LaTeX egyenletek és ábrák PHP-ben

Oldalainkba LaTeX tartalmat ágyazva be a ködös múltba száműzhetjük a matematikai egyenletek webes megjelenítésének nehézségeit.

Nyugodtan kijelenthetjük, hogy a weblogok és a wiki oldalak világát éljük. Bár újságok, általános szövegek vagy akár fényképek közzétételére ezek a rendszerek tökéletesen megfelelnek, ám ha egyszerű szövegek és képek kezelésénél többre van szükségünk, korlátaik hamar nyilvánvalókká válnak. A műszaki jellegű weblogok esetében különösen fontos a grafikonok, matematikai kifejezések, diagramok stb. megjelenítésének támogatása. Pusztán *HTML* alapon mindezt meglehetősen nehéz, ha nem egyenesen lehetetlen megoldani.
A külső alkalmazások, mint a *dia*, az *xfig* vagy a *Microsoft Egyenletszerkesztő* használata szintén bonyolult, hiszen először össze kell állítani az ábrát vagy egyenletet az adott alkalmazásban, majd kép formájában fel kell tölteni a weboldalra. Ha egy közös weblog egy másik szerkesztője módosítani szeretne egy ábrát, akkor neki is rendelkeznie kell az adott alkalmazással, illetve a kép készítésekor létrejött eredeti fájllal. Az ilyen megoldások természetes velejárója a bonyolultság, valamint az oldalakon megjelenő egyenletek és egyéb ábrák hullámzó minősége.
Írásomban bemutatom, hogyan lehet a *LaTeX*-et, ezt a kifejezetten műszaki dokumentumok készítésére fejlesztett szedőeszközt és -nyelvet *PHP*-ből hasonló igények kielégítésére használni. A *LaTeX*-et *PHP*-ből hívom meg, amikor a *HTML* tudása az összetett igények kezelésére elégtelennek bizonyul, az eredményeket pedig *PNG* képek formájában jelenítem meg; a *PNG* formátumot minden korszerű böngészőprogram támogatja. Mivel ilyenkor minden szükséges program a kiszolgálón található, minden szerkesztő és felhasználó ugyanazokat az eszközöket és csomagokat használhatja anyagai közzétételére.

Miért nem MathML?

A *W3C* szerint a *MathML* egy alacsony szintű *XML* szabálygyűjtemény matematikai anyagok leírására.
Bár a *MathML* emberi szem számára is olvasható, a legegyszerűbb esetektől eltekintve az *XML* kód előállítására egyenletszerkesztőt vagy egyéb segédprogramot kell használni. Mindemellett a jelenlegi böngészők a *MathML* nyelvnek csak egy részét támogatják, sok esetben azt is csak külső beépülő modul segítségével. Bár maga a nyelv valószínűleg ígéretes jövő elé néz, jelenlegi támogatottsága és használhatósága gyakorlatilag nulla.

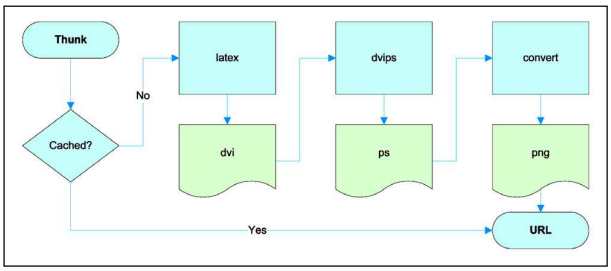
A helyzetet tovább árnyalja, hogy *Leslie Lamport LaTeX* szedőrendszere a műszaki és tudományos dokumentumok készítése terén gyakorlatilag szabvánnyá vált.
A *Donald Knuth* az 1970-es évek elejéről származó *TeX* dokumentumszervező rendszerére épülő *LaTeX* valamikor 1994-ben jelent meg. Mára kiforrott, közismert, elkötelezett felhasználói bázissal rendelkező, műszaki dokumentumok készítésére használt megoldássá vált. Természetesen szó sincs arról, hogy a *LaTeX* megismeréséhez elég volna egy róla szóló leírást a párnánk alá helyezni. Éppen ellenkezőleg. Ám akkor is tény, hogy a *MathML* jelenleg még nem képes megfelelő alternatívát kínálni a meglévő rendszerrel szemben.

