Włączanie grafik w formacie EPS do tekstów w $\LaTeX\,2_\varepsilon$ i parę innych uwag

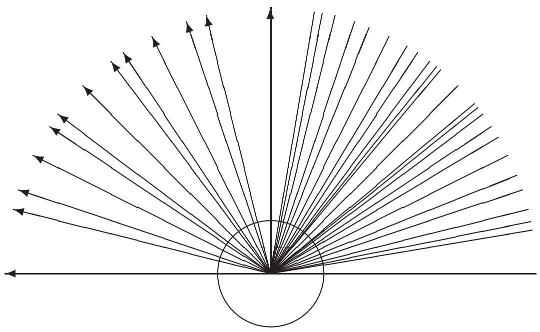
Wojciech Myszka

11 stycznia 1999 roku

Spis treści

1 Wstęp

System LaTeX nie był nigdy pomyślany jako program w którym **tworzy się** grafikę. W czasach kiedy powstawał (La)TeX nie znane były jeszcze powszechnie używane dziś formaty graficzne (PostScript, GIF, JPEG). Dostępny zestaw poleceń (`\line`, `\vector`, `\circle`, . . . `box`, `\oval`) pozwala jedynie na rysowanie linii prostych (o ograniczonych nachyleniach), wektorów (linia prosta[1] z grotem na końcu), okręgów (o średnicach ograniczonych do ok. 40pt), kół (o średnicach z jeszcze węższego zakresu), prostokątów i prostokątów o zaokrąglonych rogach.



Rysunek 1: Nachylenia linii i wektorów dostępne w $\LaTeX\,2_\varepsilon$ (oraz okrąg o największej średnicy)

Do rysowania krzywych o nieco bardziej wymyślnych kształtach służy polecenie `\bezier` składające je z pojedynczych punktów.

Wszystko bardziej skomplikowane musi być przygotowane jakimś innym programem i **włączone** do tekstu jako **zewnętrzny** obiekt. Poza zakresem naszych zainteresowań jest **wybór** programu użytego do zrobienia wykresu, szkicu czy schematu.

2 Włączanie grafik przygotowanych przez inny program

Do włączania plików graficznych w systemie $\LaTeX\,2_\varepsilon$ służy polecenie `\includegraphics`. Polecenie pozwala na włączenie praktycznie dowolnego obiektu graficznego. Jest jednak bardzo zależne od używanej wersji (implementacji) systemu TeX.

[1] Zestaw nachyleń wektorów jest jeszcze uboższy!

I tak, używany przez nas do niedawna emTEX pozwalał na łatwe włączanie rysunków w postaci czarno-białych plików graficznych PCX albo BMP.

Używany obecnie program MiKTEX oprócz plików BMP (kolorowych!) pozwala na włączanie plików EPS.

System teTEX (używany na maszynach Unixowych) obsługuje praktycznie wyłącznie grafiki w postaci EPS.

Inne implementacje dają jeszcze inne możliwości. Wydaje się jednak, że pewnego rodzaju standardem staje się przygotowywanie grafik w formacie EPS.

3 Zalety formatu EPS

Format EPS (Encapsulated PostScript) ma szereg zalet:

- Wiele programów pozwala tworzyć grafiki „wektorowe”[2] i zapisywać je jako pliki EPS.
- Włączane obiekty graficzne mogą być kolorowe.
- Obiekty można łatwo skalować, przy czym uzyskany efekt będzie bardzo różny w zależności od sposobu przygotowania obiektu graficznego: obiekty rastrowe skalują się bardzo źle, obiekty wektorowe skalują się bardzo dobrze.
- Każdy „bitowy” (rastrowy) plik graficzny może być do tego formatu przekształcony.

Wadą używania grafik w postaci EPS jest konieczność wydruku na drukarce PostScriptowej (albo korzystanie ze specjalnych programów, które przekształcają PostScript do postaci zrozumiałej przez drukarkę – najczęściej **ghostscript** i **ghostview/Gsview/gv**).

4 Tworzenie plików EPS przez inne programy

4.1 Programy systemu Windows

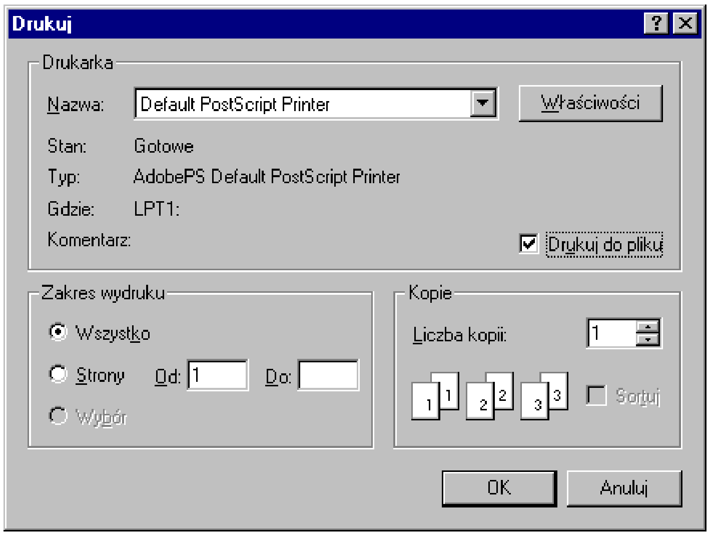
Aby w systemie Windows 95/NT najwygodniej tworzyć pliki EPS trzeba zainstalować najnowszy sterownik drukarki PostScriptowej z serwera Adobe.[3]

Opis w jaki sposób doprowadzić do utworzenia „dobrego” pliku EPS przedstawiamy poniżej na przykładzie programu SigmaPlot w systemie Windows NT.

1. Rysunek przygotowywujemy w programie SigmaPlot tak jak zwykle. Na „stronie” papieru powinien być umieszczony tylko jeden rysunek (chyba, że naszym celem jest stworzenie pliku EPS zawierającego kilka powiązanych ze sobą rysunków) bez żadnych dodatkowych napisów (numer strony, nagłówki. . .).

[2] Wektorowe, to znaczy takie, które złożone są ze stosunkowo prostych obiektów (proste, łuki okręgów, krzywe również wyższego stopnia) zadanych parametrycznie: za pomocą współrzędnych (początku, końca, środka,. . .) i pewnych dodatkowych parametrów (promień, kąt, długość).

[3] `http://www.adobe.com/`



Rysunek 2: Wybór drukarki w systemie Windows NT/95

2. Na rysunku można nanosić teksty, korzystać z kolorów i wszystkich (chyba) możliwości, które daje SigmaPLot. Nie znalazłem jeszcze sposobu na użycie polskich liter.

3. Gotowy obrazek musimy teraz zapisać na dysk we właściwym formacie. W tym celu wykonujemy następujące czynności: File|Print... Jako drukarkę wybieramy „Default PostScript Printer”. Stawiamy „ptaszka” w okienku „Drukuj do pliku”.

