LaTeX 2ε Osztály és csomagírók számára

1996. november 28.

Contents

1 Bevezetés **2**
1.1 Osztályok és csomagok írása LaTeX 2ε-hez . . . 2
1.2 Áttekintés . . . 2
1.3 Bővebb információk . . . 3

2 Osztályok és csomagok írása **3**
2.1 Régi verziók . . . 3
2.2 A ‘docstrip’ és a ‘doc’ használata . . . 4
2.3 Osztály vagy csomag? . . . 4
2.4 Parancs nevek . . . 4
2.5 Doboz parancsok és színek . . . 5
2.6 Szöveg és matematikai karakterek definiálása . . . 6
2.7 Általános stílus . . . 6

3 Az osztályok és csomagok szerkezete **8**
3.1 Azonosítás . . . 9
3.2 Osztályok és csomagok használata . . . 9
3.3 Opciók deklarálása . . . 10
3.4 Egy minimális osztály fájl . . . 12
3.5 Példa: egy helyi levél osztály . . . 12
3.6 Példa: egy hírlevél osztály . . . 13

4 Parancsok osztály és csomag írók számára **14**
4.1 Azonosítás . . . 14
4.2 Fájlok betöltése . . . 15
4.3 Opció deklarálás . . . 16
4.4 Parancsok az opciókódon belül . . . 17
4.5 Opciók mozgatása . . . 17
4.6 Késleltetett kód . . . 19
4.7 Opció feldolgozás . . . 20
4.8 Megbízható fájlparancsok . . . 22
4.9 Hibák jelzése, stb . . . 22
4.10 Parancsok definiálása . . . 23
4.11 Mozgó argumentumok . . . 24

5 Különféle parancsok, stb **25**
5.1 Elrendezési paraméterek 25
5.2 Betű változtatás 25
5.3 Az ‘openany’ opció a ‘book’ osztályban 26
5.4 Jobb, felhasználó által adott matematikai megjelenítő környezetek . . 26

6 LaTeX 2.09 osztályok és csomagok korszerűsítése **26**
6.1 Először próbáljuk ki! 27
6.2 Hibaelhárítás 27
6.3 Betű parancsok 27
6.4 Elavult parancsok 29

1 Bevezetés

A dokumentum bemutatja, hogy kell osztályokat és csomagokat írni LaTeX-ben, figyelve a meglévő LaTeX 2.09 csomagok LaTeX 2_ε-be való átírására. Az utóbbi témával foglalkozik Johannes Braams TUGboat 15.3-ban írt cikke is.

1.1 Osztályok és csomagok írása LaTeX 2_ε-hez

A LaTeX egy olyan dokumentumkészítő rendszer, amely lehetővé teszi a dokumentum írójának, hogy a tartalomra figyeljen, a szöveg formázása nélkül. Például a fejezetek nem 18pt méretű félkövér betűvel vannak jelölve, hanem `\chapter{`⟨*cím*⟩`}`-mel.

A *dokumentum osztály* tartalmazza, hogyan kell logikai kifejezéseket (például ‘`\chapter`’-t) formázássá (‘18pt félkövér jobb egyenetlen’-né) alakítani. Ezen kívül néhány dokumentum osztálytól független tulajdonság (szín, vagy ábra) a *csomagok*ban van.

Az egyik legnagyobb különbség a LaTeX 2.09 és a LaTeX 2_ε között a csomagok és osztályok írására használt parancsokban van. A LaTeX 2.09 alig támogatja a `.sty` fájlok írását, így az íróknak alacsony szintű utasításokat kellett használni.

A LaTeX 2_ε támogatja a csomagok szerkesztését magas szintű utasításokkal. Sokkal könnyebb osztályokat és csomagokat írni egymásra építve, mondjuk egy `cetechr` szakmai jelentés osztályt írni (például a Vegyészeti kutatórészlegnek) az `article` alapján.

1.2 Áttekintés

Ez a dokumentum áttekintést ad arról, hogyan írhatunk osztályokat és csomagokat LaTeX-hez. Nem mutatja be a csomagok írásához szükséges összes parancsot: ezek a *LaTeX: A Document Preparation System* vagy *The LaTeX Companion*-ban megtalálhatók. De leírja az osztályok és csomagok írásához szükséges új utasításokat.

2.7 fejezet néhány általános tanács az osztályok és csomagok írásáról. Leírja az osztályok és csomagok közti különbséget, az utasítás elnevezési szokásokat, a `doc` és `docstrip` használatát, hogy kapcsolódnak a TeX egyszerű fájl és doboz parancsai a LaTeX-kel. Tartalmaz még néhány tanácsot az általános LaTeX stílusról.

3 fejezet bemutatja az osztályok és csomagok szerkezetét. Leírja, hogy kell az osztályokat és csomagokat egymásra építeni, opciókat és parancsokat deklarálni. Tartalmaz példaosztályokat is.

4 fejezet az új osztályok és csomagok listája.

6 fejezet részletes leírás, hogyan kell LaTeX 2.09 osztályt vagy csomagot LaTeX 2_ε-re frissíteni.

1.3 Bővebb információk

A LaTeX-ről általános információk olvashatóak beleértve a LaTeX 2_ε új tulajdonságait Leslie Lamport *LaTeX: A Document Preparation System* [3] könyvében.

A LaTeX újdonságainak sokkal részletesebb leírása és több mint 150 csomag áttekintése van Michel Goossens, Frank Mittelbach és Alexander Samarin *The LaTeX Companion* [1] című könyvében.

A LaTeX alapja a TeX, melynek leírása Donald E. Knuth *The TeXbook* [2] könyvében van.

A LaTeX-hez sok olyan dokumentum tartozik, amely minden verzióban benne van. A `ltnews*.tex` fájlokban található *LaTeX News* együtt jelenik meg a félévente megjelenő LaTeX-kel. A LaTeX újdonságait leíró *LaTeX 2_ε for Authors* szerzői útmutató a `usrguide.tex`-ben van. A `fntguide.tex`-ben lévő *LaTeX 2_ε Font Selection* a LaTeX betűtípusokat írja le. A LaTeX konfigurálásával foglalkozó *Configuration options for LaTeX 2_ε* a `cfgguide.tex`-ben van, míg a LaTeX módosítása mögötti filozófiát leíró *Modifying LaTeX* a `modguide.tex`-ben található.

A LaTeX forráskódját fokozatosan átírjuk egy LaTeX dokumentumba: *LaTeX: the program*. Ez a dokumentum tartalmazza a LaTeX utasítások listáját és a `source2e.tex`-ből szedhető.

További információért lépjen kapcsolatba a helyi TeX Felhasználói Csoporttal, vagy a nemzetközi TeX Felhasználói Csoporttal. Hasznos címek:

TeX Users Group, P.O. Box 1239, Three Rivers, CA 93271-1239, USA
Fax: +1 209 561 4584 Email: `tug@mail.tug.org`

UK TUG, 1 Eymore Close, Selly Oak, Birmingham B29 4LB, UK
Fax: +44 121 476 2159 Email: `uktug-enquiries@tex.ac.uk`

2 Osztályok és csomagok írása

Ez a fejezet tartalmaz néhány általános információt a LaTeX osztályok és csomagok írásáról.

2.1 Régi verziók

Ha egy létező LaTeX 2.09 stílus fájlt frissít, akkor azt javasoljuk, hogy többé ne használja a 2.09-es verziót. Mivel a 90-es évek elején sokféle nyelvjárása létezett

a LaTeX-nek, ezért szinte lehetetlen a LaTeX különböző verzióihoz csomagot fenntartani. Természetesen egy kis ideig néhány szervezetnek párhuzamosan fenn kell tartani mindkét verziót, de ez nem vonatkozik az egyénileg készített csomagokra: ezeknek csak az új szabványos LaTeX 2_ε-t kellene támogatniuk, nem az elavult LaTeX-et.

2.2 A 'docstrip' és a 'doc' használata

Ha egy nagyméretű osztályt vagy csomagot ír, akkor érdemes megfontolni a LaTeX-kel együtt megjelent `doc` program használatát. Kétféleképpen lehet így LaTeX osztályokat és csomagokat írni. Lehet a LaTeX segítségével dokumentációt készíteni, vagy `docstrip`-pel készíteni `.cls` vagy `.sty` állományokat.

A `doc` program tud automatikusan indexeket generálni a használható definíciókról és utasításokról. Ez nagyon hasznos nagy TeX források karbantartására és dokumentálására.

A LaTeX kernel, az alapvető osztályok stb. forrásdokumentumai `doc` dokumentumok. Ezek a disztribúció `.dtx` fájljaiban találhatóak. Valójában szedheti a kernel forráskódját egyetlen hosszú dokumentumként, és kiegészítheti indexszel ha LaTeX-kel lefordítja a `source2e.tex`-et.

További információkért tanulmányozza a `docstrip.dtx` és a `doc.dtx` fájlokat, valamint a *The LaTeX Companion*-t. A használatukról példák találhatóak a `.dtx` fájlokban.

2.3 Osztály vagy csomag?

Az első teendő mielőtt új LaTeX parancsokat használunk egy fájlban, az hogy el kell dönteni *dokumentum osztály* vagy *csomag*. A döntési irányelvek:

> Ha a parancsok használhatóak dokumentum osztállyal, akkor készítsen csomagot, ha nem akkor osztályt.

