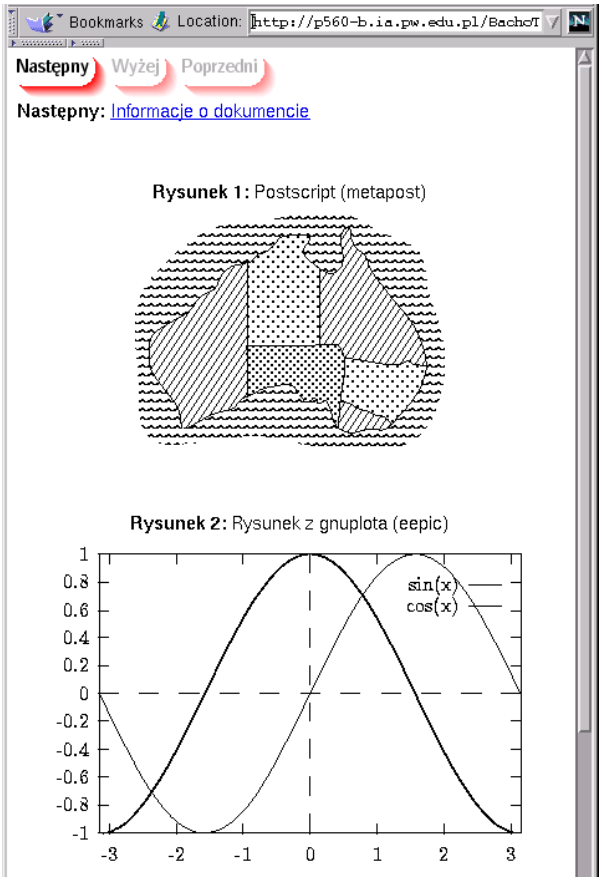
Rys. 4.

`$NEXT`, `$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`.



Rys. 5.

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 `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.

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

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

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

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

Rys. 6.

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

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.

`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 $\mathcal{AMS}$-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/`

Pliki konfiguracyjne do pakietu LaTeX2HTML, opisane w tym artykule, dostępne są na serwerze GUST w katalogu: `ftp://ftp.gust.org.pl/TeX/GUST/contrib/PBolek/latex2html/`.