Követelmények

A *UNIX* „írjunk egymással együttműködni képes programokat” filozófiáját követve ismert linuxos eszközöket fogok egymáshoz csatolni. A programok egymással együttműködve fogják előállítani a *LaTeX* forrás *PNG* formátumú megfelelőjét. A *LaTeX* egy újabb, a *dvips* és az *ImageMagick* eszközkészlettel kiegészített változatára lesz szükségünk. A végeredményt az *ImageMagick* készlet *convert* segédprogramjával fogjuk *PNG* képpé alakítani. Szerencsére a legtöbb héj hozzáférést is biztosító tárhelyszolgáltató eleve rendelkezik ezekkel az eszközökkel.

Áttekintés

A leképező rendszer egy szöveges karakterláncot vesz át, majd további feldolgozásra kiolvassa belőle a `[tex]` és `[/tex]` párok közé illesztett részeket. Ezeket a kinyert szegmenseket *thunkoknak* nevezzük. Ha egy *thunk* korábban már feldolgozásra került, vagyis a kódjának már van képi megfelelője, akkor helyére egy erre a képre vezető *URL* kerül. Ha új thunkról van szó, akkor átadásra kerül a *LaTeX*-nek, amelynek kimenete egy *DVI* fájl formájában jelenik meg. A *DVI* fájlt ezután az *ImageMagick* segítségével *PNG* fájllá alakítjuk, majd behelyezzük a gyorsítótár könyvtárba. Az újonnan létrehozott kép *URL*-jét az eredeti szövegben szereplő *thunk* helyére tesszük. Amikor az összes thunk feldolgozása befejeződött, a kapott szöveget átadjuk a hívónak. Az egyes thunkok átalakításának folyamatát az *1. ábra* szemlélteti.



1. ábra A thunkok leképezésének folyamatábrája

Használat

Azt hiszem, a legjobb az, ha felülről lefelé haladva először a leképező folyamat meghívásának módjával ismerkedünk meg, a megvalósítás részleteit hagyjuk későbbre. A motor szerepét egy egyszerű ***HTML*** felület tölti be, mely lehetőséget biztosít a ***LaTeX*** leképező rendszer tesztelésére. Segítségével láthatjuk, hogyan kell meghívni a ***render (leképező)*** osztályt. Kezdésnek az *1. kódrészletben* szereplő egyszerű sablont állítottam össze.

A fenti ***PHP*** alapú oldal egy űrlapot biztosít a ***LaTeX*** kód beírására, majd a `transform` eljárással a thunkokat a kész ***PNG*** képek ***URL***-jeire cseréli. Minden más művelet a háttérben, a `render` osztályban történik.

Alapszintű beállítások

A `render` osztály váza a *2. kódrészletben* található.

A ***PHP***-vel közölnünk kell, hogy eszközeink hol találhatók, valamint meg kell adnunk egy könyvtárat, ahova a ***PHP*** kiírhatja ideiglenes fájljait és gyorsítótárát. Az `URL_PATH` értéket szintén meg kell adni, erre a ***HTML***-ben szereplő image címkék előállításakor lesz szükség.

A látszólagos egyszerűség senkit ne tévesszen meg.

A kimenő ***PNG*** fájl módosítása céljából a ***LaTeX***-nek és az ***ImageMagicknek*** rengeteg beállítást lehet megadni, és ezekkel érdemes is megismerkedni. Itt csak a mindezt segítő keretrendszert mutatom be.

A wrap eljárás

A ***wrap (burkoló)*** eljárás fogja a ***LaTeX*** thunkot, majd bevezető és záró résszel látja el, így képez belőle érvényes ***LaTeX*** forrásfájlt. Az eljárás hasonló ahhoz, mint amikor külső eljárásokat illesztünk be egy C fájlba, vagy Java program készítésekor csomagok beemelésével bővítjük a nyelv tudását. (*3. kódrészlet*)

Mint látható, a ***LaTeX*** wrapper használata során általánosan szükséges csomagokat illesztem be. Ilyen az ***Amerikai Matematikai Társaság (AMS)*** csomagja, mely további matematikai elemeket tartalmaz, továbbá a vektoros grafikák leképezésére használható ***PSTricks*** csomag.

A `pagestyle` beállítása `empty` ***(üres)***, így a képekbe nem kerülnek oldalszámok. A thunk a `document` címkék közé kerül.

