# A LaTeX alapjai

## Alapok

A LaTeX (ejtsd: "lateh") egy szövegszerkesztésre, szedésre és megjelenítésre használható programrendszer, a tudományos publikációk elsődleges szerkesztési felülete. A jegyzet célja megismertetni a LaTeX alapjait és legfontosabb technikáit, trükkjeit. Csillagászati folyóiratok ma már kizárólag LaTeX-ben szedett kéziratokat fogadnak el, így célszerű mielőbb megbarátkozni a grafikus felületű szövegszerkesztőkhöz szokott felhasználók számára első pillanatra nehézkesnek és furcsának tűnő szövegszedési filozófiával.

A szerkesztett szöveg egy egyszerű ASCII textfájl. Erre a továbbiakra, mint LaTeX forráskódra hivatkozok. Mivel mindenféle beállításra, szövegformázásra valamilyen LaTeX paranccsal hivatkozunk, ezért a forráskód tetszőleges egyszerű szövegszerkesztőkkel írható (pl. `joe`, `pico`, `vi`, `textedit`, `mcedit` a laborgépeken `gedit`, `nedit`, stb.). A forráskód tartalmazza a megjelenítendő szöveget, valamint az összes, formázásra vonatkozó utasítást. Hogy lesz ebből nyomtatásra kész anyag? Miután befejeztük a forráskód írását (pl. `file.tex`), adjuk ki a

```
$ latex file.tex
```

parancsot! Ennek hatására a forráskód lefordul (mint pl. egy C-, vagy Pascal-program), eredményül pedig létrejön egy `file.dvi` nevű fájl. A dvi kiterjesztés a "device independent" rövidítése, ez már egy olyan bináris fájl, amit dvi-néző programmal megnézhetünk, benne a szöveget megformázva látjuk (esetünkben az `xdvi` használható). Ez még azonban nem nyomtatható formátum, pl. az ábrák nincsenek beágyazva a dvi fájlba. A nyomtatáshoz a dvi fájlt át kell fordítani a nyomtatók által „értett" postscript változatba. Ehhez a

```
$ dvips -o file.ps file.dvi
```

parancsot kell lefuttatni. Végeredményül megszületik a `file.ps`, amit a

```
$ gv file.ps &
```

paranccsal nézhetünk meg, az

```
$ lpr file.ps
```

paranccsal pedig kinyomtathatunk (előtte mindig konzultáljunk a gyakorlatvezetővel!).

## A forráskód szerkezete

A LaTeX forráskód szerkezetét néhány szigorú szabály szabja meg. A fájl elején a preambulum áll. Ebben adjuk meg a dokumentum külalakját meghatározó parancsokat. Például:

```
\documentclass[11pt,magyar]{article}
\usepackage[magyar]{babel}
\usepackage[T1]{fontenc}
\usepackage[latin2]{inputenc}
\usepackage{geometry}
\geometry{verbose,a4paper,tmargin=3cm,
bmargin=3cm,lmargin=2.5cm,rmargin=2cm}
\usepackage[dvips]{graphicx}
\usepackage{epsfig}
```

A legelső sor a legfontosabb: 12 pontos betű, magyar szöveg, és mindez az article, előre definiált dokumentum-osztályban (mint az éppen olvasott dokumentum is). A további sorok opcionálisak, elmaradásuk esetén a LaTeX article dokumentumokra alapértelmezett beállításai lesznek érvényesek. Az összes csillagászati szakfolyóiratnak megvan a saját dokumentum-osztálya, előre definiált speciális parancsokkal, vagy átdefiniált értelmű „hagyományos" parancsokkal. Így a LaTeX használatával nyomdakész formátumú kéziratokat állíthatunk elő, amelyek a képernyőnkön már úgy néznek ki a szerkesztés közben, ahogy az később a nyomtatásban is kijön. A babel csomag egy adott nyelvre vonatkozó elválasztási szabályokat illetve a megszokott ábra, táblázat illetve tartalomjegyzék és fejezettagolás neveit tartalmazza. Ha angol nyelvű dokumentumot készítünk, akkor természetesen ezt a csomagot nem kell betölteni. A `titan`-on a magyar elválasztási csomag bele van forditva a LaTeXbe, de ez nem alapkonfiguráció, így ha egy olyan gépen fordítjuk a forrásunkat melynél ebben nem vagyunk biztosak, akkor érdemes leellenőrizni. Erről a fordítás folyamán meggyőződhetünk, ugyanis a fordító kiírja, hogy mely `hyphen` csomagok vannak betöltve:

```
gaspara@titan:~/labor$ latex linux-1.tex | more
This is e-TeX, Version 3.14159-2.1 (Web2C 7.4.5)
entering extended mode
(./linux-1.tex
LaTeX2e <2001/06/01>
Babel <v3.7h> and hyphenation patterns for american, french,
german, ngerman, magyar, nohyphenation, basque, czech,
icelandic, slovak, slovene, loaded.
```

