Magyar nyelvű műszaki–tudományos tipográfia

– Diplomaterv –

Szabó Péter

Konzulensek:
Dr. Kollár István, egyetemi tanár
BME Méréstechnika és Információs Rendszerek Tanszék
dr. Wettl Ferenc, docens
BME Algebra Tanszék

Budapest, 2004. május

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
MÉRÉSTECHNIKA ÉS INFORMÁCIÓS RENDSZEREK TANSZÉK

DIPLOMATERV FELADAT

Szabó Péter
szigorló mérnök-informatikus hallgató részére
(nappali műszaki-informatika szak)

...

A feladat szövege

Magyar nyelvű műszaki–tudományos tipográfia

A magyar nyelven megjelenő természettudományos és műszaki könyvek és folyóiratok legtöbbjét angol nyelvterületen és angol nyelvű művek készítésére kifejlesztett, és csak részben honosított programokkal szedik. Ezért időszerű feladat az angol és a magyar tipográfia jellemzőinek összehasonlítása, a különbségek elemzése, és a magyar specialitások bevitele a rendszerekbe.

- Tekintse át, hogy a kézi szedés idejéből származó magyar tipográfiai szabályok és hagyományok, valamint a magyar nyelvnek a szedés közben figyelembe veendő sajátosságai mely pontokon térnek el az angolszászokétól.
- Foglalja össze a magyar nyelvű szedésre használt programok főbb tulajdonságait és korlátaikat.
- Hasonlítson össze két-három – a műszaki könyv- és folyóirat-kiadásban használt – programot annak alapján, hogy a magyar tipográfia mely elemei valósíthatóak meg bennük nagyobb nehézség nélkül, és hogyan.
- Vizsgáljon meg e programok közül egyet részletesen, és találjon megoldást a nehezebb tipográfiai szabályok betartásának megkönnyítésére. A javasolt megoldás az adott programhoz olyan dokumentumsablont és/vagy program-kiegészítéseket tartalmazzon, amelyek az eddigieknél egyszerűbben használható eszközöket biztosítanak a tipográfiai szempontból képzetlen felhasználó számára is magyar nyelvű dokumentumok magasabb színvonalú szedésében.
- Elemezze az angol nyelvű alapprogram és a magyar nyelvű használat ellentmondásait. Térjen ki a kiegészítők használatára is (például helyesírás-elemző).
- Elemezze a választott megoldást és a továbbfejlesztés lehetőségeit.

Dr. Kollár István
egyetemi tanár

Nyilatkozat

Alulírott, **Szabó Péter**, a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, és a diplomatervben csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

. . . . . .

Szabó Péter

Tartalomjegyzék

1. Bevezetés **7**
1.1. A dolgozat felépítése . . . 7
1.2. A feladat értelmezése . . . 7
1.3. A kifejlesztett szoftverek . . . 10
1.4. Fogalomtár . . . 11
1.5. A vizsgált programok . . . 14

2. A magyar nyelvi sajátosságok elemzése **17**
2.1. Betűk, szavak . . . 17
2.2. A folyószöveg egyéb elemei . . . 22
2.3. A folyószövegen kívüli részek . . . 27
2.4. A vizsgált programok összehasonlítása . . . 29

3. A LaTeX magyarítása **31**
3.1. Dokumentáció és felhasználói felület . . . 33
3.2. A LaTeX babel csomagjának magyar.ldf modulja . . . 34
3.3. Automatikus szóelválasztás . . . 57
3.4. Tárgymutató (index) készítése . . . 61
3.5. Irodalomjegyzék a magyar tipográfia szerint . . . 67
3.6. Helyesírás-ellenőrzés . . . 74

4. Összefoglalás **80**
4.1. Értékelés . . . 80
4.2. Továbbfejlesztési lehetőségek . . . 86

Ábrák jegyzéke

1. A *gy* betű két karakterrel és a *gy* ligatúra . . . 19
2. Útmutató a stílus alapján történő betűválasztáshoz . . . 21
3. A formátumgenerálás, az elválasztás és a babel kapcsolata . . . 57

Táblázatok jegyzéke

1. A kifejlesztett szoftverek . . . 11
2. Magánhangzóink helye a Latin-1, Latin-2 és Unicode kódolásokban . . . 17
3. Az vizsgált programok magyar szolgáltatásai . . . 30
4. A magyar elválasztási minták adatai . . . 58
5. Példák a LaTeX magyar elválasztási mintáival . . . 58
6. A BibTeX ékezetei és különleges betűi . . . 71
7. A helyesírás-ellenőrzés sebessége és memóriaigénye . . . 79

Abstract

Every language and national culture has its own typographical traditions. These traditions describe the visual layout of the individual components of the written works, and, on the other hand they suggest methods and technologies to produce this layout. Tradition helps the makers of books and other works so the final product will more comprehensible, legible, perspicuous, searchable for the readers speaking the given language.

In the time of hand-composition it was the task of the type-setters to obey the traditions: they placed the letters (or the matrices containing letters or other characters) to the page manually, with the most perfect distance between them. Nowadays, when books, periodicals and other works are produced using computers, type-setters use some kind of document-preparation programs to lay out he components of the page. These programs help the type-setter a lot, because they automate some of the human work: the most important jobs are the breaking of paragraphs to lines, justification and page numbering.

Some of the tasks can be automated only with considering the typographical traditions of the target language, these are e.g. automatic hyphenation, sorting of indexes, formatting, sorting and typesetting of the bibliography, formatting of footnotes, numbering of structural elements and references to them (with arabic, roman and spelled-out numbers), and the formatting of the numbering. The neglecting of the traditions often violates the spelling too, or it produces a work hard to comprehend and alien to the reader in word order and intonation markup. So it is vital that the producers of writings work with software having typographical knowledge integrated in an easy to use way. With some works a professional typographer or typesetter is not affordable, and since the author doesn't have enough background knowledge to make good typographical decisions, the programs must be designed so that the defaults are acceptable with respect to the national typographical traditions.

This thesis reviews the Hungarian typographical traditions, and examines their implementation status in some programs. Since presenting the full range of traditions is en enormous task, the area of investigation is reduced to technical and scientific works, and within this, the emphasis is on trade books and periodicals, and most of the problems solves are related to such works. The writings in the Hungarian language are usually produced with partially localized versions of software products of international companies, targeted for English markets. That's because programs with similar goals, features and usability haven't been released so far. The English programs are only partially localized: although they ship with Hunguarian user interface and documention, fonts containing all the Hungarian letters, a spell checker and hyphenation software for Hungarian (not all programs have all these features), they lack much typographical knowledge. The manual application of this knownledge is an inconvenience or a difficulty even for professional type-setters, and the non-professional user is completely helpless how and where to change the defaults.

The aim of the thesis is to present the recent work of its author to localize the LaTeX typesetting system to the Hungarian language. All the style files and auxilary programs are presented, with which now the typesetting of Hungarian text can be carried out more confortably and precisely. The availability of Hungarian documentation and the question of the Hungarian user interface is presented, and solutions are given for the typesetting of main text following the Hungarian traditions, for automatic hyphenation, for sorting and formatting of the index, for typesetting the bibliography and for spell checking. Some aspects are left without solution—the aim of the thesis is to solve the most important problems in the best possible, most automated way.