Két főbb típusa van az osztályoknak: az önálló osztályok, mint az `article`, `report` vagy `letter`, és azok amelyek más osztályok kiterjesztései vagy változatai, például a `proc` dokumentum osztály, ami az `article` osztályon alapul.

Így egy vállalatnak lehet egy helyi `ownlet` osztálya, amivel leveleket nyomtathat a cég saját levélpapírjára. Ilyen osztály készíthető a már létező `letter` osztályra alapozva, de nem használható a többi dokumentum osztállyal, tehát inkább `ownlet.cls`-t csináljunk, mint `ownlet.sty`-t.

A `graphics` csomag viszont tartalmaz parancsokat képek LaTeX dokumentumokba való beillesztésére. Mivel ezek a parancsok használhatók más dokumentum osztályokkal is, ezért inkább `graphics.sty` legyen mint `graphics.cls`.

2.4 Parancs nevek

LaTeX-ben háromféle parancs létezik.

Vannak a szerzői parancsok, mint például `\section`, `\emph` és `\times`: ezek általában rövid nevűek és csupa kisbetűsek.

Osztály és csomag író parancsok: ezek nevében többnyire vegyesen van kis- és nagybetű, például:

```
\InputIfFileExists  \RequirePackage  \PassOptionsToClass
```

Végül vannak a LaTeX-ben használt belső parancsok, például `\@tempcnta`, `\@ifnextchar` és `\@eha`: a legtöbb ilyen parancs tartalmaz `@`-t a nevében, ez azt jelenti hogy nem használhatóak dokumentumokban, csak osztály és csomag fájlokban.

Sajnos történeti okok miatt a parancsok közti különbség gyakran zavaros. Például a `\hbox` egy belső parancs, amit csak a LaTeX kernelben kellene használni, viszont `\m@ne` a konstans -1, ami lehetne `\MinusOne` is.

Azonban ez a szokás mégis hasznos: ha egy parancs nevében van `@`, akkor az nem része a támogatott LaTeX-nek—és a tulajdonságai lehet, hogy meg fognak változni a későbbi kiadásokban. Ha egy parancs nevében szerepel kis- és nagybetű is, vagy a *LaTeX: A Document Preparation System*-ben van leírva, akkor számíthat rá, hogy későbbi LaTeX 2_ε kiadások támogatni fogják a parancsot.

2.5 Doboz parancsok és színek

Még akkor is, ha nem szándékozik színeket használni a saját dokumentumokban, megfigyelve a fejezet lényegét, biztosíthatja, hogy a csomagja vagy osztálya kompatibilis legyen a `color` csomaggal. Ez hasznos lehet azoknak, akik az osztályát vagy csomagját használják és hozzáférnek színes nyomtatóhoz.

A legegyszerűbb mód biztosítani a ‘színek épségét’, ha csak LaTeX doboz parancsokat használ a TeX primitívek helyett. Ez azt jelenti, hogy `\setbox` helyett `\sbox`-ot, `\hbox` helyett `\mbox`-ot és `\vbox` helyett `\parbox`-ot vagy `minipage` környezeti változót használ. A LaTeX doboz parancsok az új tulajdonságokkal olyan hatékonyak, mint a TeX primitívek.

Mint a hibalehetőségek egy példája, nézzük meg hogy a `{\ttfamily` ⟨*szöveg*⟩`}`-ben a betűtípus pontosan a `}` *előtt* állítódik vissza, viszont a hasonló kinézetű `{\color{green}` ⟨*szöveg*⟩`}`-ben a szín csak az utolsó `}` *után*. Rendszerint ez a különbség nem okoz gondot, de nézzünk meg egy egyszerű TeX doboz kifejezést:

```
\setbox0=\hbox{\color{green} ⟨szöveg⟩}
```

A szín visszaállítása a `}` után történik és nem tárolódik a dobozban. Ennek pontosan milyen rossz következménye lehet, attól függ, hogyan valósították meg a színt: a dokumentum többi része rossz színű lesz, vagy hibát okoz a dokumentum nyomtatásánál használt dvi-programban.

Szintén érdekes a `\normalcolor` parancs. Ez általában csak `\relax` (azaz nem csinál semmit), de `\normalfont` helyett inkább ezt javasoljuk az oldal olyan részeinek a ‘dokumentum alapszín’-éhez állítására, mint például a címsor vagy fejezetcímek.

2.6 Szöveg és matematikai karakterek definiálása

Mivel a LaTeX 2ε különböző kódolásokat támogat, szimbólumok, ékezetek, összetett betűk létrehozásának parancsait az erre a célra biztosított és a *LaTeX 2ε Font Selection*-ban leírt parancsok felhasználásával kell definiálni. Ez a része a rendszernek még fejlesztés alatt van, ezért az ilyen feladatokat fokozott elővigyázatossággal kell elkészíteni.

Az ilyen típusú kódolásfüggetlen parancsokhoz a `\DeclareRobustCommand` használható.

Jegyezzük meg hogy ezentúl matematikai módon kívül nem lehet a matematikai betűtípus beállításra hivatkozni: például nincs biztosítva sem a `\textfont 1` sem a `\scriptfont 2` definíciója más módokban.

2.7 Általános stílus

Az új rendszer sok olyan parancsot tartalmaz, ami segít stabil és hordozható osztály és csomag fájlokat készíteni. A fejezet az előző parancsok értelmes használatáról ad vázlatos áttekintést.

2.7.1 Különböző fájlok betöltése

A LaTeX a következő parancsokat tartalmazza:

New description 1995/12/01

```
\LoadClass          \LoadClassWithOptions
\RequirePackage     \RequirePackageWithOptions
```

osztályok és csomagok használatára másik osztályon vagy csomagon belül. Több okból ajánlatos inkább ezeket használni, mint az egyszerű `\input` parancsot.

Az `\input` ⟨*fájlnév*⟩-vel betöltött fájlok nem lesznek benne a `\listfiles` listában.

Ha egy csomagot mindig a `\RequirePackage...` vagy a `\usepackage` parancs tölt be, még ha többször is szükséges, csak egyszer fog betöltődni. Ezzel szemben ha a csomagot `\input`-tal töltjük be, akkor többször is betöltődhet, az ilyen plusz terhelés idő- és memóriapocsékolás, és szokatlan eredményt produkálhat.

Ha egy csomag biztosítja az opció feldolgozást, akkor szintén különös eredményt kaphatunk ha a csomagot a `\usepackage` vagy `\RequirePackage...` használata helyett az `\input` tölti be.

Ha a `foo.sty` betölti a `baz.sty` csomagot az `\input baz.sty` paranccsal akkor a felhasználó hibaüzenetet kap:

```
LaTeX Warning: You have requested package ‘foo’,
               but the package provides ‘baz’.
```

Így az `\input` használata csomagok betöltésére több okból is rossz ötlet.

Sajnálatos módon ha a `myclass.sty`-t egy osztály fájllá frissíti, akkor meg kell győződnie róla, hogy bármelyik fájl amely tartalmazza az `\input myclass.sty` parancsot, még mindig működik.

Ez igaz volt az alapvető osztályokra (`article`, `book` és `report`), mivel sok meglevő LaTeX 2.09 dokumentum stílus tartalmaz `\input article.sty` parancsot. A megoldáshoz vezető szemlélet minimális `article.sty`, `book.sty` és `report.sty` fájlok létrehozása, amelyek csak a megfelelő osztály fájlokat töltik be.

Például az `article.sty` csak a következő sorokat tartalmazza:

```
\NeedsTeXFormat{LaTeX2e}
\@obsoletefile{article.cls}{article.sty}
\LoadClass{article}
```

Választhat, hogy így csinálja, vagy ha biztonságosnak gondolja, kihagyhatja a `myclass.sty` fájlt.

2.7.2 Tegyük stabillá

Helyesnek tartjuk, ha csomagok és osztályok írásánál a lehető legtöbbször LaTeX parancsot használunk.

Ezért a `\newcommand`, `\renewcommand` vagy `\providecommand` parancsokat javasoljuk a `\def...` helyett; a `\CheckCommand` szintén hasznos parancs. Ha így tesz, akkor kevésbé valószínű, hogy figyelmetlenül újradefiniál egy parancsot, ami váratlan eredményt okozhat.

Amikor egy új környezetet definiál, a `\def\foo{...}`, `\def\endfoo{...}` helyett használja a `\newenvironment` vagy `\renewenvironment` utasításokat.

Ha a ⟨*dimenzió*⟩ vagy ⟨*ugrás*⟩ regisztert kell beállítani vagy megváltoztatni, használja a `\setlength` parancsot.

Dobozok kezelésére használja inkább a LaTeX parancsokat, például `\sbox`, `\mbox` és `\parbox` mint a `\setbox`, `\hbox` és `\vbox` parancsokat.

Használja a `\PackageError`, `\PackageWarning` vagy `\PackageInfo` parancsokat (vagy a megfelelő osztály utasításokat) a `\@latexerr`, `\@warning` vagy `\wlog` helyett.

Még mindig lehetséges opciókat létrehozni a `\ds@`⟨*opció*⟩ definiálásával és az `\@options` meghívásával; de a `\DeclareOption` és `\ProcessOptions` parancsokat ajánljuk helyettük. Ezek sokkal jobbak és kevesebb memóriát használnak. Tehát:

```
\def\ds@draft{\overfullrule 5pt}
\@options
```

helyett inkább ezt használja:

```
\DeclareOption{draft}{\setlength{\overfullrule}{5pt}}
\ProcessOptions\relax
```

Ennek a gyakorlatnak az előnye, hogy a kód sokkal olvashatóbb, és ami még fontosabb, kevésbé valószínű, hogy nem működik a jövőbeli LaTeX verziókkal.