A `T1`-gyel az újfajta TeX szövegkódolást használjuk, a régebbi `OT1` helyett. Emellett a European Codinggal, amelynek segítségével az ékezetes betűket tartalmazó szavak elvállasztását segítjük elő. A `[latin2]{inputenc}`-kel azt adjuk meg, hogy az ISO 8859-2 -nek megfelelő latin karaktercsomagot akarjuk használni. Ez tartalmazza a magyar ékezetes betűket. A `geometry` csomag a lap méretét, illetve a lap egyéb geometriai tulajdonságait kezeli. A `graphicx` csomag a dokumetumba beépítendő képeket kezeli (ugyanúgy, ahogy a `epsfig is`). Ezek mellet sok más beállítást itt tehetünk meg, például, ha nem kívánjuk, hogy a dokumentumunknak legyen oldalszámozása, akkor a `\pagestyle{empty}` parancsot írjuk be. Ez a preambulum a legújabn LaTeX standardoknak felel meg, régebbi rendszereknél előfordulhat, hogy némely csomagnak más a neve, ilyenkor ezt érdemes ellenőrizni.

A preambulum után következik maga a dokumentum: minden, ami a

```
\begin{document}
```

és az

```
\end{document}
```

között szerepel. Ez lehet sima szöveg (amit fejezetekre oszthatunk), képletek, táblázatok, ábrák, tetszőleges sorrendben és elhelyezésben. Fontos: a forráskódon belül nincs sok szerepe a szöveg kinézetének. Amíg a szöveget sorfolytonosan gépeljük, az egyetlen bekezdésbe fog kerülni, új bekezdést egy üres sor beiktatásával kezdünk. A szóközök számának nincs szerepe.

## Speciális karakterek

A forráskódban a billentyűzetről bevitt összes karakter szerepelhet, amik közül 10 különleges jelentéssel bír:

\ { } % $ ~ _ ^ & #

**A \\-jel: parancskezdés**. Jelentését a preambulum példája is sugallta, ezzel kezdődik az összes LaTeX parancs. Szintén ezzel kezdődnek a kétjeles parancsok, amelyekben a \ után egy nem-alfabetikus karakter áll. Pl. az `\'a` eredménye az á betű. Sortörésre a \\\\ parancs szolgál. Ha esetleg több helyet szeretnénk valahol kihagyni függőlegesen, akkor azt a `\vskip{15mm}` paranccsal tehetjük meg. Természetesen ha nem 15 mm-t szeretnénk kihagyni, akkor többet írunk. Ugyanígy működik a `\hskip{15mm}` parancs is, csak ez horizontális, azaz vízszintes helyet hagy ki.
**A %-jel: megjegyzések**. A forráskódba megjeleníteni nem kívánt megjegyzéseket („kommenteket") tehetünk. A % jel után, az adott sor végéig szereplő szöveg csak a forráskódban jelenik meg, a LaTeX fordító figyelmen kívül hagyja.
**A { és } jelek: blokkok létrehozása**. A kapcsos zárójeleken belül szereplő részeket a fordító egységesen kezelt alegységként kezeli a forráskódon belül. Tipikus felhasználásuk a parancsok argumentumainak megadása. Emellett a szöveg formázásánál is szerephez jutnak a blokkok, pl. a

```
Ez egy mondat, közepén {\bf félkövér betűkkel} szedett szóval.
```

eredménye:

Ez egy mondat, közepén **félkövér betűkkel** szedett szóval.