Kivonat

Minden nyelvnek és nemzeti kultúrának megvannak a maga tipográfiai hagyományai. Ezen hagyományok egyrészt meghatározzák az adott nyelven készülő írásos művek egyes formai elemeinek kinézetét, másrészt pedig egy ezeknek megfelelő készítéséhez ajánlott módszereket és technológiákat javasolnak. A hagyományok segítik a könyvek és egyéb művek készítőit, hogy minél érthetőbb, olvashatóbb, áttekinthetőbb, kereshetőbb és szebb művet készítsenek az adott nyelvet értő olvasók számára.

A kézi szedés idejében a hagyományok hű követése elsősorban a szedő feladata volt: a készülő oldalra saját kezűleg helyezte el a betűket és egyéb jeleket, egymástól épp a megfelelő távolságra. Manapság, amikor az könyvek, folyóiratok és egyéb publikációk számítógéppel készülnek, az egyes elemek elhelyezését valamilyen dokumentum-előkészítő programmal kell a szedőnek megoldania. E programok akkor jelentenek igazi segítséget a szedőnek, ha munkájának nagy részét magukra vállalják; ezek közül a bekezdések sorokra tördelése, a sorkizárás és az oldalszámozás a legjelentősebbek.

Egyes feladatok csak az adott nyelv tipográfiai hagyományainak figyelembe vételével automatizálhatók, ilyen például a szóelválasztás, a mutatók rendezése, az irodalomjegyzék rendezése, formázása és szedése, a lábjegyzetek formázása, a struktrulális egységek és a rájuk való utalások számozása (arab és római számokkal vagy betűvel kiírt számnevekkel) és a számozás formázása. A hagyományoktól való eltérés sok esetben sérti a helyesírást, vagy az adott nyelv szórendjétől, hangsúlyozásától idegen, az olvasó számára nehezen befogadható művet eredményez. Épp ezért igen fontos, hogy a készítők olyan szoftverrel dolgozhassanak, melybe az adott nyelv tipográfiai tudása könnyen felhasználható módon bele van építve. Egyes esetekben anyagi okokból nem bízhatók a tipográfiai és szedési munkák szakértőre, és a mivel szerzőnek gyakran nincs meg a megfelelő előképzettsége, hogy jó tipográfiai döntéseket hozzon, ezért a programokat úgy kell felkészíteni, hogy a felkínált alapértelmezés az adott nyelv hagyományai szerint elfogadható legyen.

A dolgozat azt vizsgálja, hogy melyek a magyar tipográfiai hagyományok, ezekből mi van beépítve az egyes programokba. Mivel a teljes hagyományanyag áttekintése óriási feladat, vizsgálódási területét a műszaki–tudományos művekre, ezen belül is főleg szakkönyvekre és folyóiratokra helyezi a hangsúlyt, az ezek készítésekor felmerülő problémákat igyekszik orvosolni. A magyar művek általában nemzetközi cégek angolszász piacra szánt szoftvertermékeinek a gyártó által részben magyarított változataival készülnek. Ennek oka, hogy hasonló célú, tudásban és használhatóságban versenyképes magyar termékek nem születtek. Az angol programok csak részben vannak lokalizálva: bár általában magyar dokumentációval, felhasználói felülettel, a magyar betűket tartalmazó betűtípusokkal, magyar helyesírás-ellenőrzővel és szóelválasztóval kerülnek forgalomba (bár ezek némelyike hiányzik az egyes programokból), sok egyéb tipográfiai tudás hiányzik belőlük. E tudás kézi alkalmazása a profi szedőnek is kényelmetlenséget vagy nehézséget jelent, a képzetlen felhasználó pedig gyakran tanácstalan, hogy mit és hogyan kell változtatni az alapértelmezéseken.

A dolgozat a LaTeX szedőprogram magyarítását tűzi ki célul: bemutatja a diplomatervezés során elkészített stílusfájlokat és segédprogramokat, melyekkel magyar nyelvű szöveg LaTeX-es szedése az eddigieknél kényelmesebben és pontosabban valósítható meg. Részletesen ír a LaTeX magyar nyelvű dokumentációjáról, a LaTeX felhasználói felülete dokumentációjának kérdéséről, és megoldást ad a főszöveg magyar hagyományok szerinti szedésére, az automatikus szóelválasztásra, a tárgymutató rendezésére és formázására, az irodalomjegyzék szedésére és a helyesírás-ellenőrzésre. A dolgozat nem kínál megoldást az összes magyar hagyomány követésére, de a legfontosabbakat a lehető legjobban, minél inkább automatizálva próbálja megvalósítani.

1. Bevezetés

1.1. A dolgozat felépítése

Az 1.2. alszakasz értelmezi a feladatot és szól arról is, miért időszerű a dokumentum-előkészítés magyar vonatkozásaival foglalkozni. Az 1.3. alszakasz röviden összefoglalja a kitűzött célokat, a megoldandó feladatokat. Az is ebből az alszakaszból derül ki, hogy a dolgozatban említett számos szoftver közül pontosan melyek e sorok írójának a diplomatervezési gyakorlat során elkészített saját munkái. Az 1.4. alszakasz tisztázza a dolgozatban előforduló legfontosabb fogalmak jelentését. Az összehasonlításra kerülő, dokumentum-előkészítésre használatos programokat az 1.5. alszakasz mutatja be.

A 2. szakasz ismerteti a diplomatervezés során végzett kutató és elemző munkát. A 2.1. alszakasztól kezdődően részletes kifejtésre kerülnek a magyar nyelv azon helyesírási–tipográfiai szabályai, melyek eltérnek az angolszász hagyományoktól. A 2.4. alszakaszban található az 1.5. alszakaszban bemutatott programok összehasonlítása a szakasz korábbi részeiben ismertetett szempontok szerint.

A fenti programok közül e sorok írója a LaTeX magyarítását tűzte ki célul. Az ennek során felmerülő tervezési döntéseket és ezek indoklását, továbbá az implementációs nehézségeket és az ezekre adott megoldást ismerteti a 3. szakasz. A 3.1. alszakaszban a dokumentáció és a felhasználói felülettel, a 3.2. alszakasz a LaTeX-ben szedett főszöveg és a generált formai elemekkel, a 3.3. alszakasz az automatikus szóelválasztással a 3.4. alszakasz a tárgymutatóval, a 3.5. alszakasz az irodalomjegyzékkel, a 3.6. alszakasz pedig a helyesírás-ellenőrzéssel foglalkozik. Az ebben a fejezetben ismertetett szoftverek nagy része a diplomatervezés során készült stílusfájl, program vagy segédprogram.

A 4.1. alszakasz elemzi az elkészült szoftvereket és értékeli az elvégzett munkát. A dolgozat szakmai része a a 4.2. alszakaszban leírt továbbfejlesztési lehetőségekkel zárul.