Lehetséges, hogy rendszerünkön a fenti csomagok egy része nem áll rendelkezésre. Ha alap ***LaTeX*** rendszerünk tudását növelni szeretnénk, akkor a ***Comprehensive TeX Archive Network (CTAN)*** weboldalról tölthetünk le további csomagokat. (Lásd az internetes forrásokat.) Szerezhetünk például oszlopdiagramok, ***UML*** jelölések és

1. kódrészlet render_example.php

```
<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML
➥1.1//EN”

“http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd”>
<html>
<head>
<title>LaTeX egyenletek és ábrák PHP
➥alatt</title>
</head>
<body>
<!– a LaTeX kód beírására szolgáló űrlap –>
<form action=”render_example.php” method=”post”>
<textarea rows=”20”
          cols=”60”
          name=”render_text”></textarea><br />
<input name=”submit”
       type=”submit”
       value=”Leképezés” />
</form>
<?php
if (isset($_POST[‘submit’])) {
   echo ‘<h1>Result</h1>’;
   require(‘render.class.php’);
   $text = $_POST[‘render_text’];
   if (get_magic_quotes_gpc())
      $text = stripslashes($text);
   $render = new render();
   echo $render->transform($text);
}
?>
</body>
</html>
```

2. kódrészlet render.php

```
class render {
 var $LATEX_PATH = “/usr/local/bin/latex”;
 var $DVIPS_PATH = “/usr/local/bin/dvips”;
 var $CONVERT_PATH = “/usr/local/bin/convert”;
 var $TMP_DIR =  “/usr/home/barik/public_html/
                 ➥gehennom/lj/tmp”;

 var $CACHE_DIR = “/usr/home/barik/public_html/
                  ➥gehennom/lj/cache”;

 var $URL_PATH = “http://www.barik.net/lj/
                 ➥cache”;

 function wrap($text) { ... }
 function transform($text) { ... }
 function render_latex($text) { ... }
}
```

3. kódrészlet wrap.php

```
function wrap($thunk) {
s   return <<<EOS
    \documentclass[10pt]{article}
    % itt adhatjuk hozzá a további csomagokat
    \usepackage{amsmath}
    \usepackage{amsfonts}
    \usepackage{amssymb}
    \usepackage{pst-plot}
    \usepackage{color}
    \pagestyle{empty}
    \begin{document}
    $thunk
    \end{document}
EOS;
}
```

Karnaugh-térképek készítésére alkalmas csomagot. Bármire is van szükségünk, a keresést mindig kezdjük a gyűjteménnyel.

A render_latex eljárás

A `render_latex` eljárás (*4. kódrészlet*) az összes thunkot kigyűjti, majd egyenként dolgozza fel őket.
A thunk átadott érték szerepe egyértelmű: ez az éppen vizsgált ***LaTeX*** kódrész. A hash átadott érték a thunk ***md5*** kivonata.
Megváltoztatom az ideiglenes könyvtárat, majd a thunkot egy ideiglenes ***LaTeX*** fájlba írom. A ***LaTeX*** ezután létrehoz egy ***DVI*** fájlt. A ***LaTeX***-et a megfelelő parancssori kapcsolóval utasítom a nem interaktív működésre. A létrejött ***DVI*** fájlt a ***dvips*** segítségével ***PostScript*** formátumra hozom, eközben a -E kapcsolóval egy szövegdobozba illesztem. Ezután a ***PostScript*** fájlt átadom a `convert`-nek, mely ***PNG*** képet készít belőle. A `convert` segédprogramnak rendkívül sok kapcsolója van; az, hogy melyik beállításhalmaz a leginkább megfelelő, az adott webhelytől függ.
Végül, nem árt tudni, hogy az `exec` parancs egy hibajelző állapotkóddal tér vissza. A rövidség kedvéért a hibaellenőrzést itt elhagytam, és feltételeztem, hogy minden művelet sikeresen befejeződik. A ***LaTeX***-nek van néhány veszélyes kapcsolója is, amelyek használata egy többfelhasználós webkiszolgálón gondokat okozhat. Ha tehát bizonyos kulcsszavakat találunk a thunkban, inkább adjunk hibajelzést.

Amikor valami félrecsúszik

Ha a leképezési folyamat során valamilyen hiba történik, akkor próbáljunk kézzel átalakítani egy ***LaTeX*** fájlt. A hibakeresést az alábbi parancsokkal végezhetjük el:

```
latex –interaction=nonstopmode sajat.tex
dvips -E sajat.dvi -o sajat.ps
convert -density 120 sajat.ps sajat.png
```

Így kideríthetjük, hogy pontosan melyik lépésnél akad el a ***LaTeX*** leképezés.