   Naciskamy klawisz „Właściwości” i wybieramy z „Dokument – Opcje” znajdujące się na samym dole „PostScript Options” naciskając maleńki plusik. Musimy zadbać aby ustawić: „Postscript Output Option: <Encapsulated PostScript (EPS)>

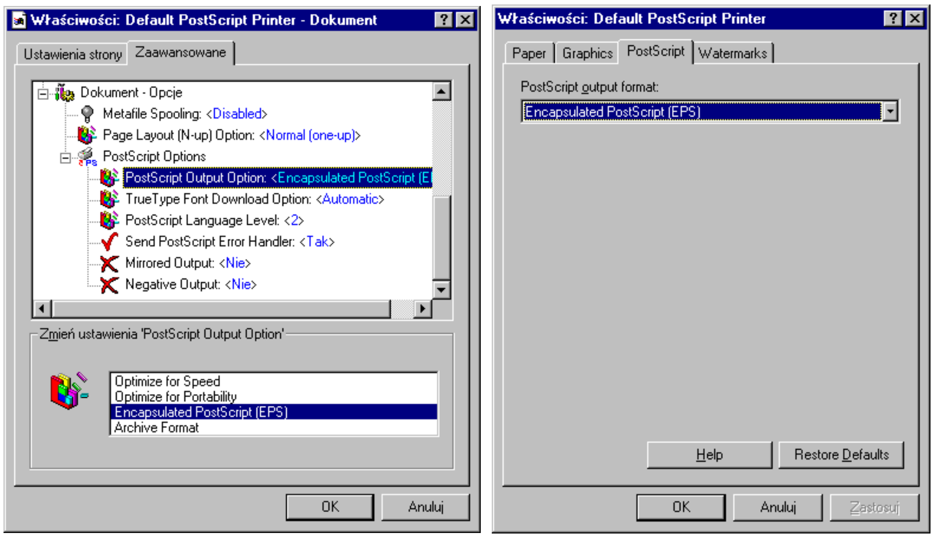
   W przypadku Windows 95 postępowanie jest analogiczne: po naciśnięciu klawisza „Właściwości” należy wybrać zakładkę PostScript i w okienku wybrać „Encapsulated PostScript (EPS)”.

   Naciskamy klawisz „OK” i jeszcze raz „OK” żeby wydrukować. System powinien zapytać nas o nazwę pliku. Podajemy cokolwiek, na przykład `rysunek.ps`.

4. Otrzymany plik PS otwieramy programem **GSview** (wystarczy na niego kliknąć dwa razy). Ghostscript posłuży nam do ostatecznego przekształcenia pliku do właściwej postaci. W tym celu wybieramy File|PStoEPS i w otwartym okienku sprawdzamy czy ptaszek jest przy „Automaticaly calculate Bounding Box” i naciskamy „Yes”. Program poprosi o podanie nazwy pliku, podajemy `rysunek.eps`.

5. Otrzymany plik .eps można już włączać do tekstów pisanych w LaTeX 2_ε poleceniem:

```
\includegraphics{rysunek}
```



Rysunek 3: Ustawienie właściwości przy „drukowaniu” do pliku

Poniżej podaję (niepełną) listę aplikacji Windowsowych generujących poprawne (na tyle, na ile byłem to w stanie osobiście sprawdzić) pliki EPS:

- GNUplot (gnu) – patrz również str. 6,
- Tkpaint (gnu)
- SigmaPlot,
- Mathcad 8 (za wyjątkiem wykresów 3D!),
- Excel,
- PowerPoint,
- PageDraw (freeware),
- Mayura Draw (shareware).

GNUplot, Tkpaint i Mayura Draw posiadają możliwość zapisu (lub eksportu) tworzonych grafik bezpośredni w postaci plików EPS.

Do konwersji grafik bitmapowych do postaci EPS można używać na przykład programu Pain Shop Pro (shareware) lub jakiegoś innego. W każdym przypadku, należy pamiętać o **wyłączeniu** właściwości, która nazywa się „Preview”.

4.2 Programy w systemie Unix

- Mathematica,

  W środowisku tekstowym postępujemy tak: gdy już mamy przygotowany obrazek (niech nazywa się on wykres):

  ```
  In(1):=wykres=Plot[Sin[x], {x, 1, 10}];
  ```

zapisujemy go do pliku `wykres` w formacie EPS poleceniem:

```
In(2):=Display["!psfix -epsf > wykres", wykres]
```

- Matlab,

  Aby wykres zapisać w postaci pliku EPS należy wydać polecenie `print` o następującej postaci:

  `print -d`*devicetype filename*

  Jako *devicetype* podać możemy: `eps` aby zapisać rysunek jako czarnobiały plik EPS lub `epsc` – rysunek kolorowy; użyć można również `eps2` lub `epsc2`.

  filename oznacza nazwę pliku w którym zostanie zapisany rysunek.

- gnuplot,

  Typ „terminala” musimy zdefiniować jako:
  `set terminal postscript eps` *color dashed* `"`*fontname*`"` *fontsize*
  gdzie:

  color przyjmuje wartości „`color`” lub „`monochrome`” (wartość domyślna – `monochrome`),

  dashed przyjmuje wartości „`solid`” lub „`dashed`” (wartość domyślna `dashed`),

  fontname nazwa fontu PS używanego do opisów (wartość domyślna Helvetica),

  fontsize wielkość fontu (domyślnie 14pt).

  Aby wykres zapisać do pliku używamy polecenie `set output` *filename*. Każdy wykres powinien być zapisywany do osobnego pliku!

  Tak na marginesie – pragnę zwrócić uwagę na pakiet **egplot** pozwalający na włączanie w tekst źródłowy (w LaTeXu) poleceń, które zostaną zapisane do pliku, a po przetworzeniu pliku przez gnuplot, w kolejnym przebiegu włączone jako grafiki EPS. Narzędzie pozwala na wygodne zintegrowanie kodu programów generujących rysunki z tekstem który się do nich odwołuje.

  Istnieje również możliwość tworzenia wykresów w innych formatach „zrozumiałych” dla systemu LaTeX. Jako terminal wybrać można:

  – `latex` (rysunek tworzony jest za pomocą elementarnych poleceń języka LaTeX: **circle**, **rule**, **line**, **vector**);

  – `pslatex` (ew. `pstex`) wykresy tworzone są z udziałem specjalnych poleceń języka PostScript włączanych do wynikowego pliku za pomocą poleceń `special`;

  – `eepic` – wymagają użycia pakietu **eepic** i specjalnego drajwera drukarki;

  – `tpic` wymaga drajwerów rozumiejących polecenia **tpic**;

  – `pstricks` wymaga użycia pakietu **pstricks** (por. rozdział 9.2);

- `texdraw` do wykorzystania z pakietem **texdraw**;
- `mf` – przygotowywuje program który powinien być później "przekształcony" do pliku `.pk` – fontu.

Wydaje się, że rysunki w postaci EPS będą rozwiązaniem najwygodniejszym...

- Tkpaint.

  Bardzo sympatyczny program do przygotowywania rysunków wykorzystujący zestaw narzędzi Tcl/Tk (i wymaga jego zainstalowania). Pod wielu względami program jest podobny do programu Xfig. Sam program został opracowany w środowisku Windows (tam dostępny jest również jako samodzielny program, nie wymagających Tcl/Tk) i „przeniesiony" do pracy w środowisku Unix.

- SDRC I–DEAS,

  Zwracam uwagę, że mimo, iż oprogramowanie (w wersji 4) potrafi wyprodukować kolorowe pliki EPS, są z nimi różne problemy:

  - pliki są bardzo duże w przypadku obrazów „cieniowanych",
  - PostScript jest trochę niestandardowy – poszczególne wiersze zakończone są dziwaczną kombinacją znaków co przeszkadza niektórym programom,
  - może zachodzić konieczność „ręcznego" wyspecyfikowania wymiarów rysunku (*Bounding Box*).

- Xfig. Program pozwala na eksport rysunków do postaci EPS.

Do konwersji grafik bitmapowych do postaci EPS można używać programu **convert** z pakietu ImageMagic lub **xv**.

Do „poprawiania" plików EPS mających źle wyznaczony BoundingBox (a również do konwersji[4] plików typu PS do EPS) użyć można programu **ps2epsi**.