1.2. A feladat értelmezése

Egy írott műnek, legyen az könyv, folyóirat vagy másfajta dokumentum, elsősorban az olvasói igényeket kell szolgálnia. Feltéve, hogy a mű eljut az olvasójához, a legfontosabb igények az érthetőség, az olvashatóság, az áttekinthetőség, a kereshetőség és a szépség. Ezek lehető legszínvonalasabb biztosításán a szerző, a tipográfus, a szedő, a szerkesztő és még sokan mások együtt fáradoznak. Minden nyelv és minden kultúra más hagyományokkal rendelkezik a könyvnyomtatás, a folyóiratok, a sajtótermékek és egyéb dokumentumok terén. E hagyományok részben szervezési jellegűek (pl. milyen munkakörök találhatók meg egy szerkesztőségben), részben pedig az elkészült műre vonatkoznak, annak kötésére, formájára, kinézetére, szövegezésére, felépítésére stb. vannak hatással. E hagyományok sosem öncélúak, hanem elődeink és kortársaink dokumentum-előkészítési munkája során felgyűlt tapasztalat letisztult összefoglalásai, és egytől egyig valamely olvasói igény jobb kielégítését szolgálják a kész műben, vagy a mű elkészítésének folyamatát teszik hatékonyabbá. A hagyományokat követve tehát általában a magyar olvasók számára érthetőbb, olvashatóbb, áttekinthetőbb, kereshetőbb és szebb művek hozhatók létre, mint a hagyományok elvetésével vagy figyelmen kívül hagyásával, és a hagyomány segíti a művek gyorsabb, kevesebb gonddal történő elkészülését is. A hagyomány tehát a dokumentum-előkészítés egyes területeinek (pl. helyesírás, tipográfia, nyomdászat, szerkesztési ismeretek) tudás- és módszeranyaga, melynek különböző részeit a dokumentumok készítői és olvasói elosztottan birtokolják.

Az egyes kultúrák és nyelvterületek hagyományai részben történeti, részben praktikus okokból térnek el egymástól. A praktikus okok közé tartozik például az, hogy az írásje-

lek a mondat felolvasásának hangsúlyozását követik, ezért az írásjelek használata eltér a különböző nyelvekben, továbbá hogy a könyv formai elemei is az adott nyelv szórendje szerint készülnek (tehát például magyar könyvben *Ábra 5* helyett *5. ábrá*nak kell szerelnie). Nem adható viszont gyakorlatias magyarázat például arra, hogy miért kell egyes földrajzi neveket kötőjellel, egybe- vagy különírni, vagy hogy miért szükséges a *j* hang kétféle betűvel való megkülönböztetése, de arra sem, miért nem szabad magyar vagy francia nyelvű művekben a pont után szedett szóköz szélességét megnövelni (angolszász tipográfiában épp hogy ez a kívánatos). A hagyomány ezen elemei rögzültek a századok során, és közvetlen értelmük vagy hasznuk ugyan nem látszik, de a tipográfia és a helyesírás főbb pontjaiban megszülető egység az olvasó számára azt az előnyt szüli, hogy nem kell minden egyes mű befogadásához új, tiszavirág-életű olvasási szokásokat kialakítania. A hagyományhoz bizonyos mértékben tehát akkor is ajánlott ragaszkodni, ha azt a dokumentum készítői nem tartják indokoltnak, mivel a sok egyéni változtatás csak nehezítené az olvasó dolgát.

A diplomatervezés során végzett kutatás összegyűjti és rendszerezi a magyar tipográfiai hagyományok azon elemeit, melyek előtérbe kerülnek egy műszaki vagy tudományos mű, elsősorban könyv vagy folyóirat készítésekor. (Nem foglalkozik a hagyományok értékelésével, indoklásával vagy történeti kialakulásával: egyedül az érdekli, hogy egy hagyomány követését miképpen lehet automatizálni.) Ezen belül is az angolszász hagyományoktól való eltérésekre koncentrál, mivel a számítógépes dokumentum-előkészítés során angol nyelvű felhasználásra és/vagy angol nyelvű szöveg feldolgozására készült programok használatosak magyar nyelvterületen. Valójában a magyar nyelv nincs különleges helyzetben e téren: mivel ilyen programot írni nehéz és több területen nagy szaktudást igénylő feladat, ezért csak nagyobb, nemzetközi cégek tudtak a felhasználók (beleértve a könyvkiadás egyes területein dolgozó szakembereket és a képzetlen felhasználókat is) széles köre számára használható, átfogó megoldást nyújtó programokat piacra dobni. A világ különböző részein, de főleg a latin betűs írású nyelvterületeken általában ugyanannak a néhány programnak angolról az adott nyelvre lefordított, „lokalizált” változatait használják. Egyes programok lokalizálása néha kimerül a menürendszer, a dialógusablakok és a dokumentáció lefordításában, de a dokumentum-előkészítésre használt programok esetén ennél jóval többre van (lenne) szükség: arra, hogy a program lokalizált változatával az adott nyelv hagyományainak megfelelő művek készülhessenek – az olvasót nem érdekli, hogy a szerző számítógépén milyen nyelven jelent meg a *File* menü, de fontos számára az, hogy a mű formai elemei a hagyományoknak megfelelő, megszokott módon készüljenek. E téren a legtöbb program lokalizációja biztosítja, hogy az adott nyelv összes betűje megjeleníthető legyen, továbbá bizonyos feliratok (pl. „Tartalomjegyzék”) az adott nyelven kerüljenek be a dokumentumba, és legyen a programban helyesírás-ellenőrzés és automatikus szóelválasztás az adott nyelvhez. Mindezek csak a hagyományok egy töredékének érvényre jutását biztosítják, és általában elmondható, hogy a tipográfusnak, a szerzőnek és a szedőnek sok területen kézzel kell biztosítania, hogy a hagyományok érvényesüljenek – annak ellenére, hogy ebből sok minden automatizálható lenne, ha a program készítői között valaki rendelkezne az adott nyelvre vonatkozó ismeretekkel.

Mindezek fényében a feladat indokoltsága nyilvánvaló. Jelenleg nem létezik olyan program, amellyel magyar a tipográfiai hagyományok feletti tudatos és éber őrködés nélkül létrehozható lenne ezeknek megfelelő dokumentum. Emiatt fölösleges teher nehezedik a szerzőkre és a szedőkre is, továbbá a kézi javítások nehézkessége és a hagyomány pontos ismeretének szükségessége sokakat visszatart attól, hogy tipográfiai szempontból minőségi kimenet igényével üljenek le művüket elkészíteni. Ezen csak a programok tipográfiai tudásának lokalizálása segíthet, amely egyrészt a stílusfájlok az adott nyelvhez való iga-

zítását jelenti, másrészt, szükség esetén programozást, tehát az alapprogram módosítását vagy kiegészítését és segédprogramok készítését is magába foglalja. A feladat időszerű is, mert manapság a számítógépek már elég gyorsak ahhoz, hogy a dokumentum-előkészítés egyes műveleteire a felhasználónak ne kelljen várnia, hanem saját – emberi – tempójában, folyamatosan dolgozhasson. Rögzültek továbbá a művek készítésének forgatókönyvei: a készítőknek egyrészt megvan a megfelelő rutinjuk a számítógép használatához, másrészt nagy tudású, kiforrott programok segítik őket, attól eltekintve, hogy e programok a magyar sajátosságoknak csak egy kis részét ismerik. A diplomatervezés keretében épp ennek az űrnek a betöltése, vagy legalábbis szűkítése a cél.

A diplomaterv főleg ilyen automatizálható elemekkel foglalkozik, és megvizsgálja, hogy ezek biztosítása milyen nehézséget okoz egy mai elterjedt dokumentumszerkesztő, kiadványszerkesztő illetve szedőprogram esetén. Ezek közül a LaTeX teljesebb magyarítását tűzi ki célul, és stílusfájlok és kiegészítő segédprogramok fejlesztését tervezi, melyek a hagyomány összegyűjtött elemei közül minél többet automatikusan érvényre juttatnak. Az automatikus itt azt jelenti, hogy egy szabály alkalmazására a felhasználónak nem kell a szabály összes előfordulási helyén külön odafigyelnie, hanem elég, ha csak egyszer rendelkezik a szabály betartásáról, és a gép a legtöbb helyen ezt megteszi – néhány nehezen automatizálható kivételes esettől eltekintve, ahol továbbra is kézi beavatkozás szükséges. A diplomaterv az egyes problémákra a lehető legjobb megoldást próbája nyújtani a mű olvasóinak és készítőinek érdekeit figyelembe véve.