4. kódrészlet render_latex.php

```
function render_latex($thunk, $hash) {
  $thunk = $this->wrap($thunk);
  $current_dir = getcwd();
  chdir($this->TMP_DIR);
  // ideiglenes LaTeX fájl létrehozása
  $fp = fopen($this->TMP_DIR . "/$hash.tex",
        ➥"w+");
  fputs($fp, $thunk);
  fclose($fp);
  // a LaTeX futtatásával ideiglenes DVI fájl
  ➥létrehozása
  $command = $this->LATEX_PATH .
             " –interaction=nonstopmode " .
             $hash . ".tex";
  exec($command);
  // a dvips futtatásával ideiglenes PS fájl
  ➥létrehozása
  $command = $this->DVIPS_PATH .
             " -E $hash" .
             ".dvi -o " . "$hash.ps";
  exec($command);
  // a PS fájl átadása a ImageMagicknek, amely
  // létrehozza a PNG fájlt
  $command = $this->CONVERT_PATH .
             " -density 120 $hash.ps
             ➥$hash.png";
  exec($command);
  // a fájl átmásolása a gyorsítótár könyvtárba
  copy("$hash.png", $this->CACHE_DIR .
       "/$hash.png");
  chdir($current_dir);
}
```

5. kódrészlet cleanup.php

```
function cleanup($hash) {
  $current_dir = getcwd();
  chdir($this->TMP_DIR);
  unlink($this->TMP_DIR . "/$hash.tex");
  unlink($this->TMP_DIR . "/$hash.aux");
  unlink($this->TMP_DIR . "/$hash.log");
  unlink($this->TMP_DIR . "/$hash.dvi");
  unlink($this->TMP_DIR . "/$hash.ps");
  unlink($this->TMP_DIR . "/$hash.png");
  chdir($current_dir);
}
```

A cleanup eljárás

A ***LaTeX*** leképezési folyamat során elég sok ideiglenes fájl jön létre. A `cleanup` ***(takarítás)*** eljárással törölhetjük ezeket – nem mintha valami bonyolult dologról volna szó. (*5. kódrészlet*)

6. kódrészlet transform.php

```
function transform($text) {
  preg_match_all("/\[tex\](.*?)\[\/tex\]/si",
  ➥$text, $matches);
  for ($i = 0; $i < count($matches[0]); $i++) {
    $position = strpos($text, $matches[0][$i]);
    $thunk = $matches[1][$i];
    $hash = md5($thunk);
    $full_name = $this->CACHE_DIR . "/" .
                 $hash . ".png";
    $url = $this->URL_PATH . "/" .
           $hash . ".png";
    if (!is_file($full_name)) {
      $this->render_latex($thunk, $hash);
      $this->cleanup($hash);
    }
    $text = substr_replace($text,
      "<img src=\"$url\" alt=\"Formula:
      ➥$i\" />",
      $position, strlen($matches[0][$i]));
  }
  return $text;
}
```

A transform eljárás

A *6. kódrészletben* látható `transform` eljárás a leképező osztályt irányítja, illetve nyilvános hozzáférési pontot biztosít a programozónak.

A *PHP* `preg_match_all` függvénye kigyűjti az összes thunkot, illetve ezek helyzetét. Ezután a hurokban sor kerül a thunkok egyenkénti feldolgozására. A következő lépés egy egyedi md5 kivonat készítése az adott thunk szövege alapján. Ebből tudjuk eldönteni, hogy egy adott thunk korábban már bekerült-e a gyorsítótárba. Ha igen, akkor meghívjuk a ***LaTeX*** leképező eljárást, és azonnal töröljük a létrejött ideiglenes fájlokat. A thunkot mindkét esetben egy ***URL***-lel helyettesítjük. Amikor az összes thunkot feldolgoztuk, a `text` változóval térünk vissza.

Egyenletkészítési példák

Nézzünk néhány példát a ***LaTeX*** segítségével előállítható egyenlettípusokra. Az egyenletek túlnyomó részét a ***Helmut Kopka*** és ***Patrick W. Daly*** által írt A ***Guide To LaTeX*** című könyvből vettem, ezt sokan az egyik legfontosabb ***LaTeX***-es alapműnek tartják.

$$\frac{a^2 - b^2}{a + b} = a - b$$

2. ábra Példa – törtek

```
[tex]
\begin{displaymath}
\frac{a^2 - b^2}{a + b} = a - b
\end{displaymath}
[/tex]
```

$$\mathrm{corr}(X,Y) = \frac{\displaystyle\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\left[\displaystyle\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2\right]^{1/2}}$$