## Ékezetek, speciális betűk, betűk megjelenése

A LaTeX egy csomó, előre definiált speciális karaktert ismer, részletesen l. ezen leírás mellékleteit. A számunkra legfontosabb magyar ékezetes betűket két módon vihetjük be:

1. Repülő ékezetek (l. melléklet). Például: á = \' a; é = \' e; ő = \H o
2. A forráskód kódkészletének helyes megadásával. Ekkor a LaTeX tudja értelmezni a forráskódban megadott ékezetes betűket. Ezt a preambulumban tehetjük meg a már ismert módon:

```
\usepackage[latin2]{inputenc}
```

A speciális karaktereket is természetesen megjeleníthetjük eredeti formájukban, ennek módozatát szintén a mellékletben találjuk meg.
Ha már ismerjük a legkülönbözőbb betűk megadását, akkor foglalkozhatunk a megjelenéssel is. A betűknek általában van alakjuk (álló, *döntött*, *kurzív*, KISKAPITÁLIS), vastagságuk (normál, **félkövér**), betűcsaládjuk (pl. antikva, groteszk, `írógép`) és méretük:

legkisebb - `\tiny`,
a kitevő mérete - `\scriptsize`,
a lábjegyzet mérete - `\footnotesize`,
kis méret - `\small`,
normál méret - `\normalsize`,
nagy - `\large`,
nagyobb - `\Large`,
még nagyobb - `\LARGE`,
legnagyobb - `\huge`,
legeslegnagyobb - `\Huge`

Ha sokszor kell váltanunk különböző betűméretek között és nem akarunk túl sokat gépelni, éljünk a(z egyébként is igen hasznos) parancsdefiníció lehetőségével (ezt akár a preambulumban, akár szöveg közben is megtehetjük):

```
\newcommand{\up}{\footnotesize}
```

Ennek hatására az `\up` új paranccsal hivatkozhatunk a sokkal hosszabb `\footnotesize` parancsra, ami a normál betűméretről a lábjegyzet méretére kapcsolja a betűket.

# Egy kis matematika

A matematikai formulák betűit általában más betűtípussal írjuk. A személyi számítógépeken elterjedt szövegszerkesztőkben (pl. WinWord, StarOffice) általában külön képletszerkesztő segíti a munkát. A LaTeX-ben a képletek szedéséhez az ún. matematikai módba kell kapcsolni. Ez kétféle lehet, annak megfelelően, hogy vannak szövegközi képletek, pl. $e^{i\pi}+1=0$, és kiemelt képletek:

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}(t)).$$

A matematika módba történő átkapcsolás: szövegközi képlet – \( és \) között; kiemelt képlet – \[ és \] között. Ezekkel ekvivalens a `\begin{math} képlet \end{math}`, a `\begin{displaymath} képlet \end{displaymath}`, illetve a `\begin{equation} képlet \end{equation}` környezetek. A LaTeX ismeri a szövegközi képlet sima TeX-beli megadását ($ képlet $), ugyanakkor a kiemelt képletek ($$ képlet $$) TeX-es megadását kerüljük, mert nem ad mindig kielégítő outputot (legalább is a szakirodalom szerint).

Tekintve, hogy a LaTeX kiindulási alapját, a TeX rendszerét Donald Knuth amerikai matematikus alkotta meg az 1970-es évek végén, nem csoda, hogy a matematikai módnak van a leggazdagabb parancskészlete. A görög betűk, gyakran használt héber betűk, matematikai ékezetek, relációjelek, műveleti jelek, függvénynevek, nyilak, határoló jelek és egyéb jelek több száz egyedi paranccsal érhetők el, áttekintésüket a mellékletben találjuk.