Szót kell ejteni a tipográfiai hagyományok változásáról. Ez fellép egyrészt egy új művészeti stílus megjelenésekor, másrészt a technika fejlődésével. Az előbbire példa a betűtípusok geometriai jellemzőinek korszakonkénti változása (lásd például a 2. ábrán), az utóbbira pedig a betűk néhány százalékos nyújtásának megjelenése a tömbös (sorkizárt) szedésben a szóközök szélesítése helyett. A huszadik század második felében a polgári felhasználók számára is elérhető számítógépek tették lehetővé, hogy a szerző és a szedő munkája átalakuljon, egyszerűsödjön. A fejlődéssel a hagyomány sok eleme elavulttá vált: például a tárgymutató-készítés [6, 213. oldal]-on leírt cédulás módszerénél ma már jóval egyszerűbb a dokumentum elektronikusan szerkesztett változatában megjelölni (és esetleg kiegészíteni, egységesíteni) a tárgymutatóba kerülő szavakat, a gép ezután automatikusan rendezi és szedi a tárgymutatót. A diplomaterv nem foglalkozik a kézi szedés idejéből származó technológiai hagyományokkal, hanem egy konkrét dokumentum-előkészítő számítógép-programot feltételez, egy mű készül, és olyan megoldást keres a magyar tipográfiai hagyományok követésére, amely jól illeszkedik az adott program lehetőségei közé, és a felhasználók számára természetes módon egészíti ki a programot. A legtöbb esetben olyan kiegészítő kívánatos, amelyről a képzetlen felhasználónak nem is kell tudnia, mégis helyes magyar dokumentum készít. (Ez azért is fontos, mert a kis költségvetésű művek készítői nem alkalmaznak profi tipográfust és szedőt.) Elképzelhető, hogy az elkészített megoldás nem fog működni az adott program egy későbbi verziójával, vagy nem felel meg a megváltozott tipográfiai igényeknek, ezért olyan fejlesztői dokumentáció megírása is cél, amely alapján mások is képesek lesznek a szoftvert módosítani.

A diplomaterv nem foglalkozik külön az elektronikus publikálás magyar vonatkozásaival, mivel ez a terület még annyira friss és képlékeny, hogy nincs értelme egy konkrét tipográfiai elképzelést automatizálni, mivel a *web design* módszerei, eredményei, és a vele szemben támasztott igények is igen gyorsan változnak. E sorok írója az elektronikus publikálás területéről nem is tud egységesítési törekvésekről vagy a tipográfusok és a weblapok készítői körében egyaránt elfogadott irányelvekről.

A diplomatervezés természetéből fakadóan és e sorok írójának szándéka szerint is az elkészített program szabad szoftver (*free software*). A licenc *GNU GPL* lett.

1.3. A kifejlesztett szoftverek

A dokumentumok készítésére használatos legfontosabb programok megismerése, és azok magyar vonatkozásainak összehasonlítása után e sorok írója a LaTeX-et vette részletes vizsgálat alá. Úgy találta, hogy a LaTeX sok tekintetben javításra, bővítésre szorul, továbbá hogy a legfontosabb javítások a LaTeX alapszintű programozásával elvégezhetők. Az alábbi fejlesztési célokat tűzte ki és valósította meg:

– *A legfontosabb javítások elvégzése.* Ilyenek például: az írásjelek utáni szóköz beállítása, a fejezetszámok és a képfeliratok magyar tipográfia szerinti szedése, a szóelválasztás javítása, a számozott és számozatlan felsorolások stílusának igazítása a magyar tipográfiához, képleten belüli sortörés engedélyezése a műveleti jel ismétlésével, tizedesvessző, képleten belüli közök csökkentése, megszólalások, megnevezett bekezdések és leíró listák díszpontjainak szedése, egyes LaTeX csomagok által generált feliratok magyarítása. A LaTeX az angoltól eltérő nyelveket a babel csomag segítségével támogatja. A babel magyar modulja, a magyar.ldf korábbi, 1.4-es verziója sok javítást végzett az alap LaTeX-en (de a fentieket nem), ezért e sorok írója úgy döntött, hogy a magyar.ldf-et fogja átírni és bővíteni. Az eredeti 1.4-es verzió kb. 20 kB méretű volt, viszont a diplomatervezés során keletkező 1.5-ös változat mérete 190 kB fölötti. A magyar.ldf-ről részletes leírás olvasható a 3.2. alszakaszban.
– *Néhány bonyolultabban kódolható szolgáltatás implementálása a LaTeX-en belül.* Ilyenek például: dátumok beolvasása és generálása, csillagos lábjegyzetek, a határozott névelő stabilabb generálása, toldalékok hangrendben megfelelő alakjainak számokhoz illesztése, utalások generálása a strukturális egység nevével, betűvel írt számnevek generálása, a tételszerű környezetek címsorainak magyarítása minden LaTeX csomag esetén, és a kész megoldás kompatibilitásának javítása más LaTeX csomagokkal. A fenti szolgáltatások is a magyar.ldf-ben valósultak meg.
– *Indexrendező segédprogram fejlesztése.* Mivel korábban nem volt olyan könnyen használható program, amely a magyar helyesírási és rendezési szabályokat figyelembe vette volna, ezért e hiányt kitöltendő készült a husort.pl (lásd a 3.4. alszakaszban).
– *Magyar bibliográfiai stílus készítése.* Korábban az irodalomjegyzék tételeit a szerzőnek kézzel kellett formáznia (például kézzel kellett kiemelni a könyv címét és a folyóirat nevét), mert az automatikus formázást végző bibliográfiai stílusok az angolszász hagyományt követték. Ezért elkészült egy stílusfájl, a huplain.bst (lásd a 3.5. alszakaszban).
– *Egyéb kiegészítők fejlesztése.* Számos egyéb segédprogram és stílusfájl készült, de ezek vagy nincsenek dokumentálva, vagy nem elég stabilak mindennapi használatra, ezért e dolgozat csak megemlíti őket. Ezek: ccs_extract.pl, lafmtgen.pl, better_afm2tfm.pl, ispell.pl, a CM-Super fontokból és a BibTeX8-ból készített Debian-csomag.

Az elkészült szoftverek legfontosabb adatait az 1. táblázat tartalmazza. E dolgozat számos egyéb programot is bemutat, melyek segítik a magyar nyelvű LaTeX dokumentumok készítését. A fenti táblázatban nem szereplő szoftverek nem e diplomaterv keretei között készültek.