4.3 Analizator HP 35xxx

Jeżeli zachodzi potrzeba załączenia „zrzutów" ekranowych analizatora, najlepiej zapisać je na dyskietce tak jak pliki przekazywane na ploter.[5]

Postępowanie prowadzące do zapisania na dyskietce pliku we właściwym formacie jest następujące:

1. Na panelu czołowym naciskamy klawisz „Print/Plot".
2. Naciskamy klawisz koło monitora opisany „More Setup".
3. Ustawiamy „Device is PLOT".
4. Ustawiamy „Output to File".

[4] Zwracam uwagę, że poprawny plik EPS nie może zawierać wielu poleceń języka PostScript. Zatem nie każdy plik PS może być przekształcony do EPS.

[5] Program **hp2xx** może być wykorzystany również do konwersji innych plików zapisanych w standardzie HPGL do innych postaci.

5. Naciskamy klawisz „Return”.
6. Nazwę pliku możemy ustalić po naciśnięciu klawisza „Output Filename” (standardowo analizator nadaje kolejnym plikom nazwy: PLOT1, PLOT2,. . .)
7. Naciskamy „Start Plot/Print”

Plik taki może być przekształcony do postaci EPS za pomocą programu hp2xx (dostępne są wersje pracujące w środowiskach UNIX lub DOS).

Polecenie ma następującą postać:

`hp2xx -m eps -f` *filename file*

gdzie: *filename* to nazwa pliku wyjściowego (w przypadku pominięcia zostanie przyjęta jako *file*) a *file* nazwa pliku wejściowego.

4.4 Konwersja map bitowych do postaci skalowalnej

Jeżeli chcemy rysunek/obrazek w postaci bitowej przekształcić do postaci EPS możemy wykorzystać na przykład jeden z kilku dostępnych programów:

- **convert** z pakietu ImageMagic (środowisko Unix),
- **xv** w środowisku Unix,
- **gws** w środowisku DOS lub Windows,
- **Paint Shop Pro** w środowisku Windows.

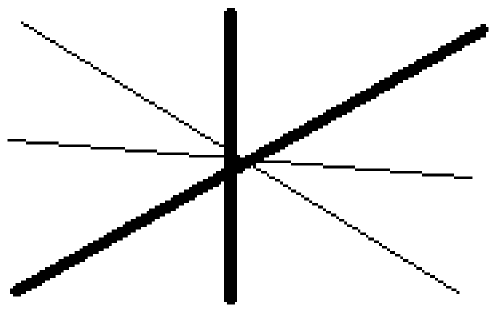
(nie jest to w żadnym wypadku pełna lista programów). W środowisku Windows możemy użyć właściwie dowolnego programu, który potrafi rysunek wydrukować – trzeba jedynie zastosować metodę opisaną wcześniej (por. punkt 4.1).

Powyższe programy zapisują mapę bitową z wykorzystaniem poleceń języka PostScript ale nie zmieniają charakteru grafiki – ciągle składa się ona z pojedynczych „punktów”. W przypadku konieczności skalowania – możemy mieć do czynienia z wszystkimi efektami skalowania map bitowych.

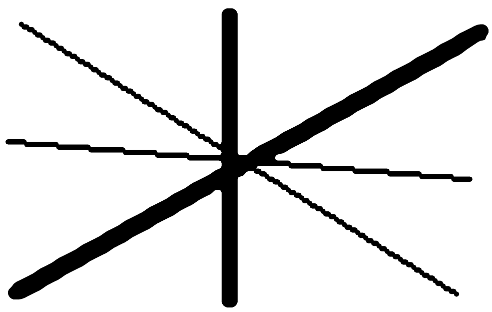
Jeżeli jednak zechcemy przekształcić grafikę rastrową do do postaci wektorowej – napotkamy spore kłopoty.

Znalazłem program o nazwie **kvec** (Shareware) który stara się dokonać tego dzieła, jednak zadowalające efekty uzyskamy jedynie w ograniczonej liczbie przypadków. . .

Zamieszczamy przykład pliku przekształconego do postaci „rastrowego” EPS (rysunek 4)a w drugim do

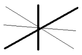
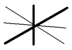


Rysunek 4: „Rastrowy” EPS uzyskany za pomocą programu Paint Shop Pro



Rysunek 5: „Skalowalny” EPS uzyskany za pomocą programu **kvec**

postaci „wektorowego” EPS (rysunek 5). W żadnym wypadku efekt nie jest idealny – niektóre „linie” ciągle nie mają charakteru linii a raczej „schodów” ale...

Na koniec przykład zdjęcia przekształconego programem **kvec**:

5 Polecenie `includegraphics`

Aby użyć polecenia `\includegraphics` w tekście dokumentu, w jego preambule (to znaczy **przed**) `\begin{document}` powinno znaleźć się polecenie:

```
\usepackage{graphicx}
```

Składnia polecenia `\includegraphics` jest następująca:
`\includegraphics[`*parametry_dodatkowe*`]{`*nazwa_pliku_graficznego*`}`
parametry_dodatkowe mogą być następujące:

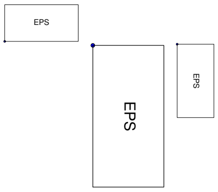
`width` określa szerokość obiektu,

`height` określa wysokość obiektu (normalnie obiekty graficzne skalowane są tak, aby zachować proporcje oryginału pomiędzy wysokością a szerokością; wówczas wystarczy podać tylko jeden z powyższych parametrów),

`keepaspectratio` w przypadku gdy podane są oba powyższe parametry, dodatkowe użycie `keepaspectratio` powoduje, że wstawiony obiekt będzie przeskalowany w taki sposób aby nie przekroczyć żadnego z zadanych rozmiarów i zachować proporcje oryginału,

`angle` określa kąt (w stopniach) obrotu obiektu, liczby dodatnie oznaczają obrót w kierunku przeciwnym do ruchu wskazówek zegara; trzeba pamiętać, że w przypadku dokonywania obrotów wielkość obracanego obiektu zależy od kolejności podawania parametrów `width` lub `height` i `angle`:

```
\includegraphics[width=1cm,angle=0]{eps}
\includegraphics[angle=-90,width=1cm]{eps}
\includegraphics[width=1cm,angle=-90]{eps}
```



`scale` parametr mówi w jakich proporcjach ma być przeskalowany cały obiekt,

`origin` parametr określa współrzędne punktu, wokół którego obracany jest obiekt, normalnie jest to punkt wstawienia obiektu czyli jego lewy, dolny róg,

`clip` parametr żądający aby wszystko to co wykracza poza wymiary obiektu było obcinane,[6]

`bb` określa wymiary rysunku (*Bounding Box*); należy je podać jako cztery liczby oddzielone odstępami; niezbędny gdy plik EPS jest pozbawiony tych informacji[7] lub gdy, LaTeX nie może tych danych z pliku EPS odczytać (na przykład z powodu zbyt długich wierszy),

`draft` powoduje wstawienie zamiast obiektu graficznego tylko nazwy pliku i ramki określającej miejsce zajmowane przez obiekt, bardzo wygodne gdy ciągle pracujemy nad tekstem, a wstawiane grafiki są bardzo skomplikowane; odwrotnością `draft` jest `final`; parametr `draft` może być użyty w wierszu `\usepackage`:

```
\usepackage[draft]{graphicx}
```

i wówczas dotyczy wszystkich włączanych grafik, lub w linii `\documentclass`:

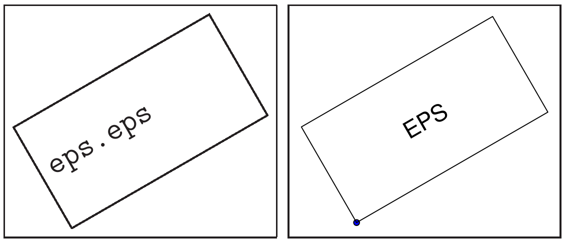
```
\documentclass[draft]{}
```

i wówczas jest to parametr obowiązujący globalnie (i modyfikujący zachowanie również innych elementów systemu LaTeX 2_ε).

```
\fbox{\includegraphics[draft,width=3cm,angle=30]{eps}}
\fbox{\includegraphics[width=3cm,angle=30]{eps}}
```

[6]Normalnie parametr nie jest potrzebny, jednak w przypadku pewnych obiektów graficznych sam rysunek jest nieco większy niż **zadeklarowana** jego wielkość. W takich sytuacjach pominięcie parametru `clip` powoduje, że grafika będzie wykraczała poza przydzielone jej miejsce.