Az alapműveletek közül a $+$, $-$, $/$ jeleket a billentyűzetről visszük be. A szorzás jele vagy a $\cdot$ (`$\cdot$`), vagy a $\times$ (`$\times$`). A hatványozás műveletét és a felső indexet a ^ jellel, az alsó indexet a _ jellel adjuk meg. Ha nem egy, hanem több jel kerül alsó, vagy felső indexbe, akkor azokat kapcsos zárójelek közé kell zárni! Egymásba ágyazott indexsorozatokat is szerkeszthetünk, pl. az $a^{a^{a^{a^{n+1}}}}$ forráskódja:

```
$a^{a^{a^{a^{n+1}}}}$
```

A törtek kifejezésére a / jel mellett a `\frac` parancs használható, aminek két argumentuma van, a számláló és a nevező. Gyökvonást a `\sqrt` paranccsal végzünk, ennek egyetlen argumentuma a kifejezés, amiből gyököt vonunk. A leggyakrabban használt függvényneveket előre definiálták (pl. `\sin`), hogy jól látszó, eltérő betűtípussal jelenjenek meg a képletekben.

Külön művészet a zárójelek kezelése. A leggyakrabban használt zárójelek a billentyűzetről is bevihetők: (, ), [, ], |, míg a kapcsos zárójeleket a \{ és \} parancsokkal adhatjuk meg. Ahhoz, hogy tényleg zárójelként funkcionáljanak (pl. összetett szerkezetekben belülről kifelé haladva nőjön a méretük), szükséges még a `\left` és a `\right` parancs is. Egy példa a legvilágosabb magyarázat: az

$$\left(1+\left(1+\left(1+x\right)^2\right)^2\right)^2$$

forráskódja a

```
\[ \left(1+\left(1+\left(1+x \right)^2\right)^ 2\right)^2\]
```

karaktersorozat.
Előfordulhat, hogy szöveget kell beágyaznunk egy formulába (pl. magyar ékezetes betűs szó egy alsó index: $P_{\mbox{\scriptsize Föld}}$). Ilyenkor az `\mbox` parancs használható:

```
$ P_{\mbox{\scriptsize Föld}}$
```

Sajnos az `\mbox` nem figyeli a matematikai környezet aktuális betűméretét, ezért kellett lecsökkenteni a méretet (más módon is kikerülhető a probléma, de most az nem érdekes). A math mód a leggazdagabb parancskészlettel rendelkező környezet a LaTeX -en belül. Bővebben róla LaTeX könyvekben olvashatunk.

## Szövegformázás

A dokumentum betűkből áll. A betűk szavakká, a szavak mondatokká, a mondatok bekezdésekké állnak össze. A LaTeX a forráskód alapján találja ki az elemek kezdetét és végét, néhány egyszerű szabály alapján. Két szót szóközzel, tabulátorral, vagy sorvége jellel lehet elválasztani. Szóközből és tabulátorból tetszőleges számú lehet, sorvége karakterből csak egy. Két sorvége karakter (két enter-billentyű, azaz egy üres sor) választja el a bekezdéseket. A mondat végét írásjel jelzi, de ha a mondatot egy nagybetű zárja (pl. egy rövidítés miatt), akkor azt külön jelezni kell (ez már túlmutat aktuális igényeinken).

A szavak elválasztását a LaTeX automatikusan végzi. Ha netán valamelyik szó nem lett jól elválasztva, akkor magunk megadhatjuk a szótagok tagolását a \- használatával, például:

```
el\-vá\-lasz\-tás.
```

A szavak mellett a szöveg nagyobb szerkezeti egységeit is kiemelhetjük valamilyen stílusváltással. A bekezdéseket behúzással emeli ki a LaTeX automatikusan, ha ezt ki akarjuk kapcsolni, használjuk a `\noindent` parancsot. A szöveg sorait kiemelhetjük balra, jobbra, vagy középre zárással. Ezeket a

```
\begin{flushleft} szöveg \end{flushleft}
\begin{flushright} szöveg \end{flushright}
\begin{center} szöveg \end{center}
```

környezetekkel érthetjük el (tetszőleges számú bekezdésen keresztül). Hosszabb idézeteket a

```
\begin{quote} szöveg \end{quote}
```

környezettel emelhetünk ki.
Végezetül a listák készítéséről röviden. A LaTeX-ben három környezetben készíthetünk listákat:

```
\begin{itemize} felsorolás \end{itemize}
\begin{enumerate} sorszámozott lista \end{enumerate}
\begin{description} leíró lista \end{description}
```

A listák elemeit mindegyik környezetben az `\item` paranccsal kell kezdeni.