A fenti szoftvereket e sorok írója a MagyarLaTeX nevű csomagba gyűjtötte össze, a csomagot a *GNU GPL* licenc alá helyezte (tehát szabad szoftvernek minősül), és letölthetővé tette a `http://www.math.bme.hu/latex/` webcímről. Valószínű, hogy a közeljövőben felkerül a CTAN-re is, a magyar.ldf legújabb verziója pedig része lesz a babel csomagnak. A MagyarLaTeX szoftvergyűjtemény tartalmazza a magyar nyelvű LaTeX dokumentumok készítéséhez szükséges legfontosabb csomagokat és stílusfájlokat, továbbá tartalmaz kiegészítő Perl szkripteket is. A Perl szkriptek a UNIX-os teTeX-hez készültek, de futtatásuk

1. táblázat. A kifejlesztett szoftverek

Név	Fájltípus	Méret, bájt
better_afm2tfm.pl	Perl szkript	35560
ccs_extract.pl	Perl szkript	6818
huplain.bst	BibTeX stílusfájl	48179
husort.pl	Perl szkript	54341
ispell.pl	Perl szkript	4381
lafmtgen.pl	Perl szkript	16590
magyar.ldf	LaTeX forrásfájl, babel modul	193214

elvileg lehetséges más rendszerek alatt is, melyeket a Perl támogat. A MagyarLaTeX fájljait általában elég az aktuális könyvtárba másolni, vagy a `texmf` fa ugyanazon könyvtárába, amelyben az archív-beli fájl található. A Perl szkriptek tetszőleges könyvtárból futnak.

E dolgozat is LaTeX-ben készült, az 1. táblázatban felsorolt összes szoftvert igénybe véve. Ezek a szoftverek egyrészt megkönnyítették a dolgozat írását és szedését, másrészt használatuk jobb minőségű művet eredményezett, mint ami nélkülözésükkel készült volna, másrészt a dolgozat elkészítése egy igen jó lehetőség volt e szoftverek tesztelésére (számos hibára fény derült, és ezek ki lettek javítva). A megjelenés alatt álló több, mint 750 oldalas [24] írása során is használták a szerzők ezeket a szoftvereket. A dupla sorköz alkalmazására olvashatósági (lásd [22, 17. oldal]) és esztétikai okokból nem került sor, továbbá azért sem, hogy ez a dolgozat példa lehessen egy tipikus, a magyar tipográfia szerint készülő dokumentumra, és az egyes formai elemek ajánlott függőleges távolsága is megfigyelhető legyen. A bemutatott tipográfiai szabályok alkalmazására sok esetben nem kell külön példát hozni, mert a dolgozat maga megfelel.

1.4. Fogalomtár

A szöveges dokumentumok számítógépes előállításához kapcsolódó legfontosabb fogalmak és szerepek:

dokumentum vagy **mű** ♦ A teljes írott mű, melynek előállításán a szerző, a tipográfus, a szedő, a szerkesztő, a lektor és mások együtt dolgoznak ([23, 3. oldal], [5, 239. oldaltól]). A dokumentum végleges formája egy vagy több lap, melyre a közlendőt hordozó betűk és egyéb jelek kerülnek. Elektronikus publikálás esetén a végeredmény egy vagy több fájl. A legnépszerűbb formátumok a PDF, a HTML és a PostScript. A dokumentum típusa általában könyv, napilap, folyóirat (vagy annak egy cikke) vagy reklámanyag. E dolgozat műszaki és tudományos témájú könyvekre és ilyen folyóiratokba szánt cikkekre összpontosít.

dokumentum-előkészítés (*document preparation*) ♦ Az a folyamat, melynek során a fent említett résztvevők előállítják a dokumentum végső, nyomtatásra vagy elektronikus közzétételre kész változatát.

szerző ♦ Az a személy, aki a dokumentum szövegét fogalmazza, az ábrákat megrajzolja vagy összeállítja stb. A műszaki–tudományos művek szerzői általában kutatók, oktatók vagy tanulók. A szerző jártas a mű tárgyterületén, de a dokumentum-előkészítés teljes folyamatát általában nem látja át. (Mindezt a szerzőre mint szerepre értve. Manapság gyakori, hogy ugyanaz az ember a szerzői feladatok mellett a tipográfus és a szedő munkáját is kénytelen elvégezni.)

kézirat ♦ A dokumentumnak az a változata, melyet a szerző juttat el a kiadóba. A kézirat minden információt tartalmaz, de még nem nyerte el végleges formáját (betűtípus,

margók stb.), és nincsenek benne kijavítva a hibák.

tipográfia ♦ Az 1970-es és '80-as években még könyvnyomtatást és nyomdászatot jelentett ([7]), beleértve a szellemi és a fizikai munkát is, de ennek számítógépes automatizálásával a szó jelentése eltolódott a tervezés felé. [22, 9. oldal] szerint a tipográfia „a szöveges közlés megformálásával, kép és szedett szöveg együttes elrendezésével [foglalkozik]. Hagyományosan a nyomtatott szövegek megtervezését tekintik tipográfiának, de ma már sok más technikája is van a szöveges közlésnek, például a képernyőre írt feliratok. A városképben, házfalakon megjelenő feliratok tervezése is tipográfiai ismereteket kíván. Általában a közlemény síkbeli elrendezését kell megtervezni [...]" A tipográfia nem kizárólag a tipográfus munkája, mert egy mű kinézetét szerző és tipográfus együtt alakítja. Paul Renner 20. századi meghatározása: a tipográfia „mechanizált grafika" (idézi [22, 8. oldal]).

tipográfus ♦ Az a személy, aki a dokumentum végső formáját, külalakját tervezi. Ebbe beletartozik a betűtípusok és a betűméret kiválasztása, a papírméret és a margók meghatározása, a színválasztás, a formai elemek (pl. fejezetcímek, élőfej, lábjegyzetek) megtervezése. Manapság sajnos gyakori, hogy külön tipográfusra és szedőre nem futja a költségvetésből, és ekkor az egész tipográfiai munka a képzetlen és egyébként is túlterhelt szerzőre hárul. A helyzetet súlyosbítja ([25, 283. oldaltól]), hogy a szerző el van látva nagy tudású számítógép-programokkal és betűtípusok ezreivel, de az ezekhez adott súgó és egyéb dokumentáció nem ad tipográfiai alapismereteket.

A szerepek keveredésére példa e dolgozat is: a tipográfus, a szerző és a szedő is e sorok írója volt egy személyben. A dolgozat készítése tipográfusi munkával kezdődött (margók, betűtípus, betűméret, vízszintes és függőleges közök meghatározása), majd a szöveg gépelésének szerzői munkája kezdődött. Néhány oldal begépelése után egy LaTeX fordítás következett (ami már szedői feladat), és javításra került néhány egyedi, valószínűleg nem ismétlődő (szedési) hiba. Mikor újszerű formai elem került a dokumentumba, e sorok írója műkedvelő tipográfusként megtervezte annak megjelenését, és a megfelelő parancsokat elhelyezte a forrásfájl preambulumába, vagyis abba a részbe, amely többek között az egész dokumentumra érvényes kinézeti beállításokat tartalmazza.

kinézeti terv (*layout*) ♦ A tipográfus munkájának eredménye. Ez papíron mintaoldal formájában, elektronikusan pedig stílusfájlként jelenik meg. A szedő dolga jóval egyszerűbb, ha megkapja a stílusfájlt, és eleve ezt használva gépeli be a művet.

szedő vagy **tördelőszerkesztő** ♦ Az a személy, aki a tipográfus által készített terv és a szerző által benyújtott kézirat alapján a dokumentum végső változatát elkészíti. Mivel a kéziratot manapság általában a szerző maga gépeli be elektronikusan szerkeszthető formában, a szedőre már csak a formázások elvégzése hárul. Ezek kiterjednek a szerző ismétlődő hibáinak javítására (például ékezetes betűk vagy írásjelek helytelen használata), a nagyobb és kisebb elemek formázására (nagyobb, ha például a szerző félkövérrel írta a fejezetcímet, ezt a szedő átformázza a fejezetcím végső stílusára, elhelyezi a sorszámot stb., kisebb a a különböző szövegen belüli kiemelések stílusának ellenőrzése, a szerencsétlen sortörések kiküszöbölése stb.).