2.7.3 Tegyük hordozhatóvá

Szintén ésszerű a fájlokat amennyire csak lehet hordozhatóvá tenni. Ezért a fájlok csak látható 7-bites szöveget tartalmazhatnak, és a fájlnevek maximum 8 karakter hosszúak lehetnek (plusz három a kiterjesztés). Természetesen a név **nem lehet** ugyanaz mint egy szabványos LaTeX-beli fájlnév, azonban hasonló tartalma lehet.

Szintén hasznos, ha a saját osztályokban vagy csomagokban megszokott előtag van, például a University of Nowhere osztályok kezdete lehet `unw`. Ez segít elkerülni, hogy minden egyetemnek saját tézis osztálya legyen ugyanazzal a `thesis.cls` névvel.

Ha használja a LaTeX kernel vagy egy csomag sajátosságait, kérjük adja meg a szükséges kiadás dátumát. Például a hiba parancsok csomag az 1994 júniusi kiadásban jelent meg, így ha ezeket használja, be kell írnia a következő sort:

```
\NeedsTeXFormat{LaTeX2e}[1994/06/01]
```

2.7.4 Hasznos hurkok

Némely csomag és dokumentum stílus újra definiálta a `\document` vagy az `\enddocument` parancsot a célja elérése érdekében. Ez a továbbiakban nem szükséges, használható az `\AtBeginDocument` és az `\AtEndDocument` hurok (lásd 4.6 fejezet). A hurok használatával kevésbé valószínű, hogy a kód nem működik a következő LaTeX verziókkal. És sokkal valószínűbb, hogy a csomag működni fog mások csomagjaival együtt.

Azonban megjegyzendő, hogy az `\AtBeginDocument` hurok kódja a bevezetés része. Tehát ide nem írhatunk bármit, különösen nem lehet típusbeállítást használni.

New description 1996/12/01

3 Az osztályok és csomagok szerkezete

A LaTeX 2_ε osztályok és csomagok jobb szerkezetűek, mint a LaTeX 2.09 stílus fájlok. Az osztály vagy csomag fájlok vázlata:

Azonosítás Itt tudjuk meg, hogy a fájl LaTeX 2_ε csomag vagy osztály, és egy rövid leírást kapunk róla.

Bevezető deklarációk Itt történik a parancsok deklarálása és egyéb állományok betöltése. Általában ezek a parancsok azok, amelyek a deklarált opciókhoz szükségesek.

Opciók Itt történik az opciók deklarálása és feldolgozása.

További deklarációk A legtöbb dolog ebben a részben történik: új változók, parancsok, betűtípusok deklarálása; és egyéb fájlok betöltése.

3.1 Azonosítás

Először az osztály, vagy csomag fájl azonosítja magát. A csomag fájlok a következőképpen teszik ezt:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{⟨csomag⟩}[⟨dátum⟩ ⟨egyéb információ⟩]
```

Például:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{latexsym}[1994/06/01 Standard LaTeX package]
```

Az osztály fájlok pedig így:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{⟨osztály név⟩}[⟨dátum⟩ ⟨egyéb információ⟩]
```

Például:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{article}[1994/06/01 Standard LaTeX class]
```

A ⟨*dátum*⟩-ot 'ÉÉÉÉ/HH/NN' formában kell megadni.

New description 1998/06/19

Ennek a dátumnak az ellenőrzése akkor történik, amikor a felhasználó dátumot ad meg a `\documentclass` vagy `\usepackage` parancsban. Például ha ezt írná:

```
\documentclass{article}[1995/12/23]
```

akkor a különböző helyen lévő felhasználók egy figyelmeztetést kapnának, hogy az `article` másolatuk nem naprakész.

Az osztály leírása megjelenik használat közben. A csomag leírása a log fájlba íródik. Ezek a leírások megnézhetők a `\listfiles` paranccsal is. A `Standard LaTeX` kifejezés **nem** használható bármilyen, a szabványos LaTeX disztribúcióban nem szereplő fájl azonósító részében.

3.2 Osztályok és csomagok használata

Az első jelentős különbség a LaTeX 2.09 stílus fájlok és a LaTeX 2_ε csomagok között az, hogy a LaTeX 2_ε támogatja a *modularitást*, olyan értelemben, hogy a fájlokat inkább kisebb építőelemekből rakjuk össze, mint nagy fájlokból.

Egy LaTeX csomag vagy osztály következőképpen tölthet be csomagot:

```
\RequirePackage[⟨opciók⟩]{⟨csomag⟩}[⟨dátum⟩]
```

Például:

```
\RequirePackage{ifthen}[1994/06/01]
```

A parancsnak ugyanolyan a szintaxisa, mint a `\usepackage` szerzői parancsnak. Az előző parancs lehetővé teszi csomagok vagy osztályok számára más csomagbeli tulajdonság használatát. Például az `ifthen` csomag betöltése után a csomagíró használhatja az 'if…then…else…' parancsokat, amiket az `ifthen` csomag tartalmaz.

Egy LaTeX osztály a következőképpen tölthet be egy másik osztályt:

```
\LoadClass[⟨opciók⟩]{⟨osztály név⟩}[⟨dátum⟩]
```

Például:

```
\LoadClass[twocolumn]{article}
```

A parancsnak ugyanaz a szintaxisa, mint a `\documentclass` szerzői parancsnak. Így az osztályok egy másik osztály szintaxisán és megjelenésén alapulhatnak. Például az `article` osztály betöltése után az osztály írójának csak azokat a részeket kell megváltoztatni, amik nem tetszenek neki, és nem kell a semmiből egy új osztályt írni.

A következő parancsok abban a gyakori esetben használhatóak amikor csak egy osztály vagy csomag fájlt akar betölteni pontosan azokkal a beállításokkal, amiket az aktuális osztály használ.

New feature 1995/12/01

```
\LoadClassWithOptions{⟨osztály név⟩}[⟨dátum⟩]
\RequirePackageWithOptions{⟨csomag⟩}[⟨dátum⟩]
```

Például:

```
\LoadClassWithOptions{article}
\RequirePackageWithOptions{graphics}[1995/12/01]
```

3.3 Opciók deklarálása

A másik jelentős különbség LaTeX 2.09 stílusok és LaTeX 2_ε csomagok és osztályok között az opciók kezelésében van. Csomagok és osztályok deklarálhatnak opciókat és ezeket megadhatják a szerzők. Például a `twocolumn` opciót az `article` osztály deklarálta.

New description 1998/12/01

Egy opciót a következőképpen kell deklarálni:

```
\DeclareOption{⟨opció⟩}{⟨kód⟩}
```

Például a `dvips` opció megvalósítása (egy kissé leegyszerűsítve) a `graphics` csomaghoz így néz ki:

```
\DeclareOption{dvips}{\input{dvips.def}}
```

Ez azt jelenti, hogy amikor a szerző azt írja, `\usepackage[dvips]{graphics}`, akkor a `dvips.def` fájl betöltödik. Egy másik példa, az `article` osztályban deklarált `a4paper` opcióval a `\paperheight` és `\paperwidth` hosszát lehet beállítani:

```
\DeclareOption{a4paper}{%
   \setlength{\paperheight}{297mm}%
   \setlength{\paperwidth}{210mm}%
}
```

Néha a felhasználók olyan opciókat kérhetnek, amiket az osztály vagy csomag nem egyértelműen deklarált. Alapértelmezés szerint ez (osztályoknál) figyelmeztetést vagy (csomagoknál) hibaüzenetet eredményez. Ezt a következő módon változtathatjuk meg:

```
\DeclareOption*{⟨kód⟩}
```

Például megadhatjuk, hogy a `fred` csomag hibaüzenet helyett inkább figyelmeztetést adjon ismeretlen opció esetén:

```
\DeclareOption*{%
   \PackageWarning{fred}{Unknown option `\CurrentOption'}%
}
```

Ezután ha a szerző azt írja, hogy `\usepackage[foo]{fred}`, akkor egy `Package fred Warning: Unknown option `foo'` figyelmeztetést kap. Egy másik példa, a `fontenc` csomag megpróbálja betölteni az ⟨*ENC*⟩`enc.def` fájlt, amikor az ⟨*ENC*⟩ opciót használja. Ezt így lehet megcsinálni:

```
\DeclareOption*{%
   \input{\CurrentOption enc.def}%
}
```

Lehetőségünk van opciókat egy másik csomagnak, vagy osztálynak átadni, a `\PassOptionsToPackage` vagy `\PassOptionsToClass` paranccsal. Például az összes ismeretlen opció `article` osztálynak való átadására ez használható:

```
\DeclareOption*{%
   \PassOptionsToClass{\CurrentOption}{article}%
}
```

Ha ezt csinálja, akkor győződjön meg róla, hogy az osztály később valamikor betöltődik, egyébként az opciók soha nem érvényesülnek!

Eddig csak megmagyaráztuk hogyan kell opciókat deklarálni, azt nem hogy kell érvényesíteni őket. A fájlnak megadott opciók végrehajtásához a következőket kell tenni:

```
\ProcessOptions\relax
```

Ez minden megadott és deklarált opcióra végrehajtja a ⟨*kód*⟩-ot (a részleteket lásd a 4.7 fejezetben).