[7]Ale wówczas nie jest prawidłowym plikiem EPS!



Jeżeli po parametrze ma być podana jakaś wartość (`width`, `height`, `angle`...) wówczas między parametrem a wartością powinien być znak = (równość):

```
\includegraphics[width=\textwidth]{obrazek}
```

(w tym przypadku grafika zajmie całą szerokość strony: `\textwidth}`.

6 Kolor w tekście

Oprócz włączania kolorowych ilustracji w tekstach przygotowywanych systemem LaTeX 2_ε zmieniać można tak kolor liter jak i kolor tła na którym się one pojawiają. W preambule dokumentu trzeba zadeklarować wykorzystanie pakietu `color`:

```
\usepackage{color}
```

Kolory mogą być definiowane w jednym z czterech trybów:

rgb kolory definiowane są za pomocą trzech składowych (czerwonej, zielonej, niebieskiej),

cmyk kolory definiowane są za pomocą czterech składowych: *cyan*[8], karmazynowy, żółty i czarny.

gray tak na prawdę nie mamy do czynienia z kolorami tylko z odcieniami szarości,

named można używać tylko wcześniej zdefiniowanych kolorów (ten tryb nie zawsze jest dostępny).[9]

Składowe przyjmują wartości pomiędzy 0 a 1.

Generalnie (poza sytuacją gdy możemy skorzystać z trybu „named") każdy używany kolor (poza white, red, green, blue, cyan, magenta, yellow, black) powinien być zdefiniowany. Służy do tego polecenie:
`\definecolor`{*name*}{*model*}{*color speciffication*}

Poleceniem definiującym obowiązujący kolor tekstu jest: `\color`{*name*}. Działa ono analogicznie jak na przykład `\bf`.

Można też użyć polecenia `\textcolor`{*name*}{*text*} – zmienia ono tylko kolor wskazanego tekstu.

Kolejnym poleceniem jest: `\colorbox`{*name*}{*text*} – tworzy pudełko, którego tło przyjmuje zadany kolor oraz `\fcolorbox`{*name1*}{*name2*}{*text*} (pierwszy parametr określa kolor ramki a drugi kolor tła).

[8] Turkusowy??? tak przynajmniej nazwała go moja córka.

[9] Gdy korzystamy z programu dvips można sprawdzić zawartość pliku `color.pro` – zawarte są tam definicje wszystkich kolorów, które możemy bezpiecznie wykorzystywać.

7 Inne efekty specjalne

7.1 Skalowanie obiektów

Polecenie `\scalebox`{*h-scale*}`[`*v-scale*`]`{*argument*} pozwala[10] przeskalować dowolny obiekt występujący jako *argument*. Gdy *v-scale* zostanie pominięty – obiekt będzie skalowany z zachowaniem proporcji.

```
\scalebox{2}[3]{Ala ma kota}
```

Ala ma kota

Zamiast `\scalebox` można użyć polecenia `\resizebox`{*width*}{*height*}{*argument*}

```
\resizebox{1cm}{1cm}{Ala ma kota}
\resizebox{1cm}{!}{Ala ma kota}
```

Ala ma kota Ala ma kota

powodującego przeskalowanie obiektu do zadanych wymiarów. Użycie jako drugiego argumentu wykrzyknika `!` spowoduje przeskalowanie obiektu z zachowaniem proporcji.

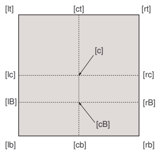
7.2 Obroty obiektów

Polecenie `\rotatebox[`*options*`]`{*argument*} może być wykorzystane do obrotu dowolnego obiektu.

Parametr *options* pozwala na określenie między innymi współrzędnych punktu wokół którego będzie obracany obiekt. Można je podać w postaci `x=`*xdim*`,y=`*ydim* co wskaże bezwzględne współrzędne punktu obrotu.

Alternatywnie można współrzędne podać w sposób „względny” korzystając ze schematu przedstawionego na rysunku 6.

[10] Wymaga wykorzystania pakietu `graphicx`.



Rysunek 6: Sposób oznaczania charakterystycznych punktów obiektu

```
\begin{tabular}{|c|c|}
\hline
\rotatebox{90}{Liczba porzadkowa} & obiekt \\
\hline
      1         &  aaa   \\
      2         &  bbb   \\
\hline
\end{tabular}
```

Liczba porzadkowa	obiekt
1	aaa
2	bbb

Jeżeli zachodzi potrzeba umieszczenia w tekście dużej tabeli ”w poprzek” – można w tym celu użyć pakietu „lscape”:

```
\usepackage{lscape}
```

i środowiska `landscape`

8 Poprawianie plików EPS

Znam takich ludzi, którzy twierdzą, że lepiej jest nauczyć się „programowania” w języku PostScript niż korzystać z jakichś wymyślnych programów do przygotowania niezbyt skomplikowanych rysunków.

Nie namawiam nikogo do takiego działania.

Istnieje jednak czasami potrzeba zmodyfikowania (lub takiego przygotowania) pliku EPS aby umieścić na rysunku symbole normalnie niedostępne w programie którego używamy – na przykład opis w języku polskim czy symbole matematyczne.

Do osiągnięcia takich efektów użyć należy pakietu psfrag. Pozwala on w sposób „automagiczny” podmienić wstawione „znaczniki” zadanymi ciągami znaków.

Dodatkowo pozwala on w opisach umieszczać polecenia LaTeXa, które zostaną odpowiednio „zinterpretowane” na etapie przetwarzania PostScriptu.

Procedura postępowania jest następująca:

1. W dokumencie użyć musimy pakietu `graphicx` (lub `graphics`).
2. Dodatkowo używamy pakietu `psfrag`.
3. W przygotowywanym rysunku umieścić musimy „znaczniki”; są to krótkie teksty, najlepiej **jednowyrazowe**; umieszczamy je w tych miejscach, gdzie chcemy umieścić „specjalne” teksty.
4. W dokumencie używamy polecenia `\psfrag` które spowoduje zastąpienie każdego znacznika zadanym przez nas tekstem.