3. ábra Példa – két változó, X és Y kapcsolata

```
[tex]
\begin{displaymath}
\mathop{\mathrm{corr}}(X,Y)=
\frac{\displaystyle
\sum_{i=1}^n(x_i-\overline x)
(y_i-\overline y)}
{\displaystyle\biggl[
\sum_{i=1}^n(x_i-\overline x)^2
\sum_{i=1}^n(y_i-\overline y)^2
\biggr]^{1/2}}
\end{displaymath}
[/tex]
```

$$I(z) = \sin(\frac{\pi}{2}z^2)\sum_{n=0}^{\infty}\frac{(-1)^n\pi^{2n}}{1\cdot 3\cdots(4n+1)}z^{4n+1} - \cos(\frac{\pi}{2}z^2)\sum_{n=0}^{\infty}\frac{(-1)^n\pi^{2n+1}}{1\cdot 3\cdots(4n+3)}z^{4n+3}$$

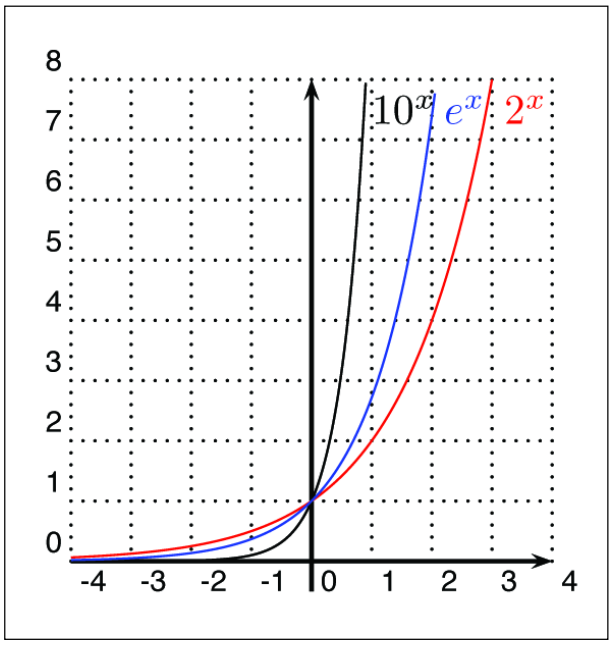
4. ábra Példa – összetettebb egyenlet

```
[tex]
\begin{displaymath}
I(z) = \sin( \frac{\pi}{2} z^2 ) \sum_{n=0}^\infty
    \frac{ (-1)^n \pi^{2n} }{1 \cdot 3
    \cdots (4n + 1) } z^{4n + 1}
    -\cos( \frac{\pi}{2} z^2 ) \sum_{n=0}^\infty
    \frac{ (-1)^n \pi^{2n + 1} }{1 \cdot 3
    \cdots (4n + 3) } z^{4n + 3}
\end{displaymath}
[/tex]
```

Grafikonrajzolási példák

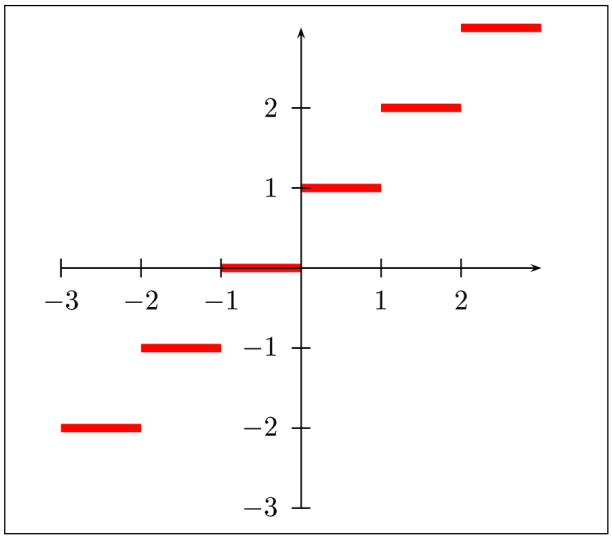
Bár a ***LaTeX*** elsősorban matematikai szedésre szolgál, kiegészítő csomagok segítségével, mint például a ***PSTricks***, más területeken is jól alkalmazható. A grafikonokat ***Herbert Vosstól*** kaptam. Weblapján (lásd a forrásokat) további példákat is lehet találni arra, hogy a ***PSTricks*** segítségével hogyan lehet tesztelni a ***LaTeX*** leképező rendszert. Összetettebb példáinak helyes megjelenítése, miért is tagadnánk, bizony komoly munkával jár.

```
[tex]
\psset{unit=0.5cm}
\begin{pspicture}(-4,-0.5)(4,8)
\psgrid[subgriddiv=0,griddots=5,
   gridlabels=7pt](-4,-0.5)(4,8)
\psline[linewidth=1pt]{->}(-4,0)(+4,0)
\psline[linewidth=1pt]{->}(0,-0.5)(0,8)
\psplot[plotstyle=curve,
   linewidth=0.5pt]{-4}{0.9}{10 x exp}
\rput[l](1,7.5){$10^x$}
\psplot[plotstyle=curve,linecolor=red,
```