Például, ha a `jane` csomag ezt tartalmazza:

```
\DeclareOption{foo}{\typeout{Saw foo.}}
\DeclareOption{baz}{\typeout{Saw baz.}}
\DeclareOption*{\typeout{What's \CurrentOption?}}
\ProcessOptions\relax
```

és a szerző azt írja, hogy `\usepackage[foo,bar]{jane}`, akkor a következő üzeneteket fogja látni: `Saw foo.` és `What's bar?`

3.4 Egy minimális osztály fájl

Egy csomag vagy osztály legfontosabb dolga új parancsok definiálása, vagy a dokumentum megjelenésének megváltoztatása. Ez a csomag törzsében történik olyan parancsok használatával, mint a `\newcommand` és a `\setlength`.

A LaTeX 2_ε több új parancsot tartalmaz az osztály és csomag írók segítésére, bővebben a 4 fejezetben.

Három dolgot *kell* tartalmaznia minden osztály fájlnak: ezek a `\normalsize` meghatározása, a `\textwidth` és a `\textheight` értékei. Tehát egy minimális dokumentum osztály fájl[1] így néz ki:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{minimal}[1995/10/30 Standard LaTeX minimal class]
\renewcommand{\normalsize}{\fontsize{10pt}{12pt}\selectfont}
\setlength{\textwidth}{6.5in}
\setlength{\textheight}{8in}
```

Azonban ez az osztály nem fogja támogatni a lábjegyzeteket, a margókat stb., sem a kétbetűs betűtípus parancsokat, mint például az `\rm`; ezért a legtöbb osztály ennél többet fog tartalmazni!

3.5 Példa: egy helyi levél osztály

Egy vállalatnak lehet saját levél osztálya a levelek stílusának kialakítására. Ez a fejezet egy ilyen osztály egyszerű megvalósítását mutatja be, habár egy valóságos osztály több szerkesztést igényel.

Az osztály a `neplet.cls` név bevezetésével kezdődik.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{neplet}[1995/04/01 NonExistent Press letter class]
```

[1] Ez az osztály `minimal.cls` néven benne van a szabványos kiadásban.

A következő rész átadja az opciókat a `letter` osztálynak, amely az `a4paper` opcióval lett betöltve.

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{letter}}
\ProcessOptions\relax
\LoadClass[a4paper]{letter}
```

Azért, hogy a vállalat levél fejlécét használjuk, újradefiniáljuk a `firstpage` oldal stílust: ezt az oldal stílust használják a levelek első oldalán.

```
\renewcommand{\ps@firstpage}{%
   \renewcommand{\@oddhead}{⟨ide kerül a levélfej⟩}%
   \renewcommand{\@oddfoot}{⟨ide kerül a levéllább⟩}%
}
```

Ennyi az egész!

3.6 Példa: egy hírlevél osztály

Az `article` osztály egyik változatával készíthető egy egyszerű hírlevél LaTeX-ben. Az osztály a `smplnews.cls` név bevezetésével kezdődik.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{smplnews}[1995/04/01 The Simple News newsletter class]

\newcommand{\headlinecolor}{\normalcolor}
```

A legtöbb megadott opciót átadja az `article` osztálynak: kivéve a `onecolumn` opciót, amelyet kikapcsol és a `green` opciót, ami a főcímet zöldre állítja be.

```
\DeclareOption{onecolumn}{\OptionNotUsed}
\DeclareOption{green}{\renewcommand{\headlinecolor}{\color{green}}}

\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}

\ProcessOptions\relax
```

Ezután betölti az `article` osztályt a `twocolumn` opcióval.

```
\LoadClass[twocolumn]{article}
```

Mivel a hírlevelet színesben kell nyomtatni, betölti a `color` csomagot. Az osztály nem ad meg eszközmeghajtó opciót, mert ezt a `smplnews` osztály felhasználójának kell meghatároznia.

```
\RequirePackage{color}
```

Majd az osztály újradefiniálja a `\maketitle`-t, hogy a címet megfelelő színben és Helvetica félkövér dőlt betűtípusban, 72 pt méretben mutassa.

```
\renewcommand{\maketitle}{%
   \twocolumn[%
      \fontsize{72}{80}\fontfamily{phv}\fontseries{b}%
      \fontshape{sl}\selectfont\headlinecolor
      \@title
   ]%
}
```

Ez felülírja a `\section`-t és kikapcsolja a fejezetszámozást.

```
\renewcommand{\section}{%
   \@startsection
      {section}{1}{0pt}{-1.5ex plus -1ex minus -.2ex}%
      {1ex plus .2ex}{\large\sffamily\slshape\headlinecolor}%
}

\setcounter{secnumdepth}{0}
```

Beállítja a három alapvető dolgot.

```
\renewcommand{\normalsize}{\fontsize{9}{10}\selectfont}
\setlength{\textwidth}{17.5cm}
\setlength{\textheight}{25cm}
```

A gyakorlatban egy osztály ennél sokkal többet kíván: tartalmaznia kell a kiadási szám, a cikkek szerzője, oldal stílusok és hasonlók parancsait; de ez a vázlat kiindulási pontot nyújt. Az `ltnews` osztály fájl nem sokkal bonyolultabb, mint ez.

4 Parancsok osztály és csomag írók számára

Ez a fejezet röviden leírja az osztály és csomag írók új parancsait. Az új rendszer más jellegzetességeinek megismeréséhez lásd még: *LaTeX: A Document Preparation System*, *The LaTeX Companion* és *LaTeX 2ε for Authors*.

4.1 Azonosítás

Először a saját osztály vagy csomag fájlok azonosítására használt parancsokat tárgyaljuk.

`\NeedsTeXFormat {`⟨*formátum név*⟩`} [`⟨*kiadási dátum*⟩`]`

Ez a parancs jelzi a TeX-nek, hogy a fájl a ⟨*formátum név*⟩ formátumban kerül feldolgozásra. Az elhagyható ⟨*kiadási dátum*⟩ paraméterben megadható a formátumhoz szükséges legkorábbi kiadás dátuma. Ha a kiadás dátuma régebbi mint a megadott, akkor figyelmeztetést kapunk. A szabványos ⟨*formátum név*⟩ `LaTeX2e`. Ha megadtunk dátumot, annak ÉÉÉÉ/HH/NN formában kell lennie.

Példa:

```
\NeedsTeXFormat{LaTeX2e}[1994/06/01]
```

`\ProvidesClass {`⟨*osztály név*⟩`} [`⟨*kiadási információ*⟩`]`
`\ProvidesPackage {`⟨*csomag név*⟩`} [`⟨*kiadási információ*⟩`]`

Ez deklarálja, hogy az aktuális fájl tartalmazza az ⟨*osztály név*⟩ dokumentum osztály vagy a ⟨*csomag név*⟩ csomag definícióit.

Ha használjuk az elhagyható ⟨*kiadási információ*⟩-t, akkor tartalmaznia kell:

- a fájl jelen verziójának kiadási dátumát ÉÉÉÉ/HH/NN formában;
- ezek után következhet egy szóköz és egy rövid leírás, esetleg a verziószámmal.

A fenti szabályokat pontosan be kell tartani, hogy ezek az információk felhasználhatóak legyenek (osztályok esetén) a `\LoadClass` vagy `\documentclass` vagy (csomagok esetén) a `\RequirePackage` vagy `\usepackage` által, a kiadás dátumának tesztelésére.

Az egész ⟨*kiadási információ*⟩-t megmutatja a `\listfiles` parancs, ezért nem lehet túl hosszú.

Példa:

```
\ProvidesClass{article}[1994/06/01 v1.0 Standard LaTeX class]
\ProvidesPackage{ifthen}[1994/06/01 v1.0 Standard LaTeX package]
```

`\ProvidesFile {`⟨*fájl név*⟩`} [`⟨*kiadási információ*⟩`]`

Ez hasonló az előző két parancshoz, kivéve, hogy itt az egész fájl nevet a kiterjesztéssel együtt meg kell adni. Ez a parancs a fő osztály és csomag fájloktól különböző fájlok deklarálásához használható.

Példa:

```
\ProvidesFile{T1enc.def}[1994/06/01 v1.0 Standard LaTeX file]
```

Jegyezzük meg, hogy a `Standard LaTeX` kifejezést **nem szabad** használni bármilyen fájl azonosítási részében, csak azokban amelyek benne vannak a szabványos LaTeX disztribúcióban.

4.2 Fájlok betöltése

A parancsok ezen csoportját saját dokumentum osztályok vagy csomagok létrehozására használhatjuk, már meglévő osztályokra vagy csomagokra építve.

New feature 1995/12/01

`\RequirePackage [`⟨*opció lista*⟩`] {`⟨*csomag név*⟩`} [`⟨*kiadási információ*⟩`]`
`\RequirePackageWithOptions {`⟨*csomag név*⟩`} [`⟨*kiadási információ*⟩`]`

A csomagok és osztályok ezen parancsokat használhatják más csomagok betöltésére.

A `\RequirePackage` használata ugyanolyan, mint a `\usepackage` szerzői parancsé.

Példa:

```
\RequirePackage{ifthen}[1994/06/01]
\RequirePackageWithOptions{graphics}[1995/12/01]
```

`\LoadClass` [⟨*opció lista*⟩] {⟨*osztály név*⟩} [⟨*kiadási információ*⟩]
`\LoadClassWithOptions` {⟨*osztály név*⟩} [⟨*kiadási információ*⟩]

Ezen parancsok *csak* osztály fájlokban használhatóak, csomag fájlokban nem; egy osztály fájlon belül legfeljebb egyszer szerepelhetnek. New feature 1995/12/01

A `\LoadClass` használata ugyanolyan, mint a `\documentclass` osztály fájlok betöltése esetén.