## Táblázatok

A LaTeX-ben tett első kirándulásunkat fejezzük be a táblázatok szerkesztésével. Egy táblázat készítése előtt csak annyit kell tudnunk, hogy hány oszlopa lesz, illetve hogy melyik oszlopot merre szeretnénk igazítani. A háromféle igazításhoz (bal, jobb, közép) egy-egy betű (l, r, c) tartozik, az ezekből alkotott karaktersorozat lesz a `tabular` környezet egyetlen argumentuma. Minden oszlophoz egy betű tartozik. Ezután megadjuk a táblázat összes sorát, az utolsó sort kivéve mindegyiket a \\ paranccsal zárjuk. Az oszlopok közé a & jelet tesszük. Pl. a

| | | |
|:---|---:|:---:|
| Első sor | 15.6 | 8.2 |
| Második sor | 23.1 | 0.12 |
| Harmadik sor | 0 | 12 |

forrása a következő:

```
\begin{tabular}{|l|rc|}
\hline
Első sor & 15.6 & 8.2\\
Második sor & 23.1 & 0.12\\
Harmadik sor & 0 & 12\\
\hline
\end{tabular}
```

A táblázatok szerkesztésénél rengeteg apró finomságot be lehet még állítani, aminek ismertetése azonban túlmutat jelen céljainkon.

## Szövegek tagolása

Ha hosszabb szöveget (dolgozatot, diplomamunkát) szerkesztünk, akkor ezt a szöveget valószínűleg szeretnénk bevezetésre, tartalomjegyzékre, fejezetekre, alfejezetekre, irodalomjegyzékre, köszönetnyílvánításra, függelékre és egyéb részekre tagolni. Ehhez tökéletes környezetet tud nyújtani a LaTeX , ugyanis a tartalomjegyzéket automatikusan tudja generálni, ha az egyes fejezeteket a megfelelő módon vezetjük be.
Ha fedőlapot szeretnénk, akkor azt a `\begin{titlepage}` és a `\end{titlepage}` parancsok közé kell, hogy írjuk. Ezt a lapot a LaTeX nem oldalszámozza.
Magát a tartalomjegyzéket a nagyon egyszerű `\{tableofcontents}` paranccsal hozhatjuk létre. Mivel a LaTeX fordítás közben érzékeli majd, hogy hol, melyik oldalnál vannak a fejezetek, ezt a tartalomjegyzéket maga generálja. Ha a babel magyar csomagja be van töltve, akkor „Tartalomjegyzék”-ként el is nevezi. Mivel csak utólag megy végig a szövegen, ezért ahhoz, hogy ez létrejöjjön, kétszer kell egymás után lefordítani a forráskódot.

```
$ latex szoveg.tex
This is TeX, Version 3.14159 (Web2C 7.4.5)
(./szoveg.tex
LaTeX2e <2001/06/01>
...
$ latex szoveg.tex
This is TeX, Version 3.14159 (Web2C 7.4.5)
(./szoveg.tex
LaTeX2e <2001/06/01>
...
$ dvips -o szoveg.ps szoveg.dvi
This is dvips(k) 5.92b Copyright 2002 Radical Eye Software
(www.radicaleye.com) ' TeX output 2005.01.28:1620' -> szoveg.ps
...
$
```

A „Bevezetés”-t általában nem szokták számozással a tartalomjegyzékben megjeleníteni, viszont ez is egy tagolási egység. Ha azt akarjuk, hogy szépen jelenjen meg, mint a többi egység, viszont számozás nélkül, akkor a következő trükköt játszuk meg:

```
\section*{}
\addcontentsline{toc}{section}{Bevezetés}
```

Mint látjuk a `\section{Ez egy fejezetcím}` paranccsal adhatjuk meg, hogy ez egy új fejezet innentől kezdve. A LaTeX így a kellő helyeket teszi a cím elé és után, sorszámozza a fejezeteket ...stb. A * -gal azt mondtuk neki, hogy ezt a `\section`-t ne tegye bele a tartalomjegyzékbe és ne is számozza. Viszont szeretnénk, ha számozás nélkül szerepelne a tartalomjegyzékben. Ezt tesszük meg a második sorban lévő parancssorral.