Gyakori, hogy a tipográfus és a szedő ugyanaz a személy, de a két szerep időben ekkor is elkülönül egymástól: a kinézeti terv még azelőtt elkészül, hogy a szerző begépelte a kéziratot, a szedő csak ezután kezd dolgozni.

szerkesztő ♦ Az a személy, aki megtervezi, átlátja és felügyeli a mű születésének összes mozzanatát, összefogja és szakmailag irányítja a többiek munkáját. Egy műnek több szerkesztője is lehet ([5, 239. oldal]).

szövegszerkesztő (*text editor*) ♦ Olyan program, mellyel sima szövegfájlok hozhatók

létre. A sima szövegfájl formázási lehetőségei minimálisak: új sort lehet kezdeni, továbbá több egymás utáni szóközzel vagy tabulátor-karakterekkel a betűket egymás alá lehet igazítani. A szövegfájlok további korlátja, hogy a jelkészletük nem teljes (256 különböző karakter lehet egy sima szövegfájlban), bár ez a korlátozás a Unicode használatával kikerülhető. A szövegszerkesztő általában fix szélességű betűket használ, ugyanazt a betűtípust az egész szövegre, egységesen. A szövegszerkesztő tehát az írógéphez hasonló (azzal a fontos előnnyel, hogy a már begépelt szöveg módosítható). A vele készült szöveg nehezen tekinthető át és nehezen olvasható, továbbá egyes formai elemek (főleg ábrák, táblázatok és matematikai képletek, pl. $\sqrt{\alpha^2 + \beta^2}$) bevitele szinte lehetetlen. Példák szövegszerkesztőre: vi, Emacs és NotePad.

dokumentumszerkesztő (*word processor*) ♦ Olyan program, mellyel WYSIWYG* módon lehet a dokumentumot begépelni és megformázni. A hagyományos betű- és bekezdésformázási lehetőségeken kívül képek, ábrák, táblázatok és matematikai képletek is elhelyezhetők a dokumentumban, továbbá a dokumentumszerkesztő képes egyes formai elemeket (pl. tartalomjegyzék) önmaga generálni. A sor- és oldalhatárokat is a program állapítja meg a főszöveg és a kiegészítő úszó és körülfolyatott objektumok (pl. ábrák, táblázatok) alapján.

Általában a szerző, a tipográfus és szedő ugyanazt a dokumentumszerkesztőt használja, de más céllal. Az is gyakori, hogy a szerző egy dokumentumszerkesztőt, a tipográfus és a szedő viszont egy kiadványszerkesztőt használ, amely be tudja olvasni a szerző által küldött kéziratot. A mai dokumentumszerkesztők – sajnálatos módon – ömlesztve kínálják e három szerep által használt funkciókat, ezzel arra csábítva a szerzőt, hogy ő maga formázza nyomdakészre a dokumentumot, de ez csak ritkán ad szép végeredményt, mivel a szerzőnek a formázáshoz általában nincs se előképzettsége, se művészi érzéke. Manapság igen sok amatőr kiadású könyv jelenik meg, ahol nem alkalmaznak képzett tipográfust és szedőt, így a szerző jó szándékú formázásai kerülnek a végleges műbe. Szerencsére ez a gyakorlat műszaki–tudományos könyvkiadásban ritka, viszont az egyetemi oktatási segédanyagok többségén megfigyelhető, hogy a készítők közül senki sem rendelkezett alapvető tipográfiai ismeretekkel. (Ez részben azzal magyarázható, hogy a segédanyagok kis költségvetéssel, gyakran csak az oktatók magánszorgalmából készülnek. Az alapok elsajátíthatók például a *Dokumentumszerkesztés…* c. BME-n oktatott féléves kurzus keretében.) Példák dokumentumszerkesztőre: az OpenOffice dokumentumszerkesztője, AbiWord, Microsoft Word.

kiadványszerkesztő (*desktop publishing software*) ♦ A dokumentum végleges változatának elkészítésére kiélezett WYSIWYG szerkesztőprogram. Formázási és egyéb vizuális képességei meghaladják a dokumentumszerkesztőkéit, például az iniciálék létrehozása, az ábrák szöveggel való körülfolyatása és az egyes szövegrészek összekapcsolt dobozokba történő betöltése könnyebb, mint egy dokumentumszerkesztőben. Hátránya az ilyen programoknak, hogy bizonyos formai elemeket (pl. tartalomjegyzék, lábjegyzetszámozás) nem tudnak automatikusan generálni, ezért ezeket kézzel kell bevinni – és változás, például két fejezet megcserélése esetén a módosításokat is kézzel kell végigvezetni, ami sok hiba forrása lehet. A kiadványszerkesztőt használó tipográfus külön stílusokat tervez az oldalakra (pl. címoldal, szövegoldal, teljes oldalas kép; a formázás határozza meg a margókat, az élőfejet), a bekezdésekre (pl. a sorközt, a beütés mértékét) és a betűkre (pl. betűtípust, a betűméretet és a variációkat: félkövér, dőlt, aláhúzott stb.). A szedő általában csak a tipográfus által definiált stílusokat alkalmazza,

* Azt kapod, amit látsz (*What You See Is What You Get*): a bevitel és a formázások során a képernyőn megjelenő dokumentum éppen úgy néz ki, mint a végleges, nyomtatható változat.

és esetleg apró módosításokat végez, hogy elkerülje a vizuális szempontból előnytelen sor- és oldaltöréseket. Példák kiadványszerkesztőre: QuarkXPress, InDesign, Scribus.

szedőprogram (*typesetting software*) ♦ Olyan, általában nem WYSIWYG program, amely a szedő munkáját automatikusan elvégzi. Ehhez persze szükséges, hogy a kinézeti tervet tartalmazó stílusfájl egy determinisztikus algoritmus legyen, amely minden döntési helyzetben egyértelmű utasítást tartalmaz arra nézve, hogy az adott szövegelemet hogyan kell kiszedni. Szükséges továbbá, hogy a kézirat minden információt tartalmazzon a számítógép számára érthető formában (pl. melyek a fejezetcímek, az ábrák mely szövegrészekhez kapcsolódnak, mely szavak kerüljenek a tárgymutatóba stb.). A kézirat általában egy sima szövegfájl, ami a szövegen és az ábrákon kívül a fent említett struktúraleíró információt is tartalmazza. A dokumentum végleges változata a szedőprogram futtatásával kapható: a program bemenete a stílusfájl és a kézirat, a kimenet pedig a kész mű valamely közvetlenül nyomtatható vagy elektronikus publikálásra alkalmas formátumban. (Ebből a szempontból a webböngészők is szedőprogramnak minősülnek: a stílusfájl a CSS stíluslap, a kézirat pedig a HTML oldal, kiegészítve a képekkel és egyéb multimédia elemekkel. E technológia szedési lehetőségei azonban igen korlátozottak, például nem helyezhető el feltételes kötőjel a kéziratban, továbbá a betűtípus és betűméret nem írható elő teljes bizonyossággal; ezért a továbbiakban a HTML-lel nem foglalkozunk.) Ajánlott a struktúra és a megjelenítés szétválasztása: a kéziratba kerül az, hogy mely szövegrészek minősülnek fejezetcímnek, a stílusfájlba pedig az, hogy a fejezetcímet milyen betűtípussal kell szedni. Egy szigorú kinézeti terv nem engedi meg a szerzőnek pl. a dőlt betű használatát: helyette azt kell jeleznie, hogy valamit hangsúlyozni kíván, vagy egy idegen nyelvű kifejezést akar kiemelni stb. Szedőprogram például a LaTeX és a Troff.