Példa:

```
\LoadClass{article}[1994/06/01]
\LoadClassWithOptions{article}[1995/12/01]
```

A két `WithOptions` verzió pontosan ugyanazokkal az opciókkal tölti be az osztály vagy csomag fájlt, melyeket az aktuális (osztály vagy csomag) fájl használ. További használatuk tárgyalását lásd később, a 4.5 fejezetben. New feature 1995/12/01

4.3 Opció deklarálás

A következő parancsok a dokumentum osztályok és csomagok opcióinak deklarációjával és kezelésével foglalkoznak. New description 1998/12/01

Van néhány parancs, kimondottan ezen parancsok ⟨*kód*⟩ argumentumán belüli használatára (lásd lent).

`\DeclareOption` {⟨*opció név*⟩} {⟨*kód*⟩}

Itt ⟨*opció név*⟩ egy ‘deklarált opció’ lesz abban az osztályban vagy csomagban, amibe tettük.

A ⟨*kód*⟩ argumentum a futtatandó kódot tartalmazza, ha ez az opció meghatározott az adott osztályban vagy csomagban; bármilyen érvényes LaTeX 2ε szerkezetet tartalmazhat.

Példa:

```
\DeclareOption{twoside}{\@twosidetrue}
```

`\DeclareOption*` {⟨*kód*⟩}

Ez minden olyan opcióra, ami az osztályban vagy csomagban adott, de nem közvetlenül deklarált, deklarálja a futtatandó kódot; ezen kód neve ‘alapértelmezett opció kód’ és tartalmazhat bármilyen érvényes LaTeX 2ε szerkezetet.

Ha egy osztály fájl nem tartalmaz `\DeclareOption*` parancsot, akkor alapértelmezésként az adott osztály minden specifikált, de nem deklarált opciója egyszerűen átadódik minden csomagnak (mint ahogy az adott osztály meghatározott és deklarált opciói).

Ha egy csomag fájl nem tartalmaz `\DeclareOption*` parancsot, akkor alapértelmezésként az adott csomag minden specifikált, de nem deklarált opciója egy-egy hibát fog eredményezni.

4.4 Parancsok az opciókódon belül

Ez a két parancs kizárólag egy `\DeclareOption` vagy egy `\DeclareOption*` ⟨*kód*⟩ argumentumán belül használható. Más, ezen argumentumokban gyakran használt parancsok a következő néhány alfejezetben találhatóak.

`\CurrentOption`

Ez kicserélődik az adott opció nevére.

`\OptionNotUsed`

Ez az adott opció 'használaton kívüli opciók'-hoz való hozzáadását eredményezi.

Már speciális eljárás nélkül használhatunk kettős keresztet (#) ezen ⟨*kód*⟩ argumentumokon belül (régebben meg kellett kettőzni azokat). New feature 1995/06/01

4.5 Opciók mozgatása

A most következő két parancs szintén nagyon hasznos `\DeclareOption` vagy `\DeclareOption*` ⟨*kód*⟩ argumentumán belül:

`\PassOptionsToPackage {`⟨*opció lista*⟩`} {`⟨*csomag név*⟩`}`
`\PassOptionsToClass {`⟨*opció lista*⟩`} {`⟨*osztály név*⟩`}`

A `\PassOptionsToPackage` parancs az ⟨*opció listá*⟩-ban található opciókat adja át a ⟨*csomag név*⟩ csomagnak. Ez azt jelenti, hogy az ⟨*opció listá*⟩-t hozzáadja egy későbbi, ⟨*csomag név*⟩-re vonatkozó `\RequirePackage` vagy `\usepackage` parancs által használt opciók listájához.

Példa:

```
\PassOptionsToPackage{foo,bar}{fred}
\RequirePackage[baz]{fred}
```

ugyanaz, mint

```
\RequirePackage[foo,bar,baz]{fred}
```

Hasonlóan, `\PassOptionsToClass` arra használható egy osztály fájlban, hogy egy másik, `\LoadClass` paranccsal betöltendő osztálynak opciókat adjunk át.

Hasonlítsuk össze ennek a két parancsnak a hatását és használatát a következő kettővel (dokumentálva fent, a 4.2 fejezetben):

New description 1995/12/01

```
\LoadClassWithOptions
\RequirePackageWithOptions
```

A `\RequirePackageWithOptions` parancs hasonló a `\RequirePackage`-hez, de a közvetlenül megadott vagy a `\PassOptionsToPackage` által átadott bármely opció helyett mindig pontosan ugyanazzal az opció listával tölti be a szükséges csomagot, amit az adott osztály vagy csomag használ.

A `\LoadClassWithOptions` fő célja, hogy egy osztályt egyszerűen egy másikra építhessünk, például:

```
\LoadClassWithOptions{article}
```

Ezt hasonlítsuk össze a következő, némiképp különböző szerkezettel:

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions\relax
\LoadClass{article}
```

Ahogy fent használjuk, a hatások többé-kevésbé ugyanazok, de az első sokkal kevesebb gépelést igényel; és a `\LoadClassWithOptions` eljárás kissé gyorsabban fut.

Egyébként, ha az osztály saját opciókat deklarál, akkor a két felépítés különböző. Hasonlítsuk például össze ezt:

```
\DeclareOption{landscape}{\@landscapetrue}
\ProcessOptions\relax
\LoadClassWithOptions{article}
```

ezzel:

```
\DeclareOption{landscape}{\@landscapetrue}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions\relax
\LoadClass{article}
```

Az első példában az article osztály pontosan akkor fog a `landscape` opcióval betöltődni, ha az adott osztályt azzal hívtuk. Ezzel ellentétben a második példa esetén sosem hívjuk a `landscape` opcióval, mivel abban az esetben article csak az alapértelmezett opciókezelőtől kapja az opciókat, de ezt a kezelőt nem használjuk `landscape`-re, mert ez az opció közvetlenül deklarált.

4.6 Késleltetett kód

Ezt a két első parancsot szintén a `\DeclareOption` vagy `\DeclareOption*` ⟨*kód*⟩ argumentumán belüli használatra szánták.

`\AtEndOfClass {`⟨*kód*⟩`}`
`\AtEndOfPackage {`⟨*kód*⟩`}`

Ezek a parancsok úgy deklarálják ⟨*kód*⟩-ot, hogy belsőleg kerül mentésre, majd az egész osztály vagy csomag fájl feldolgozása után hajtódik végre.

Ezen parancsok ismételt használata engedélyezett: az argumentumok kódja a deklarációjuk sorrendjében kerül tárolásra (és később futtatásra).

`\AtBeginDocument {`⟨*kód*⟩`}`
`\AtEndDocument {`⟨*kód*⟩`}`

Ezek a parancsok belső mentésre és futtatásra deklarálják ⟨*kód*⟩-ot, mialatt a LaTeX a `\begin{document}`-et vagy `\end{document}`-et dolgozza fel.

Az `\AtBeginDocument` argumentumában levő ⟨*kód*⟩ a `\begin{document}` kód vége körül hajtódik végre, *miután* a betű kiválasztó táblák beállításra kerültek. Éppen ezért ez egy hasznos hely olyan kód elhelyezésére, amely végrehajtásának minden típusbeállító előkészítés után kell sorra kerülnie és amikor a dokumentum normál betűje az épp aktuális betű.

Az `\AtBeginDocument` hurok használata nem javasolt olyan kód esetén, amely bármilyen típusbeállítást végez, mivel a típusbeállítás eredménye megjósolhatatlan.

New description 1995/12/01

Az `\AtEndDocument` argumentumában megadott ⟨*kód*⟩ az `\end{document}` kód elején hajtódik végre, *mielőtt* az utolsó oldal elkészülne és bármilyen fennmaradó környezet feldolgozása előtt. Ha a ⟨*kód*⟩ egy részét ezen két eljárás után akarjuk végrehajtani, akkor a ⟨*kód*⟩ megfelelő részére be kell illesztenünk egy `\clearpage`-et.

Ezen parancsok ismételt használata engedélyezett: az argumentumok kódja a deklarációjuk sorrendjében kerül tárolásra (és később futtatásra).

`\AtBeginDvi {`⟨*speciális opciók*⟩`}`

New feature 1994/12/01

Ezek a parancsok egy doboz regiszterbe mentik azokat a dolgokat, amelyeket a `.dvi` fájlba írunk a dokumentum első oldalának ‘shipout’ elejére.

Ezt nem ajánlatos olyan esetben használni, amikor bármilyen típusbeállító részt akarunk a `.dvi` fájlhoz adni.

Ennek a parancsnak az ismételt használata is engedélyezett.

4.7 Opció feldolgozás

`\ProcessOptions`

Ez a parancs minden kiválasztott opcióra lefuttatja a ⟨*kód*⟩-ot.

Először leírjuk, hogy működik a `\ProcessOptions` a csomag fájlokban, aztán azt, miben különbözik az osztály fájlok esetén.

Hogy részleteiben megértsük, mit csinál a `\ProcessOptions` csomag fájlokban, ismernünk kell a *helyi* és *globális* opciók közötti kükönbséget.

- **Helyi opció**, melyet közvetlenül az adott csomaghoz határoztunk meg a következők bármelyikének ⟨*opciók*⟩ argumentumában:

  `\PassOptionsToPackage{`⟨*opciók*⟩`} \usepackage[`⟨*opciók*⟩`]`
  `\RequirePackage[`⟨*opciók*⟩`]`

- **Globális opció** minden más, amit a szerző a `\documentclass[`⟨*opciók*⟩`]` argumentumában meghatározott.

Például tegyük fel, hogy a dokumentum így kezdődik:

```
\documentclass[german,twocolumn]{article}
\usepackage{gerhardt}
```

miközben a `gerhardt` csomag az alábbi módon hívja a `fred` csomagot:

```
\PassOptionsToPackage{german,dvips,a4paper}{fred}
\RequirePackage[errorshow]{fred}
```

ekkor:

- `fred` helyi opciói `german`, `dvips`, `a4paper` és `errorshow`;
- `fred` egyetlen globális opciója `twocolumn`.

Ha a `\ProcessOptions`-t hívjuk, a következők történnek.

- *Először* minden opcióra, melyet a `\DeclareOption` eddig `fred.sty`-ban deklarált, megnézi, hogy az adott opció globális vagy helyi-e `fred` számára: ha igen, a megfelelő kód kerül lefuttatásra.

  Ez abban a sorrendben történik, mint ahogy az adott opciókat `fred.sty`-ban deklaráltuk.

- *Majd* minden megmaradó *helyi* opcióra a `\ds@`⟨*opció*⟩ parancs fut le ha ezt valahol (nem egy `\DeclareOption`-ben) definiáltuk; különben az ‘alapértelmezett opció kód’ hajtódik végre. Ha nincs deklarálva alapértelmezett opció kód, akkor hibaüzenetet kapunk.

  Ez abban a sorrendben történik, mint ahogy az adott opciókat megadtuk.

Ezen eljárásban a rendszer biztosítja, hogy egy opcióhoz rendelt kód legfeljebb egyszer hajtódik végre.

A példához visszatérve, ha `fred.sty` a következő alakú:

```
\DeclareOption{dvips}{\typeout{DVIPS}}
\DeclareOption{german}{\typeout{GERMAN}}
\DeclareOption{french}{\typeout{FRENCH}}
\DeclareOption*{\PackageWarning{fred}{Unknown ‘\CurrentOption’}}
\ProcessOptions\relax
```

akkor ezen dokumentum feldolgozásának az eredménye:

```
DVIPS
GERMAN
Package fred Warning: Unknown ‘a4paper’.
Package fred Warning: Unknown ‘errorshow’.
```

Jegyezzük meg a következőt:

- a `dvips` opcióhoz tartozó kód előbb hajtódik végre, mint a `german` opcióé, mivel ebben a sorrendben lettek `fred.sty`-ban deklarálva;
- a `german` opcióhoz tartozó kód csak egyszer hajtódik végre, mégpedig a deklarált opciók feldolgozása közben;
- az `a4paper` és `errorshow` opciók a `\DeclareOption*` által deklarált kód figyelmeztetését eredményezik (specifikálásuk sorrendjében), míg a `twocolumn` opció nem: ennek oka, hogy `twocolumn` globális opció.

A `\ProcessOptions` így dolgozik osztály fájlokban is, kivéve: *minden* opció helyi; és `\DeclareOption*` alapértelmezett értéke hiba helyett `\OptionNotUsed`.

New description 1995/12/01

Jegyezzük meg, hogy mivel `\ProcessOptions`-nek létezik `*`-alakja, helyesebb a nem-csillag alakot `\relax`-szel követni, ahogy az előző példákban láttuk, hiszen ez megakadályozza a szükségtelen előreolvasást és az esetlegesen megjelenő félrevezető hibaüzeneteket.

`\ProcessOptions*` `\@options`

Ez olyan, mint a `\ProcessOptions`, de a hívó parancsok által meghatározott sorrendben hajtja végre az opciókat, az osztály- vagy csomagbeli deklaráció sorrendje helyett. Csomag esetén ez azt jelenti, hogy a globális opciókat dolgozza fel először.

A LaTeX 2.09 `\@options` parancsát ugyanígy írták meg, hogy megkönnyítse a régi dokumentumstílusok LaTeX 2_ε osztály fájlokká való alakítását.

`\ExecuteOptions {`⟨*opció lista*⟩`}`

Ez az ⟨*opció lista*⟩ minden opciójára sorrendben végrehajtja a `\ds@`⟨*opció*⟩ parancsot (amennyiben ez a parancs nem definiált, azt az opciót csendben elhagyja).

Arra használható, hogy egy ‘alapértelmezett opció listát’ adjon közvetlenül a `\ProcessOptions` előtt. Például tegyük fel, hogy egy osztály fájlban be szeretnénk állítani a kívánt alapértelmezett formát: kétoldalas nyomtatás; 11pt betűméret; kéthasábos. Ezt így adhatjuk meg:

```
\ExecuteOptions{11pt,twoside,twocolumn}
```

4.8 Megbízható fájlparancsok

Ezek a parancsok a fájl inputtal foglalkoznak; biztosítják, hogy nem létező fájlok kérését felhasználóbarát módon kezelik.

`\IfFileExists {`⟨*fájl név*⟩`} {`⟨*igaz*⟩`} {`⟨*hamis*⟩`}`

Ha a fájl létezik, az ⟨*igaz*⟩ részben meghatározott kód hajtódik végre.

Ha a fájl nem létezik, a ⟨*hamis*⟩ részben meghatározott kód hajtódik végre.

Ez a parancs *nem* olvassa be a fájlt.

`\InputIfFileExists {`⟨*fájl név*⟩`} {`⟨*igaz*⟩`} {`⟨*hamis*⟩`}`

Ez beolvassa a fájlt, ha ⟨*fájl név*⟩ létezik, közvetlenül a beolvasás előtt az ⟨*igaz*⟩ részben levő kód végrehajtódik.

Ha a fájl nem létezik, a ⟨*hamis*⟩ részben meghatározott kód hajtódik végre.

A megvalósítás alapja `\IfFileExists`.

4.9 Hibák jelzése, stb

Ezen parancsokat mások által írt osztályok és csomagok használhatják hibajelzésre vagy a szerzők informálására.

`\ClassError {`⟨*osztály név*⟩`} {`⟨*hiba szöveg*⟩`} {`⟨*súgó szöveg*⟩`}`
`\PackageError {`⟨*csomag név*⟩`} {`⟨*hiba szöveg*⟩`} {`⟨*súgó szöveg*⟩`}`

Ezek hibaüzenetet adnak. Látható a ⟨*hiba szöveg*⟩ és megjelenik a `?` hiba prompt. Ha a felhasználó leüti a `h` billentyűt, kiíródik a ⟨*súgó szöveg*⟩.

A ⟨*hiba szöveg*⟩ és ⟨*súgó szöveg*⟩ belsejében: `\protect` használható egy parancs végrehajtásának megakadályozására; `\MessageBreak` sortörést, `\space` pedig egy szóközt ír.

Jegyezzük meg, hogy a ⟨*hiba szöveg*⟩ egy mondatvégi pontot fog tartalmazni, ezért mi ne tegyünk az argumentumba.

Például:

```
\newcommand{\foo}{FOO}
\PackageError{ethel}{%
   Your hovercraft is full of eels,\MessageBreak
   and \protect\foo\space is \foo
}{%
   Oh dear! Something's gone wrong.\MessageBreak
   \space \space Try typing \space <return>
   \space to proceed, ignoring \protect\foo.
}
```

a következő képernyőt eredményezi:

```
! Package ethel Error: Your hovercraft is full of eels,
(ethel)                and \foo is FOO.

See the ethel package documentation for explanation.
```

Ha a felhasználó `h`-t üt, ezt fogja látni:

```
Oh dear! Something's gone wrong.
  Try typing  <return>  to proceed, ignoring \foo.
```

`\ClassWarning` {⟨*osztály név*⟩} {⟨*figyelmeztető szöveg*⟩}
`\PackageWarning` {⟨*csomag név*⟩} {⟨*figyelmeztető szöveg*⟩}
`\ClassWarningNoLine` {⟨*osztály név*⟩} {⟨*figyelmeztető szöveg*⟩}
`\PackageWarningNoLine` {⟨*csomag név*⟩} {⟨*figyelmeztető szöveg*⟩}
`\ClassInfo` {⟨*osztály név*⟩} {⟨*tájékoztató szöveg*⟩}
`\PackageInfo` {⟨*csomag név*⟩} {⟨*tájékoztató szöveg*⟩}

A négy `Warning` parancs hasonló a hibaparancsokhoz, kivéve, hogy csak egy figyelmeztetést eredményeznek a képernyőn, hiba prompt nélkül.

Az első kettő, a `Warning` verziók a sor számát is kiírják, ahol a figyelmeztetés bekövetkezik, míg a második kettő, a `WarningNoLine` verziók nem.

A két `Info` parancs hasonló, kivéve, hogy csak az átírat fájlba naplózzák az információkat, a sor számával együtt. Ennek a kettőnek nincs `NoLine` verziója.

A ⟨*figyelmeztető szöveg*⟩ és ⟨*tájékoztató szöveg*⟩ belsejében: `\protect` használható egy parancs végrehajtásának megakadályozására; `\MessageBreak` sortörést, `\space` pedig egy szóközt ír.

Ezek szintén ne végződjenek pontra, mivel egy automatikusan hozzájuk adódik.

4.10 Parancsok definiálása

A LaTeX 2ε biztosít néhány plusz eljárást olyan parancsok (újra)definiálására, melyeket osztály és csomag fájlbeli használatra terveztek.

Ezen parancsok `*`-alakjai arra használhatóak, hogy a TeX feltételek szerint nem túl hosszú parancsokat definiáljunk. Ez olyan parancsok hibakeresésére használható, melyek argumentumai előreláthatólag nem tartalmaznak egész bekezdéseket.

New feature 1994/12/01

`\DeclareRobustCommand` {⟨*parancs*⟩} [⟨*szám*⟩] [⟨*alapértelmezés*⟩] {⟨*definíció*⟩}
`\DeclareRobustCommand*` {⟨*parancs*⟩} [⟨*szám*⟩] [⟨*alapértelmezés*⟩] {⟨*definíció*⟩}

Ez a parancs ugyanazokat az argumentumokat kapja, mint a `\newcommand`, de egy stabil parancsot deklarál, még ha a ⟨*definíció*⟩ valamely kódja gyenge is. Ezt a parancsot új stabil parancsok definiálására használhatjuk, vagy létező parancsok újradefiniálására és stabillá tételére. Ha egy parancsot újradefiniálunk, az átírat fájlba egy bejegyzés kerül.

Például, ha `\seq` a következőképpen van definiálva:

```
\DeclareRobustCommand{\seq}[2][n]{%
  \ifmmode
    #1_{1}\ldots#1_{#2}%
  \else
    \PackageWarning{fred}{You can't use \protect\seq\space in text}%
  \fi
}
```

Ekkor a `\seq` parancs mozgó argumentumokban használható, akkor is, ha `\ifmmode` nem, például:

```
\section{Stuff about sequences $\seq{x}$}
```

Azt is jegyezzük meg, hogy nincs szükség az `\ifmmode` elé `\relax`-et tenni a definíció elején; mivel a `\relax` rossz időben történő végrehajtása elleni védelme belsőleg biztosított.

`\CheckCommand` {⟨*parancs*⟩} [⟨*szám*⟩] [⟨*alapértelmezés*⟩] {⟨*definíció*⟩}
`\CheckCommand*` {⟨*parancs*⟩} [⟨*szám*⟩] [⟨*alapértelmezés*⟩] {⟨*definíció*⟩}

Ez ugyanazon argumentumokat kapja, mint a `\newcommand`, de ahelyett, hogy definiálná a ⟨*parancs*⟩-ot, egyszerűen csak ellenőrzi, hogy a ⟨*parancs*⟩ jelenlegi definíciója megegyezik-e a ⟨*definíció*⟩ által adottal. Ha a definíciók eltérnek, hibaüzenetet kapunk.

Ez a parancs a rendszer állapotának vizsgálatához hasznos, mielőtt a saját csomagunk elkezdi módosítani a parancsok definícióit. Különösen annak ellenőrzését teszi lehetővé, hogy más csomagok nem definiálták-e újra ugyanazt a parancsot.

4.11 Mozgó argumentumok

New description 1994/12/01

A mozgó argumentumok feldolgozása (azaz mozgatása) közbeni védelem beállítását újra megvalósították, ahogy az `.aux` fájlból más, például `.toc` fájlba történő írás eljárását is. Részletek a `ltdefns.dtx` fájlban találhatóak.

Reméljük, ezek a változtatások nem sok csomagot érintenek.

5 Különféle parancsok, stb

5.1 Elrendezési paraméterek

`\paperheight`
`\paperwidth`

Ezt a két paramétert rendszerint az osztály állítja be az alkalmazott papír méretére. A tényleges papírméretnek kell lennie, nem úgy, mint `\textwidth` és `\textheight`, melyek a margón belüli fő szövegrész méretei.

5.2 Betű változtatás

`\MakeUppercase {`⟨*szöveg*⟩`}`
`\MakeLowercase {`⟨*szöveg*⟩`}`

A TEX az `\uppercase` és `\lowercase` primitíveket biztosítja a szöveg betűinek megváltoztatására. Ezeket néha dokumentum osztályok használják, például csupa nagybetűsre állítani fejléceket.

New feature 1995/06/01

Sajnos ezen TEX primitívek nem változtatják meg például az `\ae` vagy `\aa` parancsok segítségével elérhető karaktereket. Hogy legyőzzük ezt a problémát, a LATEX a `\MakeUppercase` és `\MakeLowercase` parancsokat nyújtja.

Például:

`\uppercase{aBcD\ae\AA\ss\OE}`	ABCDæÅßŒ
`\lowercase{aBcD\ae\AA\ss\OE}`	abcdæÅßŒ
`\MakeUppercase{aBcD\ae\AA\ss\OE}`	ABCDÆÅSSŒ
`\MakeLowercase{aBcD\ae\AA\ss\OE}`	abcdæåßœ

A `\MakeUppercase` és `\MakeLowercase` parancsok önmagukban stabilak, de vannak mozgó argumentumaik.

Ezek a parancsok az `\uppercase` és `\lowercase` TEX primitíveket használják és így számos váratlan ‘sajátosságuk’ van. Megváltoztatják többek között a szöveg argumentumukban található összes betűt (kivéve a kontroll szekvenciák neveinek karaktereit): beleértve matematikai szöveget, környezeti neveket és címkeneveket.

Például:

```
\MakeUppercase{$x+y$ in \ref{foo}}
```

eredménye $X + Y$ és az alábbi figyelmeztetés:

```
LaTeX Warning: Reference ‘FOO’ on page ... undefined on ...
```

Hosszútávon inkább csupa nagybetűs betűtípust szeretnénk használni, mint a `\MakeUppercase`-hez hasonló parancsokat, de ez pillanatnyilag lehetetlen, mivel ilyen betűtípusok nem léteznek.

Ahhoz, hogy a nagy/kisbetűk lehetőleg jól működjenek, és hogy gondoskodjunk a helyes elválasztásról, az *kell*, hogy betűk megváltoztatására a LaTeX 2ε ugyanazon megadott táblákat használja egy dokumentumon belül. A használt tábla a `T1` betűkódoláshoz készült; ez megfelelően működik a hagyományos TeX betűtípusokkal minden Latin ábécében, de problémát okozhat más ábécék használata esetén.

New description 1995/12/01

5.3 Az 'openany' opció a 'book' osztályban

Az `openany` opció lehetővé teszi, hogy fejezet- és hasonló kezdetek bal oldali lapon jelenjenek meg. Eddig ez az opció csak `\chapter`-re és `\backmatter`-re volt hatással. Most befolyásolja `\part`-ot, `\frontmatter`-t és `\mainmatter`-t is.

New description 1996/06/01

5.4 Jobb, felhasználó által adott matematikai megjelenítő környezetek

`\ignorespacesafterend`

Tegyük fel, hogy olyan szöveg megjelenítéséhez szeretnénk környezetet definiálni, amely úgy számozott, mint egy egyenlet. Ennek egy egyszerű módja a következő:

New feature 1996/12/01

```
\newenvironment{texteqn}
  {\begin{equation}
     \begin{minipage}{0.9\linewidth}}
    {\end{minipage}
   \end{equation}}
```

Mégis, ha kipróbáljuk, talán észrevehetjük, hogy nem működik tökéletesen, ha egy bekezdés közepén használjuk, mivel megjelenik egy szavak közötti szóköz a környezet után, az első sor elején.

Létezik egy (nagyon hosszú nevű) parancs, ami kiküszöböli ezt a problémát; a következőképpen kell beszúrni:

```
\newenvironment{texteqn}
  {\begin{equation}
     \begin{minipage}{0.9\linewidth}}
    {\end{minipage}
   \end{equation}
   \ignorespacesafterend}
```

6 LaTeX 2.09 osztályok és csomagok korszerűsítése

Ez a fejezet olyan változtatásokról ír, melyekre szükségünk lehet, amikor egy létező LaTeX stílust csomaggá vagy osztállyá bővítünk, de először legyünk bizakodóak.

Sok meglévő stílus fájl bármilyen változtatás nélkül működni fog LaTeX 2_ε alatt. Amikor minden rendesen fut, kérjük, tegyen egy megjegyzést az újonnan elkészült csomag vagy osztály fájlba, jelezve ezzel, hogy az új LaTeX szabvánnyal működik; majd adja tovább.

6.1 Először próbáljuk ki!

Az első teendőnk a saját stílus kipróbálása 'kompatibilitási mód'-ban. Az egyetlen változtatás, amit esetleg ehhez el kell végeznünk, a fájl kiterjesztésének módosítása `.cls`-re: ez csak akkor szükséges, ha a fájlt fő dokumentum stílusként használtuk. Ezután, más módosítások nélkül kell futtatnunk a LaTeX 2_ε-t egy olyan dokumentumon, amely használja ezt a fájlt. Ez feltételezi, hogy létezik egy megfelelő fájl gyűjtemény, amely teszteli a stílus fájl által nyújtott összes funkciót. (Ha még nincs ilyen, itt az idő, hogy készítsünk egyet!)

Most úgy kell módosítanunk a teszt dokumentum fájlokat, hogy LaTeX 2_ε dokumentumok legyenek: ezen művelet részleteiért lásd a *LaTeX 2_ε for Authors*-t, aztán újra tesztelni kell azokat. Megtörtént a teszt dokumentumok LaTeX 2_ε natív és LaTeX 2.09 kompatibilitási módban való próbája.

6.2 Hibaelhárítás

Ha egy fájl nem működik a LaTeX 2_ε-vel, annak valószínűleg két oka lehet.

- A LaTeX most tartalmaz egy stabil, jól definiált tervezői felületet betűk kiválasztásához, amely sokban különbözik a LaTeX 2.09 belsejétől.
- A saját fájlunk talán használ olyan LaTeX 2.09 belső parancsokat, amelyeket megváltoztattak vagy esetleg töröltek.

Hibakeresés közben esetleg több információra van szükségünk, mint amennyit a LaTeX 2_ε rendszerint megjelenít. Ezt úgy érjük el, hogy az `errorcontextlines` számláló értékét az alapértelmezett -1 helyett egy sokkal nagyobb értékre, például 999-re állítjuk.

6.3 Betű parancsok

Néhány betű és méret parancsot most a LaTeX kernel helyett a dokumentum osztály definiál. Ha egy LaTeX 2.09 dokumentum stílust olyan osztályra frissítünk, amely nem tölt be szabványos osztályt, akkor esetleg szükség lehet ezen parancsokhoz definíciók hozzáadására.

`\rm \sf \tt \bf \it \sl \sc`

Ezen rövid alakú betűkiválasztó paracsok egyikét sem a LaTeX 2_ε kernel definiálja. Ezeket minden szabványos osztály fájl definiálja.

Ha saját osztály fájlban szeretnénk definiálni valamelyiket, több elfogadható út közül választhatunk.

Egy lehetséges definíció a következő:

```
\newcommand{\rm}{\rmfamily}
...
\newcommand{\sc}{\scshape}
```

Ez ortogonálissá teszi a betű parancsokat; például `{\bf\it szöveg}` félkövér, dőlt betűt eredményez, valahogy így: ***szöveg***. Ettől viszont matematikai módban használva hibát okoznak.

Definiálásra egy másik lehetőség:

```
\DeclareOldFontCommand{\rm}{\rmfamily}{\mathrm}
...
\DeclareOldFontCommand{\sc}{\scshape}{\mathsc}
```

Ettől `\rm` szöveg módban ugyanúgy fog viselkedni, mint `\rmfamily` (lásd fent) és matematikai módban `\rm` kiválasztja a `\mathrm` matematikai ábécét.

Ekkor `${\rm math} = X + 1$` eredménye ‘$\mathrm{math} = X + 1$’ lesz.

Ha nem akarjuk, hogy a betű kiválasztás ortogonális legyen, a szabványos osztályokat kell követnünk és így definiálni:

```
\DeclareOldFontCommand{\rm}{\normalfont\rmfamily}{\mathrm}
...
\DeclareOldFontCommand{\sc}{\normalfont\scshape}{\mathsc}
```

Ez például azt jelenti, hogy `{\bf\it szöveg}` eredménye közepes súlyú (inkább, mint félkövér), dőlt, így: *szöveg*.

`\normalsize`
`\@normalsize`

A `\@normalsize` parancs megmaradt a LaTeX 2.09 csomagok kompatibilitása érdekében, melyek esetleg használták az értékét; de egy osztályban való újradefiniálásának nem lesz hatása, mivel mindig visszaáll, hogy ugyanazt jelentse, mint `\normalsize`.

Ez azt jelenti, hogy az osztályoknak most `\@normalsize` újradefiniálása helyett `\normalsize` újradefiniálása *szükséges*; például (meglehetősen hiányosan):

```
\renewcommand{\normalsize}{\fontsize{10}{12}\selectfont}
```

Jegyezzük meg, hogy a LaTeX kernel úgy definiálja `\normalsize`-t, hogy az egy hibaüzenet.

```
\tiny \footnotesize \small \large
\Large \LARGE \huge \Huge
```

Ezen ‘szabványos’ méretváltoztató parancsok egyike sem a kernelben definiált: bármelyiket osztály fájlban kell definiálnunk, ha szükségünk van rá. Mindegyiket a szabványos osztályok definiálják.

Ez azt jelenti, hogy `\renewcommand` használandó `\normalsize` esetén, a többi méretváltoztató parancsnál pedig `\newcommand`.

6.4 Elavult parancsok

Néhány csomag nem működik a LaTeX 2ε-vel, hiszen egy belső LaTeX parancstól függenek, amelyet sosem támogatott és most megváltozott vagy eltávolították.

Sok esetben lesz egy stabil, magas szintű megvalósítása annak, ami előzőleg alacsony szintű parancsokat igényelt. Tanulmányozza át a 4 fejezetet, hogy megtudja, használhat-e LaTeX 2ε osztály és csomag író parancsokat.

Természetesen ha a csomagja vagy osztálya újradefiniált bizonyos kernel parancsokat (azaz a `latex.tex`, `slitex.tex`, `lfonts.tex`, `sfonts.tex` fájlokban definiáltakat), akkor szüksége lesz alapos ellenőrzésre az új kernellel szemben, hátha megváltozott a megvalósítás vagy már nem is létetik a parancs a LaTeX 2ε kernelben.

A LaTeX 2.09 túl sok belső parancsa került feldolgozásra vagy tűnt el ahhoz, hogy itt mindet felsoroljuk. Ellenőrizze le azokat, amelyeket használt vagy módosított.

Néhány nagyon fontos, már nem támogatott parancsot azonban fel kell sorolnunk.

```
\tenrm \elvrm \twlrm ...
\tenbf \elvbf \twlbf ...
\tensf \elvsf \twlsf ...
   ⋮
```

A (körülbelül) hetven ilyen alakú belső parancs már nem definiált. Amennyiben saját osztály vagy csomag fájlja használja valamelyiket, *kérjük*, a *LaTeX 2ε Font Selection* útmutatásai alapján cserélje le új betű parancsokra.

Például a `\twlsf` parancs a következőre cserélhető:

```
\fontsize{12}{14}\normalfont\sffamily\selectfont
```

Másik lehetőség a `rawfonts` csomag használata, ahogy ezt a *LaTeX 2ε for Authors* leírja.

Azt is jegyezzük meg, hogy a LaTeX 2.09 legtöbb előre betöltött betűje már nem töltődik be előre.

```
\vpt \vipt \viipt ...
```

Ezek voltak a LaTeX 2.09 belső méretkiválasztó parancsai. (A LaTeX 2.09 kompatibilitási módban még mindig használhatóak.) Kérjük, helyettük a `\fontsize` parancsot használja: a *LaTeX 2ε Font Selection* részletezi.

A következő például `\vpt` helyett írható:

```
\fontsize{5}{6}\normalfont\selectfont
```

`\prm`, `\pbf`, `\ppounds`, `\pLaTeX` ...

A LaTeX 2.09 számos `\p` kezdetű parancsot használt, hogy 'védett' parancsokat nyújtson. Például `\LaTeX` úgy volt definiálva, mint `\protect\pLaTeX`, `\pLaTeX` definíciója pedig a LaTeX logót eredményezte. Ettől lett `\LaTeX` stabil, holott `\pLaTeX` nem volt az.

Ezeket a parancsokat most `\DeclareRobustCommand` segítgégével újra megvalósították (leírva a 4.10 fejezetben). Ha a csomagja újradefiniált egy `\p`-parancsot, el kell távolítania az újradefiniálást és a `\DeclareRobustCommand` használatával újradefiniálnia a nem-`\p` parancsot.

`\footheight`
`\@maxsep`
`\@dblmaxsep`

A LaTeX 2_ε nem használja ezen paramétereket, ezért ezeket eltávolították, kivéve a LaTeX 2.09 kompatibilitási módban. Osztályok már nem állíthatják be ezeket.

References

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[2] Donald E. Knuth. *The TeXbook*. Addison-Wesley, Reading, Massachusetts, 1986. Revised to cover TeX3, 1991.

[3] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

LaTeX 2ε Összefoglaló lap: régi stílusok frissítése

A *LaTeX 2ε for Class and Package Writers* fejezet-hivatkozásai.

1. Osztály vagy csomag legyen? Lásd a 2.3 fejezetet a kérdés megválaszolásához.
2. Ha egy másik stílus fájlt használ, akkor szükség lesz annak a frissített változatára. Lásd a 2.7.1 fejezetet más osztály és csomag fájlok betöltéséről szóló információkért.
3. Próbáljuk ki: lásd a 6.1 fejezetet.
4. Működik? Kiváló, de talán még mindig van néhány változtatásra szoruló dolog, hogy a fájlból jól struktúrált LaTeX 2ε fájl legyen, mely stabil és hordozható. Tehát olvassa el a 2, különösen a 2.7 fejezetet. Hasznos példák találhatóak még a 3 fejezetben.

   Ha a fájl új betűket, jeleket vagy betűváltoztató parancsokat állít be, ajánlott még a *LaTeX 2ε Font Selection* elolvasása.
5. Nem működik? Három lehetőség van:

   - hibaüzenetet kapunk a fájl beolvasása közben;
   - hibaüzenetet kapunk teszt dokumentumok feldolgozása közben;
   - nincs hibaüzenet, de a kimenet nem a várakozásnak megfelelő.

   Ne felejse el az utolsó lehetőséget megvizsgálni.

   Ha eljutott idáig, akkor szüksége lesz a 6 fejezet elolvasására, hogy megtalálja a fájl működésre bírásához vezető megoldást.