Ha új lapon szeretnénk kezdeni a következő fejezetet vagy csak úgy új lapon akarjuk folytatni az irományunkat, akkor azt `\newpage` paranccsal tehetjük meg. Fejezetnek alfejezetét pedig a `\subsubsection{alfejezetcím}` paranccsal hozhatunk létre.

A „köszönetnyílvánítást” és az „összefoglalást” általában ugyanúgy hozzuk létre, mint a „bevezetést”. Az „irodalomjegyzék”-et többféle módon létrehozhatjuk. Legegyszerűbben a következőképpen:

```
\addcontentsline{toc}{section}{Irodalomjegyzék}
\bibliography{tdk}
\begin{thebibliography}{99}
\bibitem[Barsony et.  al, 1997]{bars97} Barsony, M. et al.:
   1997, {\it Astrophysical Journal Supplement}, {\bf 112},109.
\bibitem[Bally et al., 1983]{bally83} Bally, J., Lada CJ.:
   1983, {\it Astrophysical Journal}, {\bf 265}, 824-847.
\bibitem[Balog et al., 2004]{balog04} Balog, Z. et al.:
   2004, {\it Astronomical Journal}, in press
...
\end{thebibliography}
```

A dolgozatok írása közben az ember folyamatosan bővítheti az irodalomjegyzéket, a lényeg, hogy szerzők szerint ABC sorrendben legyenek a referenciák. A dolgozatban többféleképpen lehet hivatkozni. Ha a szövegkörnyezetbe ezután beírjuk valahová azt például, hogy `\cite{bars97}`, akkor az a szövegben úgy fog megjelenni, hogy: (Barsony et. al, 1997). Ugyanígy majd látni fogjuk, hogy ábrákra és táblázatokra is lehet hivatkozni,

# Képek, ábrák beillesztése

LaTeX forrásba legkönnyebben `eps` (enchanced postscript) vagy szimpla `ps` (postscript) fajlt tudunk beilleszteni. Ezt sokféleképpen tehetjük meg:

```
\begin{figure}
\begin{center}
\psfig{figure=img/szines.eps,width=10cm}
\end{center}
\end{figure}
```

Ezzel egy képet vízszintesen középre helyezünk. Magát a képet a `tex` fájlhoz képesti elérési út szerint adjuk meg (ez esetben a kép az `img` mappán belül volt található). A képnek valamelyik paraméterét, így szélességét vagy magasságát tudjuk szabályozni. Általában a cikkekben, illetve dolgozatokban grafikonokat szoktunk ábrázolni, melynek így felirata szokott lenni. A grafikonokra hivatkozni is kell a szövegekben. Ezt a két feladatot is nagyon egyszerűen hajthatjuk végre:

```
\begin{figure}
\begin{center}
\psfig{figure=img/kep.eps,height=9cm}
\caption{Ide kerül a kép felirata}
\label{fig:hivatk}
\end{center}
\end{figure}
```

Ezzel a képünk az előzőekhez hasonlóan bekerül a szövegbe, azzal a különbséggel, hogy most a kép alatt meg fog jelenni egy felirat (pl. „22. ábra: Ide kerül a kép felirata"). A képek, táblazatok számozása automatikus. Az ábrára a `\ref{fig:hivatk}` kóddal hivatkozhatunk (természetesen logikusan azt a "label"-t megadva, amit használtunk a képnél). Meg lehet azt is csinalni, hogy a szöveg a képet körbevegye, illetve subképekre is lehet osztani egy fő képet, s így külön `a.)` illetve `b.)` ábrákat kapunk. Ezekre lehet külön is hivatkozni.Táblázatoknak feliratott illetve hivatkozást hasonlóan adhatunk, mint a képek esetében.