5. ábra Példa – a 10^x, az e^x és a 2^x függvény grafikonja

```
   linewidth=0.5pt]{-4}{3}{2 x exp}
\rput[l](2.2,7.5){\color{blue}$e^x$}
\psplot[plotstyle=curve,linecolor=blue,
   linewidth=0.5pt]{-4}{2.05}{2.7183 x exp}
\rput[l](3.2,7.5){\color{red}$2^x$}
\rput(4,8.5){\color{white}change\normalcolor}
\rput(-4,-1){\color{white}bounding
box\normalcolor}
\end{pspicture}
[/tex]
```



6. ábra Példa – a Ceil függvény

```
[tex]
\SpecialCoor
\begin{pspicture}(-3,-3)(3,3)
   \multido{\i=-2+1}{6}{%
      \psline[linewidth=3pt,linecolor=red]
      (\i,\i)(! \i\space 1 sub \i)}%
      \psaxes[linewidth=0.2mm]{->}(0,0)(-3,-3)(3,3)
\end{pspicture}
[/tex]
```

A rendelkezésre álló megvalósítások

Jelenleg a weben számos *LaTeX* leképezőt lehet találni, ezek egy része jobban, más része kevésbé jól működik. *Steve Mayer* például jelenleg *Benjamin Zeiss* eredeti *PHP*-s *LaTeX* leképezőjét tartja karban. *Mayer* több beépülő modult is készített népszerű weblog rendszerekhez, többek közt a *WordPresshez*. Ha valaki moduláris megoldást keres a weboldalához, akkor ezt bátran tudom javasolni.
Érdemes még megemlíteni *John Walker textogif*-jét. Ez egy *Perl* program, mely a *LaTeX2HTML* eszközökkel, *CGI* alapon képezi le a képeket *GIF* vagy *PNG* formátumba. Szintén kiváló megoldás *John Forkosh* C-ben íródott, *CGI*-ként futó *mimeTeX*-e. Előnye, hogy használatához nincs szükség a *LaTeX*-re vagy az *ImageMagickre*, ám ennek a leképezési minőségben jelentkezik az ára.

Összefoglalás

A *LaTeX*-et *wiki* vagy *weblog* oldallal összeilleszteni első látásra rémisztő feladatnak tűnik. Ha viszont sikerül ráéreznünk a megoldásra, többé nem fogjuk érteni, korábban hogyan bírtuk ki nélküle. A fenti modellt követve azt is láthatjuk, hogy a *LaTeX* mellett más nyelveket hogyan tudunk beágyazni a *PHP*-be. Érdemes még megfontolni a *Gnuplot* használatát függvényábrázolások előállítására, míg az *Octave* segítségével komplex kifejezéseket tudunk kiértékelni, a *POV-Rayjel* pedig 3D ábrákat tudunk készíteni.
Jelenleg a weblogíró közösség által lefedett témakörök, viszonylagos sokszínűségük ellenére, eléggé egyoldalúnak mondhatók. A programozástól különböző területekkel foglalkozó műszaki szakemberek egyelőre távol maradnak a weblogírás világától, egyszerűen azért, mert a gondolataik átadásához szükséges eszközök hiányoznak. Bízom benne, hogy a webes *LaTeX* leképező rendszer segítségével sikerül majd áthidalni ezt a szakadékot.

Linux Journal 2005. március, 131. szám

Titus Barik kisvállalkozásokkal foglalkozó informatikai tanácsadó. Aktív weblogíró és műszaki könyvmoly. Weblogja a barik.net címen található.

KAPCSOLÓDÓ CÍMEK

- www.mayer.dial.pipex.com/tex.htm
- www.fourmilab.ch/webtools/textogif/textogif.html
- www.forkosh.com/mimetex.html
- www.pstricks.de
- www.imagemagick.org
- tech.irt.org/articles/js081