A gyakorlatban mégis szükség van szedőre, mert vannak olyan formai elemek, melyek elhelyezésére nincs jó algoritmus (pl. mi történjen egy túl széles táblázattal; mit kell átvariálni, hogy elkerüljünk egy igen szerencsétlen sortörést; hogyan kell az ábrákat ízlésesen egymás mellé helyezni). Az is elképzelhető, hogy túl bonyolult lenne elmagyarázni a szerzőnek, hogy pontosan milyen formában küldje el a kéziratot. Ekkor a szerző csak egy egyszerűsített utasítást kap, és a szedő néhány helyen átírja a kéziratot (nem a szöveget, csak a metainformációt), hogy pontosan megfeleljen a kinézeti tervben előírtaknak. A szedőnek általában jóval kevesebb helyen kell beavatkoznia, mintha a szerző szedőprogram helyett WYSIWYG dokumentumszerkesztővel készítette volna el a kéziratot.

E dolgozatban nem áll módunkban végigkövetni egy műszaki–tudományos mű útját a szerzőtől az olvasóig. Ehelyett csak az angolszász hagyományoktól eltérő magyar nyelvi sajátosságokra szorítkozunk, melyek a tipográfus, a szerző és a szedő munkáját érintik. A három szerepet együtt fogjuk tekinteni, mert manapság egy szerzőnek, főleg a felsőoktatásban, mindháromhoz értenie kell, ugyanis az egyes jegyzetek, segédanyagok elkészítésekor – anyagi okok miatt – minden munka a szerzőre hárul.

1.5. A vizsgált programok

A dolgozatban az alábbi programokat vizsgáljuk meg magyar nyelvű művek létrehozásának szempontjából:

Microsoft Word ♦ Ez legelterjedtebb dokumentumszerkesztő az egész világon. Van magyar változata is magyar felhasználói felülettel és magyar nyelvű dokumentumok készítését támogató kiegészítőkkel: helyesírás-ellenőrző, nyelvtani ellenőrző, szóelválasz-

tó és szinonimaszótár. Könnyű elsajátítani a használatát, és viszonylag gyorsan lehet benne egy egyszerűbb elképzeléseknek megfelelő dokumentumot létrehozni. Ha egy műnek több szerzője van, akkor gyakran azért döntenek a Word mellett, mert azt mindannyian tudják legalább alapszinten használni, továbbá a legtöbb számítógépre könnyen telepíthető. Már általános- és középiskolákban oktatják a Word használatát, és ezt a számítógépes alapismeretek részének tekintik a meghirdetett (pl. titkárnői) állásokban is.

QuarkXPress ♦ A világon és Magyarországon is legnépszerűbb kiadványszerkesztő az InDesign mellett a QuarkXPress. Napilapok, színes magazinok, képes enciklopédiák és képes szakácskönyvek szedésére használják, vagyis olyan művekben, melyek lapjai sok hasábból állnak, gazdag formavilágúak, sok rajtuk a kép, a kiemelés és egyéb vizuális elem. Alapszintű használata nem sokkal bonyolultabb, mint a Wordé, de sok mindent be kell állítani, hogy elfogadható kimenet keletkezzen, és a betűtípusok helyes telepítése sem könnyű (ezzel szemben Wordbe a program indítása után azonnal el lehet kezdeni gépelni). Borsos ára miatt csak kiadók engedhetik meg maguknak a saját, legális példányt. QuarkXPressben minden tipográfiai feladat aránylag könnyen megoldható, de a megoldás csak korlátozott mértékben automatizálható: a lábjegyzetek számozását arab számról csillagra átírni például csak úgy lehetséges, ha a dokumentum összes lábjegyzetét kézzel módosítjuk. Terjedelmi okokból nem foglalkozunk külön az InDesignnal, melyben nem jelentkezik a QuarkXPress számos hátránya: például InDesignban nincs probléma az ékezetes betűkkel, mert a program Unicode alapú, és az OpenType fontok lehetőségeit is kihasználja, továbbá mind bekezdés-, mind oldalformázási lehetőségei is fejlettebbek.

Bár a QuarkXPress nem a legnépszerűbb műszaki és tudományos művek kiadására, Magyarországon is használják e célra, főleg kevés képletet és sok ábrát tartalmazó művek esetén. E dolgozat a 4-es verzió kiegészítők nélküli változatát hasonlítja össze más programokkal.

LaTeX ♦ A LaTeX (ejtsd: latekh vagy létekh) az 1980-as évek elején született szedőprogram. Nevének első részét a szerző, Leslie Lamport vezetéknevéből kapta, míg a TeX végződés pedig arra utal, hogy a LaTeX a TeX-et kiegészítő makrócsomag. Fő hátránya, hogy nem WYSIWYG (ez akkoriban még nem volt követelmény!), fő előnyei viszont egyrészt, hogy megalkuvás nélkül szedhetők vele egyszerű és bonyolult matematikai formulák (és ezek variánsai: fizikai képletek, kémiai átalakulásokat leíró egyenletek stb.), másrészt hogy a sima szöveg tipográfiai minősége páratlanul jó, még akkor is, ha a csak egy képzetlen, a LaTeX-hel éppen ismerkedő szerző készíti a dokumentumot, tipográfus és szedő nélkül. A már elkészült mű utólag is átformázható, általában kézi módosítás nélkül. A tanulási ciklus hosszabb, mint más programok esetén, akár hetekig vagy hónapokig is eltarthat, amíg a szerző kiismeri magát a LaTeX forrásban használható parancsok között. Ez viszont nem jelent több hónapos időveszteséget, mert ezalatt a szerző tipográfiai ismereteket szerez, és nagyobb tudatossági fokra lép a dokumentum-előkészítés egész területén. E sorok írója állítja, hogy egy ugyanolyan minőségű műszaki–tudományos mű dokumentumszerkesztővel vagy kiadványszerkesztővel való elkészítése nem gyorsabb és nem is kevésbé megterhelő, mint a LaTeX-hel, ha mindezt magas igényű és tapasztalt szerkesztő és szedő végzi. Ez nem vonatkozik a sok vizuális szerkesztést igénylő művekre: napilapokban (cikkek sok hasábban szedve, változatos elrendezésben, körülfolyatott képekkel) és reklámplakátokon; ezek szedése LaTeX-hel rémisztően körülményes.

A LaTeX dokumentummodellje a következő: a mű páros és páratlan (sorszámú) oldalakból áll; az oldalak többségén a főszöveg található egy vagy két hasábban; a főszöveg

alatt lábjegyzet, körülötte élőfej, élőláb és széljegyzet látható. A főszöveget néhány soros kiemelt matematikai képletek szakíthatják meg (de képlet elhelyezhető a szövegen belül is, például $a^2 \geq b^2 + c^2$). Az ábrák, táblázatok és egyéb magas, el nem törhető objektumok úsztatásra kerülnek: az oldal alján vagy tetején jelennek meg, és így nem zavarják a főszöveg folyását. (Nem helyezhető el tehát – korlátozások nélkül – körülfolyatott kép, vagy több, változó szélességű szövegdoboz ugyanazon az oldalon.) A műszaki–tudományos művek jól illeszkednek ebbe a modellbe*, matematikai folyóiratok és könyvek készítésére nincs is a LaTeX-nek versenytársa (az egyéb megoldások többsége is TeX-en alapul).