# LaTeX trükkök

## Képek - alképek

A LaTeX-ben meg lehet csinálni azt is, hogy egy kép nem csak egy darab ábrából, hanem alábrákból is áll. Ilyenkor az egyes képeket további a), b), . . . stb. feliratozással látja el és a hivatkozásoknál is így jelenik meg.

Ahhoz, hogy ez működjön, szükség van a „subfigure” csomag betöltésére.

```
\usepackage{subfigure}
```

Egy a) illetve b) alábrából álló ábracsoportot a következő forrással tudunk beilleszteni:

```
\begin{figure}[!h]
\begin{centering}
\subfigure[felirat 1]{
\label{hivatkozas1}
\begin{minipage}[t]{0.5\textwidth}
\centering\psfig{figure=img/abra1.eps,height=5cm}
\end{minipage}}%
\subfigure[felirat 2]{
\label{hivatkozas2}
\begin{minipage}[t]{0.5\textwidth}
\centering\psfig{figure=img/abra2.eps,height=5cm}
\end{minipage}}%
\caption{Az egesz felirata}
\label{azegeszrehivatkozas}
\end{centering}
\end{figure}
```

Ezt a forráskódot szépen lehet paraméterezni a beillesztendő képek méretének megfelelően a `xx\textwidth` állításával. Ez adja meg, hogy a lapmérethez képest mekkora „minipage”-„minilap”-ra szorítsa be az ábrát.

### Képek-körbeszedve

A MS Word használók biztosan megszokták, hogy a beillesztett képeket a szöveggel körbe tudják venni, ezzel is jobban kihasználva a lapok üres területeit. Jó hír, hogy nincs olyan probléma, amit a LaTeX ne tudna „lekezelni”, legfeljebb kicsit macerásabb, de a végeredményt tekintve, érdemes az a kevéske szenvedés. A képek szöveggel való körbevételét a `wrapfigure` segítségével érhetjük el. Ezt a `subfigure` csomaghoz hasonlóan a preambulumban be kell tölteni.

```
\usepackage{wrapfig}
```

Egy képet pedig ehhez hasonló forráskóddal tudunk beilleszteni a szövegbe:

```
\begin{wrapfigure}[20]{l}
[5pt]{10cm}
\hspace{0mm}\psfig{figure=img/kep22.eps,height=7cm}
\caption{Ábrafelirat}
\label{hivatkozas22}
\end{wrapfigure}
A kép utáni szöveget kezdjük el így ide írni sorfolytonosan ...
```

A `{wrapfigure}` utáni `[]`-ben szereplő szám adja meg, hogy az kép utáni hány sort tegye a kép mellé. Azaz teljes mértékben tudjuk szabályozni, hogy hova és mennyit tegyen a LaTeX . Az `{l}`-lel azt adjuk meg, hogy a kép az oldal bal oldalán legyen. Az `[5pt]`-vel azt adjuk meg, hogy a kép mellett mekkora helyet hagyjon ki, illetve a `{10cm}`-rel a szélességét a „doboznak”, amit a képnek hagyott. A többi magától érthető.

## A verbatim mód

Gyakran előfordul az az eset, amikor a dolgozatunkba, írásunkba például szeretnénk beültetni valamelyik programunk forráskódját. Ekkor a legalkalmasabb választás az úgynevezett „verbatim” mód. A

```
\begin{verbatim}
\end {verbatim}
```

közé írt szöveget a LaTeX nem formázza, olyan formátumban helyezi be a dokumentumba ahogyan a LaTeX forráskódba beírtuk. Ez kényelmes, hisz a szóközök megmaradnak, amik egy program forráskódjának az átláthatóságát növelik. Természetesen a speciális karaktereket át kell írni a „verbatim” módban is.