Polecenie `psfrag` ma następującą postać:
`\psfrag`{*tag*}`[`*pos*`]` `[`*pspos*`]` `[`*scale*`]` `[`*rot*`]`{*replacement*}

Znaczenie poszczególnych parametrów jest następujące:

tag znacznik: tekst, którego **każde** wystąpienie w tekście będzie zastępowane;

pos punkt odniesienia wstawianego tekstu, dwie lietry: jedna z zakresu {`t`,`b`,`B`, `c`} a druga {`l`, `r`, `c`} (patrz również rysunek 6); wartość domyślna `cc`;

pspos punkt odniesienia tekstu PostScriptowego; wartość domyślna `cc`;

scale współczynnik skali, domyślnie 1;

angle kąt obrotu (w stopniach) wstawianego tekstu wokół punktu odniesienia; standardowo 0;

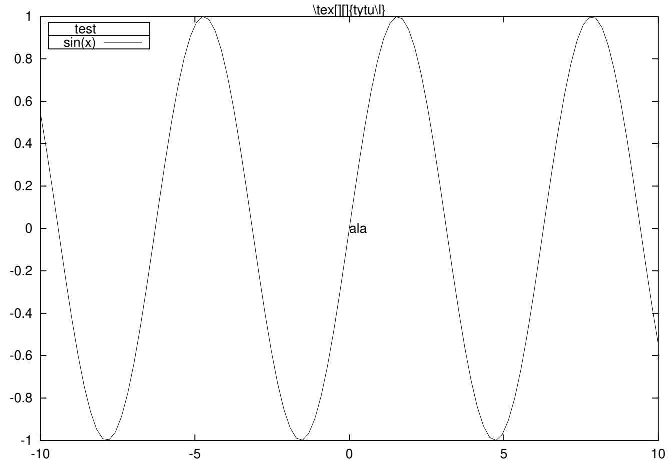
replacement wstawiany tekst (lub ogólniej obiekt) LaTeXowy.

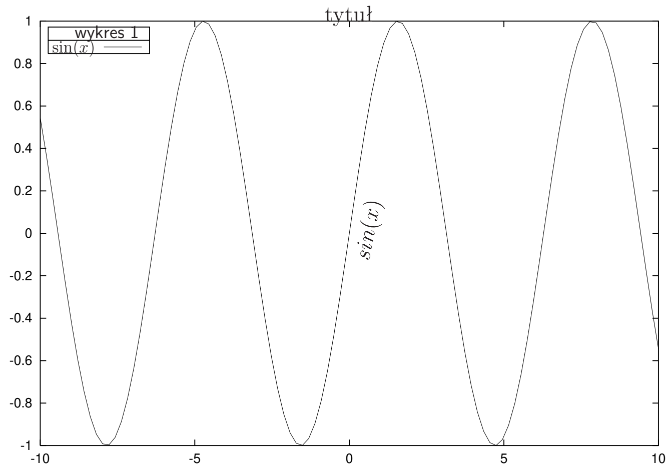
Jeżeli chcemy, aby pewne elementy tekstu w pliku graficznym były interpretowane zgodnie z regułami LaTeXa muszą mieć one postać:
`\tex`[*pos*] [*scale*] [*rot*]{*LaTeX text*}

Znaczenie parametrów jest identyczne jak parametrów polecenia `psfrag`.

Poniższy przykład obrazuje niektóre możliwości pakietu.

```
\includegraphics[width=.6\textwidth]{ala}
%
\psfrag{ala}[t][t][1][79]{$sin(x)$}
\psfrag{test}[l][l][.8]{{\sf wykres 1}}
\psfrag{sin\(x\)}[r][r][.75]{$\sin(x)$}
%
\psfragscanon
\resizebox{.6\textwidth}{!}{\includegraphics{ala}}
\psfragscanoff
```





9 Inne sposoby przygotowywania rysunków

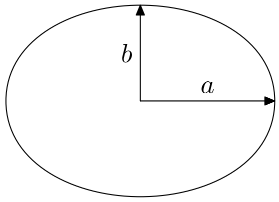
9.1 Metapost

METAPOST to system o strukturze podobnej do systemu METAFONT: posiada on rozbudowany zestaw poleceń i kompilator zamieniający je – w złożonym nierzadko procesie – w plik EPS. Nie opisujemy tu wszystkich możliwości systemu odsyłając do lektury dokumentacji [**?**]

Do przygotowywania rysunków można również użyć programu METAFONT. Jednak obiekty, które on tworzy są znakami, które muszą być zamieniane do postaci mapy bitowej osobno dla każdej rozdzielczości. METAPOST dający na wyjściu dosyć prosty plik EPS jest pod tym względem znacznie wygodniejszy.

Poniżej dwa przykłady:

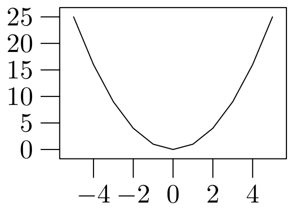
```
beginfig(1);
a=.7in; b=0.5in;
z0=(0,0); z1=(a,0); z2=(0,b);
z0=.5[z1,z3]=.5[z2,z4];
draw z1..z2..z3..z4..cycle;
drawarrow z0..z1;
drawarrow z0..z2;
label.top(btex $a$ etex, .5[z0,z1]);
label.lft(btex $b$ etex, .5[z0,z2]);
endfig;
end
```



Przykład ten można objaśnić następująco: `z0` *to współrzędne środka elipsy o półosiach* `a` *i* `b`. `z1` *i* `z2` *to dwa wierzchołki elipsy. współrzędne punktów* `z3` *i* `z4` *dobierane są tak, aby* `z0` *leżał na środku odcinków* `[z1,z3]` *i* `[z2,z4]`*. Polecenie* `draw` *nakazuje połączyć wszystkie punkty linią zamkniętą (*`cycle`*).* `drawarrow` *rysuje strzałki,* `label` *wstawia napisy…*

W ten sposób można tworzyć całkiem skomplikowane rysunki. Można też tworzyć specjalne makra ułatwiające tworzenie powtarzalnych rysunków. Oto najprostszy przykład. Plik `dane.dat` zawiera dwie kolumny danych – współrzędne punktów pewnego wykresu:

```
input graph;
beginfig(1);
 draw begingraph(3cm,2cm);
 gdraw "dane.dat";
 endgraph;
endfig;
end
```



9.2 PSTricks

PSTricks to bardzo obszerny zestaw makr służących do rysowania z wykorzystaniem możliwości udostępnianych przez język PostScript. Makra mogą być wykorzystywane zarówno w LaTeX 2_ε jak i w TeXu.

Każde z makr generuje pewien zestaw poleceń w języku PostScript, które są włączane jako obiekty **special** do pliku DVI. Nie będą one zazwyczaj interpretowane (i widoczne) gdy używamy „zwykłej" przeglądarki DVI. Jednak po przetworzeniu DVI do PS, na przykład za pomocą programu **dvips**, uzyskujemy żądane rezultaty.

Pakiet PSTricks powstał znacznie wcześniej niż LaTeX 2_ε i z tego powodu wśród jego poleceń znaleźć można też funkcje dublujące się z tymi oferowanymi przez pakiety **graphics**, **graphicx**, **color**.

Główną chyba wadą systemu PSTricks jest to, że LaTeX nic nnie wie na temat wielkości generowanego obiektu. Panować nad tym musi autor zamykając cały rysunek w otoczeniu `picture` lub `pspicture`.

10 Rysunki „oblane” tekstem

Czasami zachodzi konieczność (zwłaszcza, gdy rysunek jest niewielki) „oblania” go tekstem. Operacja ta nie zawsze przynosi dobre efekty zwłaszcza, gdy mamy za mało tekstu do „oblania” rysunku. Inne niż tekst „obiekty” nie bardzo się do tego nadają.

Podstawowy problem związany z takimi rysunkami to odpowiednie „przyczepienie” rysunku do fragmentu tekstu. Generalnie na sprawę patrząc bardzo często drobne zmiany w tekście, a co za tym idzie w jego rozłożeniu na stronach potrafią mieć kolosalny wpływ na układ rysunków. Tu sytuacja jest jeszcze gorsza – na ogół musimy oblewany rysunek przyczepić do jakiegoś tekstu. Gdy tekst się „rozjedzie” efekty mogą być katastrofalne...

Najprostszym rozwiązanie tego problemu jest zamknięcie obrazka i tekstu w osobnych środowiskach „minipage” o odpowiedniej szerokości i umieszczenie ich obok siebie.