## Jó tanácsok

Az átlagos kezdő Unix felhasználó zavarban szokott lenni a fájlrendszer illetve a kezelőfelület láttán. Az egész `/home` könyvtár átláthatóságát nagyban növeli, és későbbi fájlkeresésekben nagy hasznunkra lehet, ha már a kezdetektől fogva rendszerezzük fájljainkat. Egy LaTeX dokumentum elkészítésében így ajánlatos a következőek szerint eljárni:

1. Magát a dokumentumot egy saját mappán belül kezdjük el írni. Valószínüleg nem ez lesz az egyedüli dokumentumunk, mely az adott témakörben készül, így nem feltétlen előnyös rögton a `/home`-unkba se rakni. Azaz például egy `/home/gaspara/cikkek/halmazok/ngc2126` mappában kezdjünk el dolgozni.
2. A mappán belül hozzunk létre külön mappát az ábráknak. Ha netán itt vannak gnuplot scriptek is, meg adattömbök is a scriptekhez, azokhoz is érdemes mindent rendszerezni.
3. A dokumentum forráskódja az adott mappában van, és az egyes képek elérési útvonala így például `figure=img/kep.eps`, ha az `img` mappában van.
4. Érdemes csinálni egy „Makefile”-t, mellyel a fordítást végezzük. Sokkal egyszerűbb (főleg ha sokszor fordítjuk), mellesleg kevés extra fájlt hagyunk magunk után, ezzel is csökkentve a szemét számát a gépünkön.

## A Makefile

A „Makefile” egy egyszerű ASCII text fájl, melyet abban a könyvtárban helyezünk el, ahol a fordítandó program forráskódunk, esetünkben a `tex` fájl, található. A fájlt `Makefile`-nak vagy `MAKEFILE`-nak kell elnevezni (gyakoribb a `Makefile`). Ha a fordítási műveleteket tartalmazó fájlt helyesen írtuk meg, akkor ezek után ha a promptban kiadjuk a

```
$ make
```

parancsot, akkor lefordítja a LaTeX forráskódot, létrehozza a Postscript fájlt, és a sallangokat meg letörölheti. Egy példa „Makefile” a következőképp néz ki:

```
all:  latex.ps

latex.dvi:→ latex.tex
→→→→ latex latex.tex
→→→→ latex latex.tex

latex.ps:  → latex.dvi
→→→→ dvips latex.dvi -o latex.ps
→→→→ rm latex.aux latex.log latex.dvi

tgz:
→→→→ tar cvzf doksi.tgz latex.*
```

A tgz sor segítségével például egyszerűen összetömöríthetjük a munkánkat egyetlen `tgz` fájlba, ugyanis ha ez a sor megtalálható a „Makefile”-unkban, akkor a

```
$ make tgz
```

parancsra létrejön a tgz fájl. Mint látható a program kétszer fordítja le a forráskódot, azaz rögtön létre is jön a Tartalomjegyzék. A „Makefile” általános Unix-os dolog, nem kifejezetten a LaTeXhez lett kitalálva, segítségével bonyolult, objektum-orientált forráskódok fordítása válik lehetővé.

Hogy hogy is működik a „Makefile”? Az elején megadjuk neki az `all`-lal, hogy mi a fő feladata. Esetünkben az, hogy létrehozza a `latex.ps` nevű fájlt. Mint látható, ez az `all`-lal egy sorban van írva. Ez nagyon fontos, ez azt jelöli, hogy valahol a fájlban van egy szabály arra, hogy hogyan állítsa elő a fájlt (vagy esetleg azt, hogy elérhető a gyökérben). A `latex.ps` sorban láthatjuk, hogy neki egy `latex.dvi` nevű fájl kell. Erre meg egy másik parancssor van, hogy hogyan is állítható elő. Miért így adjuk meg, és nem pedig egyben? Azért mert az elején biztosan lesz sok „bug”-os sor, így akkor felesleges a dvi-ból a Postscript-et előállítani, ekkor elég a

```
$ make latex.dvi
```

parancsot kiadni, és csak a latex-hes sorok hajtódnak parancsként végre.

Természetesen a LaTeXhel kivitelezhető nyomdatechnikai trükkök es eljárások száma végtelen, gyakorlatilag bármit elő lehet állítani a LaTeX segítségével konferencia posztertől kezdve egészen előadás diákig bármit. Ez a kis dokumentum csak egy kis ízelítőt adott a LaTeX „erejéről”. Ha valamikor is elakadnátok, segítségül lehet hívni a www.google.co.hu segítségét. A neten rengeteg LaTeXhes tanács, trükk és tipp található.