Wadą takiego rozwiązania, jest to, że jakakolwiek zmiana w ministronie z tekstem może popsuć cały efekt wizualny.

Inne możliwości daje pakiet o nazwie `wrapfig`. Udostępnia on dwa środowiska o nazwach `wrapfigure` i `wraptable` pozwalające osiągnąć efekty, które (czasami) spełnią nasze wymagania...

Użycie ich jest następujące:

`\begin{wrapfigure}[`*nl*`]{`*placement*`}[`*overhang*`]{`*width*`}`
figure
`\end{wrapfigure}`

Znaczenie poszczególnych parametrów jest następujące:

nl Nieobowiązkowy parametr, mówiący ile linii tekstu powinno być „krótszych” (w normalnych warunkach wartość ta zostanie wyznaczona automatycznie, czasami jednak zachodzi konieczność korekty).

placement Obligatoryjny parametr mówiący gdzie ma być umieszczony rysunek:

- **r** po prawej stronie,
- **l** po lewej stronie,
- **i** „wewnątrz” (przy druku dwustronnym – inaczej dla stron parzystych, inaczej dla nieparzystych),
- **o** „na zewnątrz”.

overhang Określa jak bardzo rysunek będzie „wystawał” na margines (normalnie nie będzie wystawał).

width Definiuje szerokość wstawianego rysunku.

figure Wstawiany rysunek.

Zamiast pakietu **wrapfig** użyć można również: **floatflt** lub **picins**.

11 Znaki wodne

Właściwie to wykracza poza zasadniczy temat (włączanie do tekstu ilustracji czy wykresów), ale przygotowywując jakiś tekst lub slajdy możemy zechcieć umieścić na każdej stronie logo...

Aby osiągnąć taki efekt musimy posłużyć się pewną „sztuczką”. Pakiet nazywa się `fancyhdr`. Pozwala on (i jest to jego zasadnicza funkcja) bardzo łatwo definiować i modyfikować wygląd nagłówków i stopek stron.

Sztuczka polegać będzie na tym, że jako jeden z nagłówków definiować będziemy znak graficzny, który umieszczany będzie na każdej stronie. Pamiętać przy tym należy, że nagłówek drukowany jest **przed** wydrukowaniem zawartości strony a stopka **po**. Zatem grafika nakładana w stopce może (o ile jest „nieprzeźroczysta”) przysłonić tekst.

Pamiętać trzeba o odpowiednim pozycjonowaniu obiektu który chcemy umieścić na stronie. Doknać tego można za pomocą polecenia `\put`.

Najprostsze rozwiązanie wyglądać może zatem tak:

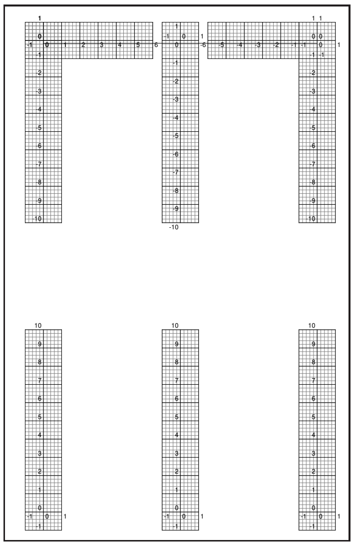
```
\fancyhead[L]%
 {
  \unitlength 1cm
  \begin{picture}(0,0)
   \put(0,-20){\includegraphics[width=\textwidth]{obiekt}}
  \end{picture}
 }
```

Niestety, nie jest to najlepsze rozwiązanie: podczas tworzenia każdej strony powyższe polecenia będą za każdym razem interpretowane. Lepszym rozwiązaniem może być utworzenie z obiektu graficznego „kontenera” (`\savebox`) i włączanie go do tekstu:

```
\newsavebox{\mygraphics}
\sbox{\mygraphics}{\includegraphics[width=\textwidth]{obiekt}}
...
\fancyhead[L]{\setlength{\unitlength}{1cm}
\begin{picture}(0,0)
\put(0,-20){\usebox{\mygraphics}}
\end{picture}}
```

Niestety, plik graficzny będzie włączony do wynikowego pliku PostScriptowego wielokrotnie.

Aby ułatwić sobie określenie pozycji w której wstawiany ma być obrazek można posłużyć się następującymi poleceniami, które produkują coś w rodzaju papieru milimetrowego:



```
\fancyhead[L]{\psgrid(0,0)(-1,-10)(1,  1)%
 \psgrid(0,0)(-1,-1)( 6,1)}
\fancyhead[R]{\psgrid(0,0)(-1,-10)(1,  1)%
 \psgrid(0,0)(-1,-1)(-6,1)}
\fancyhead[C]{\psgrid(0,0)(-1,  1)(1,-10)}
\fancyfoot[L]{\psgrid(0,0)(-1, -1)(1, 10)}
\fancyfoot[R]{\psgrid(0,0)(-1, -1)(1, 10)}
\fancyfoot[C]{\psgrid(0,0)(-1, -1)(1, 10)}
```

12 Drukowanie „czterostronne"

Pod pojęciem „drukowanie czterostronne" rozumiem robienie ładnych książeczek formatu A5. Osiągnięcie tego celu wymaga przeorientowania kartek, zmniejszenia ich oraz dwustronnego wydruku na drukarce.

Reorientację stron w pliku PS tak aby tworzyły książeczkę (albo jedną „zszywkę")wykonuje program **psbook** z pakietu **psutils**. Zmniejszenie oraz ułożenie stron po dwie na jednej kartce A4 wykonuje program **psnup** z tego pakietu.

Poniższa sekwencja poleceń realizuje zadanie (`plik.dvi` jest tym, co chcemy wydrukować):

```
dvips plik.dvi -o plik.ps
psbook plik.ps >plik1.ps
psnup -2 plik1.ps >plik2.ps
```

`plik2.ps` musimy wydrukować „dwustronnie".

Powyższą procedurę można uprościć (nie będą tworzone pliki pośrednie):

```
dvips -f1 plik.dvi | psbook |psnup -2 >plik2.ps
```

Najlepszym rozwiązaniem do druku dwustronnego jest zakup drukarki z odpowiednią przystawką (*duplex*). Niestety wyposażenie takie podraża koszt drukarki i znacznie komplikuje jej konstrukcję.

Jeżeli urządzenia takiego nie mamy – musimy sobie radzić inaczej: osobno drukować strony parzyste a osobno nieparzyste i zadbać o ręczne przeniesienie (i ewentualnie przesortowanie) kartek.

Odpowiednie funkcje realizuje program **psselect** (opcja `-o` wybiera z pliku strony nieparzyste a `-e` strony parzyste).

```
psselect -o plik2.ps > plik2_o.ps
psselect -e plik2.ps > plik2_e.ps
```

w zależności od sposobu prowadzenia papieru przez drukarkę może się okazać, że zmienić trzeba kolejność stron, wówczas piszemy:

```
psselect -e -r plik2.ps > plik2_e.ps
```

Program **psmandup** z pakietu **a2ps** realizuje powyższe dwa polecenia tworząc jeden plik PostScriptowy zawierający pomiędzy stronami nieparzystymi a parzystymi dodatkowe polecenie nakazujące pobieranie papieru z podajnika ręcznego. Użytkownikowi pozostaje jedynie przełożenie kartek z tacki odbiorczej drukarki do podajnika ręcznego:

```
psmandup plik2.ps -o plik2_.ps
```

Albo prościej:

```
dvips -f1 plik.dvi | psbook |psnup -2 |psmandup -o plik2_.ps
```

(Pamiętać trzeba tylko o jednym: o użyciu opcji `twoside` w linii `\documentclass[]{}`)

13 Lektury dodatkowe

A Źródła

Odsyłacze do większości wymienionych tu pakietów i programów znaleźć można w `http://sunsite.icm.edu.pl/pub/CTAN/help/Catalogue/catalogue.html`. Poniżej podajemy bardziej dokładne odsyłacze do miejsc gdzie programy można znaleźć (w miarę możliwości będą to miejsca w krajowe – co nie znaczy, że łatwo dostępne).

drajwery Adobe ...

gnuplot ...

Tkpaint ...

PageDraw, MayuraDraw ...

Xfig ...

hp2xx ...

graphics, graphicx ...

color ...

lscape ...

psfrag ...

METAPOST ...

PSTricks ...

wrapfig ...

picins ...

floatflt ...

fancyhdr ...

`epslatex.pdf` ...

`pltexmf.zip` ...

Polskie wzorce przenoszenia ...

B Spalszczanie LaTeX 2_ε

Właściwie nie ma to wielkiego związku z tematem zasadniczym, ale ogromna ilość stale powtarzanych pytań „w jaki sposób zmusić LaTeX 2_ε aby poprawnie przenosił polskie teksty" czy „jak uzyskać polskie litery" zmusza do opisania procedury krok po kroku.

Dziś jest już ona bardzo prosta – opracowane zostały znakomite zestawy makr, polskie fonty, polskie wzorce przenoszenia. LaTeX 2_ε jest „prawie gotowy"[11] do wprowadzania tekstów z użyciem polskich liter kodowanych zgodnie z zasadami systemu operacyjnego komputera którego używamy.

Spolszczyć LaTeX 2_ε możemy na dwa sposoby:

1. Używając pakietu **babel**.
2. Używając pakietu **platex**.

Nie mam zamiaru dyskutować na temat wyższości jednego pakietu nad drugim. Poniżej najbardziej podstawowe różnice:

- **platex** standardowo używa tak zwanych fontów **pl** (kodowanych w układzie OT4) – są to fonty **cm** wzbogacone jedynie o specyficzne dla języka polskiego znaki; **babel** używa fontów **ec** (kodowanie T1) – jest to zestaw fontów zawierający również znaki specyficzne również dla innych języków. Dodajmy, że korzystając z jednego i z drugiego systemu możemy korzystać z każdego z tych fontów – o ile są tylko zainstalowane.

- **platex** jako standardowej używa notacji „ciachowej" do zapisu polskich liter; **babel** standardowo używa jeszcze gorszej notacji: polskie litery wprowadzane są z prefiksem `"` (cudzysłów). Zatem `"r` w systemie **babel** to `/z` w systemie platex i `ż` w życiu codziennym. Na całe szczęście oba systemy pozwalają na wprowadzanie liter w sposób naturalny (to znaczy używając kodowania właściwego dla używanego systemu operacyjnego).

[11] Są jednak jeszcze bardzo subtelne problemy – właściwie niezauważalne dla „zwykłego" użytkownika.

- Pakiet **platex** zawiera kilka konstrukcji typowych dla polskich tradycji typograficznych, których nie ma w systemie **babel**.

- System **babel** potrafi być niekompatybilny z pewnymi konstrukcjami „klasycznego” systemu LaTeX 2ε.

- **babel** może być bardziej popularny na świecie, ale i tak poprawne korzystanie z polskich wzorców przenoszenia wymaga ich fizycznego zainstalowania w systemie i wygenerowania tak zwanego pliku formatu (o tym później).

Potrzebować będziemy:

- Polskich wzorców przenoszenia. Zwracam uwagę, że zawsze należy sprawdzić czy wzorce przenoszenia które znajdują się w systemie są poprawne. Te „dobre” (na dziś: 11 stycznia 1999 roku) mają oznaczenie:
  `% This is PLHYPH.TeX - the Polish hyphenation patterns`
  `% version 3.0a, Wednesday, May 17th, 1995`
  Spotkałem (starsze) dystrybucje systemy TeX w których była znacznie starsza wersja tego pliku.

- Plik `pltexmf.zip` (zawiera on praktycznie wszystko co jest potrzebne do spolszczenia LaTeXa). Jeżeli, oczywiście, chcemy używać **platexa**.

- Gdy korzystamy z emTeXa – przydać się może plik `polski7.zip`[12]

B.1 Babel

Sytuacja w zasadzie jest dosyć prosta. (Zakładam, że pakiet **babel** jest już zainstalowany.) Kopiujemy plik `plhyph.tex` tam gdzie znajdują się inne wzorce przenoszenia. W MikTeXu będzie to kartoteka `\texmf\tex\generic\hyphen`. Następnie znajdujemy plik `language.dat` (w MikTeXu: `\texmf\tex\generic\hyphen\local\language.dat`) i dopisujemy w nim linię:
`polish plhyph.tex`

Ostatnim etapem jest wygenerowanie nowego „formatu” (patrz punkt na stronie czyli wydanie polecenia:
`initex latex.ltx`
i skopiowanie powstałego pliku do właściwej kartoteki.

5 4 23 Piszemy po polsku używając następującej konstrukcji:

```
\documentclass[]{}
\usepackage[polish]{babel}
\usepackage[T1]{fontenc}
\usepackage[latin2]{inputenc}
\begin{document}
...
\end{document}
```

[12]Lub, być może, nowsza jego wersja.

B.2 PLaTeX

Pakiet przygotowany przez Mariusza Olko i Macieja Wolińskiego jest następcą, bardzo w swoim czasie popularnego, pakietu LaMeX. O ile tamten działał w „środowisku” $\LaTeX$2.09 – PLaTeX funkcjonuje w środowisku $\LaTeX\,2_\varepsilon$. Posiada też tryb 99 % zgodności z systemem LaMeX.

Instalacja jest stosunkowo prosta. Najlepiej zdobyć plik `pltexmf.zip` i rozpakować go we właściwym miejscu (to jest w tej kartotece, gdzie znajduje się (pod)kartoteka `\texmf\`). Wszystkie potrzebne pliki trafią automatycznie na swoje miejsca. **Polecam ten sposób postępowania!** Przy okazji instalujemy bardzo wiele innych użytecznych podczas pisania po polsku plików.

Jeżeli z jakichś powodów nie wybieramy tej drogi – możemy zainstalować sam pakiet PLaTeX „ręcznie”. Potrzebować będziemy jedynie pliki `platex-1.01.zip`, `pl-mf.zip`. Jeżeli dodatkowo chcemy ułatwić sobie życie – możemy wziąć jeszcze plik `pl-tfm.zip`.

1. Plik `pl-mf.zip` zawierający wzorce polskich liter rozpakowywujemy w kartotece `\texmf\fonts\source\public`. Utworzona zostanie podkartoteka `pl` zawierająca wszystkie pliki.

2. Plik `pl-tfm.zip` rozpakowywujemy w kartotece `\texmf\fonts\tfm\public\pl`.[13] Ten krok możemy pominąć, wówczas pliki TFM zostaną automatycznie wygenerowane w miarę potrzeb.

   Możemy zrezygnować z punktów 1 i 2 jeżeli nie zamierzamy korzystać z fontów **pl**!

3. Plik `platex-1.01.zip` rozpakowywujemy w jakiejś kartotece roboczej nie znajdującej się w strukturze `texmf`.

4. Po rozpakowaniu wykonujemy polecenie:
   `tex platex.ins`
   Odpowiedzieć musimy na kilka pytań:
   `*** Czy chcesz wygenerować pliki?`
   Powinniśmy odpowiedzieć `t`(ak)!

   Na wszystkie pytania typu:
   ```
   File cp1250.def already exists somewhere on the system.
   Overwrite it if necessary? [y/n]
   ```
   odpowiadamy twierdząco. Na tym etapie żadne istniejące już w systemie pliki nie będą kasowane!

   Kolejne pytanie dotyczy używanych fontów:
   `*** Czy na Twoim komputerze sa zainstalowane czcionki ‘pl’?`
   Jeżeli wykonaliśmy punkty 1 i 2 powyższej procedury – odpowiadamy twierdząco.

   I to już koniec! Kopiujemy wszystkie pliki o rozszerzeniach .STY, .DEF, .FD oraz pliki `hyphen.cfg` i `plhyph.tex` na właściwe miejsce, czyli do kartoteki `\texmf\tex\latex\platex\` (którą tworzymy w miarę potrzeby).

[13] Jeżeli kartoteka nie istnieje – powinniśmy ją utworzyć.

Jeżeli korzystaliśmy z „gotowca” – po rozpakowaniu go we właściwym miejscu od razu przechodzimy do do punktu 4, ale musimy zmienić kartotekę roboczą na `\texmf\tex\latex\platex\`.

Teraz musimy podjąć pewną decyzję – jeżeli na naszym komputerze jest zainstalowany już pakiet babel powinniśmy przekopiować plik `plhyph.tex` we „właściwe miejsce” (w przypadku pakietu MikTeX będzie to: `\texmf\tex\generic\hyphen\`) i zmodyfikować plik `\texmf\tex\generic\hyphen\local\language.dat`. Jeżeli nie mamy zamiaru korzystać z pakietu babel a jest on zainstalowany na naszym komputerze – być może powinniśmy go usunąć...

Przechodzimy teraz do generacji formatu. Procedura jest zasadniczo identyczna do tej opisanej na stronach: 22 i 25. Jeżeli nie mamy zamiaru używać pakietu babel – generację pliku formatu wykonujemy w kartotece `/texmf/tex/latex/platex` w przeciwnym razie w jakiejś innej kartotece.

Jeżeli korzystamy z pakietu emTeX sprawa jest nieco bardziej skomplikowana. Struktura katalogów systemu jest zupełnie inna i w związku z tym nie będziemy mogli skorzystać z „gotowca” **pltexmf**. Pozostaje zatem metoda opisana w tym punkcie – po dokładnym zapoznaniu się ze strukturą kartotek systemu emTeX. Znacznym ułatwieniem będzie skorzystanie z procedur pomocniczych zawartych w pakiecie `polski7.zip`.

C Instalacja nowszej wersji LaTeX 2$_\varepsilon$

Dwa razy do roku: w czerwcu i w grudniu (albo, czasami, nieco później) udostępniana jest nowa (poprawiona czy może udoskonalona) wersja systemu LaTeX 2$_\varepsilon$. W pewnym momencie pojawi się LaTeX3, docelowy, doskonalszy system składu.

Można ignorować pojawiające się nowsze wersje, jednak czasami wprowadzają one nowe funkcje (lub likwidują stare błędy) i wówczas warto zainstalować nowszą wersję...

Aby łatwo i bezboleśnie unowocześniać swoje oprogramowanie warto utrzymywać porządek w kartotekach gdzie przechowywane są różne pakiety. Podstawowe składniki systemu LaTeX 2$_\varepsilon$ znajdują się w kartotece `/texmf/tex/latex/base/`. Równocześnie unowocześniany jest też zestaw makr o nazwie **tools** znajdujący się w kartotece `/texmf/tex/latex/tools/`. Pamiętajmy, aby nie dodawać tam żadnych innych plików.

Unowocześnianie prowadzimy w kilku krokach:

1. Sprowadzamy nowe wersje makr LaTeX 2$_\varepsilon$ i tools.

2. Kopiujemy je do jakichś roboczych kartotek (dwu różnych).

3. Zapoznajemy się z plikami `ltnews`*nn*`.tex` i `changes.txt`, `00readme.txt`, `install.txt` (inne pliki o tym rozszerzeniu też warto przestudiować!)

4. W kartotece gdzie skopiowaliśmy bazowe pliki systemu LaTeX wydajemy polecenie:
   `initex unpack.ins`

jest to procedura powodująca „rozpakowanie” wszystkich niezbędnych pakietów systemu. Zajmuje to od kilku do nawet kilkudziesięciu minut (na bardzo wolnym komputerze).

5. Teraz musimy wyprodukować tak zwany „LaTeX format”. Otóż system TeX jest specjalnym językiem składu dokumentów. LaTeX to zestaw makropoleceń wykorzystujących ten język. Teoretycznie jest możliwe podczas każdego przetwarzania dokumentu latexowego czytać i przetwarzać te makropolecenia. Znacznie jednak sensowniej jest przetworzyć je raz i zapisać efekt na dysku w postaci pliku (to jest właśnie ów format) będącego odwzorowanie zawartości pewnego fragmentu pamięci operacyjnej programu. Później wystarczy załadować w odpowiednie miejsce programu ten plik i można szybko zająć się kompilacją pliku użytkownika.

   Polecenie:
   `initex latex.ltx`
   powoduje wygenerowanie pliku `latex.fmt` będącego właśnie nowym obrazem wszystkich makr używanych przez LaTeX.

6. Plik `latex.fmt` kopiujemy na jego właściwe miejsce.[14] Radzę wcześniej zachować „starą” wersję pliku (na przykład zmieniając jej nazwę).

7. Następnie zachowujemy gdzieś zawartość katalogu `\texmf\tex\latex\base` a w to miejsce kopiujemy wszystkie pliki, które powstały w procesie rozpakowywania pakietu LaTeX to znaczy:

   - `latexbug.tex`, `testpage.tex`, `lablst.tex`, `idx.tex`, `nfssfont.tex`, `small2e.tex`, `sample2e.tex` i `docstrip.tex`.
   - `*.cls` – definicje klas,
   - `*.clo` – pliki z definicjami opcji klas,
   - `*.sty` – pakiety,
   - `*.fd` – pliki definicji fontów,
   - `*.def` – różne definicje czytane podczas przetwarzania dokumentów,
   - `*.cfg` – pliki konfigfuracyjne: tylko dla TeXpertów.

8. Pliki o rozszerzeniu `*.ist` przenosimy do kartoteki czytanej przez program **MakeIndex**

9. W kartotece gdzie mamy wszystkie pliki źródłowe pakietu **tools** rozpakowywujemy go poleceniem:
   `initex tools.ins`
   Wszystkie powstałe pliki kopiujemy do (opróżnionej i zabezpieczonej wcześniej) kartoteki: `\texmf\tex\latex\tools`.

[14] Zależy ono od implementacji której używamy i warto się wcześniej zapoznać z jej strukturą katalogów. W przypadku współczesnych implementacji **MikTeX**, **teTeX**, **Web2c** będzie to zazwyczaj: `\texmf\`*implementation*`\fmt\`

(W powyższej procedurze najwygodniej jest kopiować wszystkie pliki, które właśnie powstały – mają bieżącą datę modyfikacji.)

Dodam jeszcze, że większość pakietów użytkowych systemu LaTeX 2_ε rozpowszechniana jest w postaci plików o rozszerzeniu `.dtx`. Zawierają one na ogół bardzo dobrze skomentowany kod źródłowy, zapisany w takiej postaci aby po odpowiednim przekształceniu uzyskać albo sam pakiet (pliki `.sty` i towarzyszące) albo jego dokumentację: nie tylko techniczną ale również użytkową. Dokumentację uzyskujemy po kompilacji pliku poleceniem:

```
latex plik.dtx
```