Meg kell említenünk a TeX és a LaTeX közti különbséget. A TeX egy makrók segítségével programozható szedőprogram-keretrendszer, amelynek beépített (*primitive*), alacsony szintű parancsai lehetővé teszik egy kinézeti terv definiálását. A parancsok azonban olyannyira alacsony szintűek, hogy a tipográfusnak hetekig kéne kódolnia a szerző által is használható stílusfájlt. E hosszú munka megspórolható, ha a tipográfus kiindul egy ún. formátumból (*format*). A formátum egy általános makrócsomag, amely a TeX primitívekre építve egy egységes, de azoknál barátságosabb felületet nyújt mind a kinézeti terv definiálása, mind a mű begépelése számára. Ilyen formátum a plain TeX, a ConTeXt, az $\mathcal{AMS}$-TeX és a LaTeX is. Ezek közül csak az utóbbival foglalkozunk, mert csak erre teljesül, hogy egyszerre általános célú, moduláris, bővíthető, elterjedt, magyar nyelvű könyvből megtanulható (pl. [23] és a megjelenés alatt álló [24]) és szabad szoftver (*free software*). A LaTeX fejlesztése nem fejeződött be az 1980-as évek elején: például az azóta született PostScript, PDF (és Flash) kimeneti formátumok támogatását utólag építették be. Ma is aktívan fejlesztett formátum a kereskedelmi ConTeXt, és a magyar GrailTeX.

Szövegszerkesztőkkel a továbbiakban azért nem foglalkozunk, mivel a sima szövegfájlok formázásai lehetőségei olyan korlátozottak, hogy a legalapvetőbb olvasói igények (jó olvashatóság, áttekinthetőség) sem elégíthetők ki velük. Megemlítjük a szövegfájlok egyetlen magyar vonását: az ékezetes betűket is tartalmazó szövegfájlok kódolása általában Latin-2, UTF-8 vagy UTF-16. A Latin-2 kódolású fájlok szinte az összes szövegszerkesztőben megnyithatók, az UTF-16 megnyitható például a Windows XP NotePadjében, az UTF-8 (és az UTF-16 is) pedig XEmacs-ben. A LaTeX források (stílusfájlok és kéziratok) is szövegszerkesztőben készülnek – a dokumentum végleges formája szempontjából azonban teljesen mindegy, hogy melyikben.

Számos egyéb program létezik, melyek több-kevesebb bosszúsággal használhatók dokumentum-előkészítésre (például AbiWord, KWord, SciWord, az OpenOffice dokumentumszerkesztője, a MagyarOffice dokumentumszerkesztője, LyX, FrameMaker). Az összehasonlításban azonban csak a fenti három program szerepel, mivel ők ma a saját kategóriájuk legelterjedtebb programjai Magyarországon. A Word elsőségét, úgy tűnik, más program nem fenyegeti, a QuarkXPress és az InDesign egymásnak erős versenytár-

* Donald E. Knuth, a Stanford egyetem oktatója éppen azért írta a TeX-et, mert nem volt elégedett a kimenettel, melyet *A számítógép-programozás művészete* ([9]) c. könyve első kiadásának szedői produkáltak. Akkoriban még nem létezett olyan program, amely megbirkózott volna a matematikai képletekkel. Knuth háromkötetes, összesen kb. 2000 oldalas, rengeteg képletet és számos ábrát tartalmazó könyve volt a TeX első éles alkalmazása. A könyv magyar kiadása ([11]) nem TeX-hel készült, annak ellenére, hogy akkoriban a TeX már 16 éve létezett, és a külföldi matematikai és természettudományos publikációk többségét TeX-ben készítették. Magyarországi elterjedésére várni kellett. E tekintetben úttörőnek számít az 1990-ben kiadott [4]. 1991-ben már a TypoTeX kiadó is jelentetett meg TeX-hel készült könyvet. [11] szedésében a Műszaki Könyvkiadó és a Kossuth Nyomda munkatársai kiváló munkát végeztek, a szöveg és a képletek is gyönyörűek.

sai (és feltörekvő szabad szoftver a Franz Schmid által által fejlesztett Scribus, melynek magyarítása a Scribus.hu), a LaTeX-hel pedig – tudását tekintve – egyedül a ConTeXt és a feltörekvő GrailTeX konkurál (de ezen TeX-változatokat a műszaki–tudományos művek szerzőinek többsége nem ismeri).

2. A magyar nyelvi sajátosságok elemzése

Az alábbiakban ismertetjük, hogy mik a magyar tipográfia azon sajátosságai, melyek eltérnek az angolszász hagyományokétól. Mivel az összehasonlításban részt vevő 3 program egyaránt egy angolszász tőről fakad, az alábbi lista egyben arról is tájékoztat, mely pontokon lehet szükséges e programok magyarítása. Magukat a programokat a 2.4. alszakaszban hasonlítjuk össze. Nem térünk ki az egész latin betűs nyelvterületen és/vagy Európában egységesen alkalmazott hagyományokra (pl. arra, hogy a szöveg felülről lefele, azon belül balról jobbra folyik, továbbá hogy milyen megfontolások alapján választja a tipográfus a tükörszélességet*, a betűméretet és a margókat). A listában található tényanyag főleg [5, 22, 3, 6, 21]-en alapul, a hivatkozások az egyes elemeknél részletesen meg vannak jelölve. E szakasz néhány pontja kitér arra is, hogy a három összehasonlított program hogyan felel meg az adott sajátosságnak, mivel a 2.4. alszakaszban található táblázatba hosszabb, szöveges jellemzés nem fér.

2.1. Betűk, szavak

2. táblázat. Magánhangzóink helye a Latin-1, Latin-2 és Unicode kódolásokban

	l1	l2	uni		l1	l2	uni		l1	l2	uni		l1	l2	uni
a	97	97	97	á	225	225	225	A	65	65	65	Á	193	193	193
e	101	101	101	é	233	233	233	E	69	69	69	É	201	201	201
i	105	105	105	í	237	237	237	I	73	73	73	Í	205	205	205
o	111	111	111	ó	243	243	243	O	79	79	79	Ó	211	211	211
ö	246	246	246	ő		245	337	Ö	214	214	214	Ő		213	336
õ	245		245	ô	244	244	244	Õ	213		213	Ô	212		212
u	117	117	117	ú	250	250	250	U	85	85	85	Ú	218	218	218
ü	252	252	252	ű		251	369	Ü	220	220	220	Ű		219	368
ũ			361	û	251		251	Ũ			360	Û	219		219

– *Ábécé.* A magyar ábécé kisbetűi sorrendben: a á b c cs d dz dzs e é f g gy h i í j k l ly m n ny o ó ö ő p q r s sz t ty u ú ü ű v w x y z zs. A többjegyű mássalhangzók karakterenként felépíthető az angol ábécé 26 betűjéből (pl. n + y = ny). A hosszú magánhangzók többsége része a Latin-1 (nyugat-európai) 8-bites kódlapnak, melyet a nyugat-európai piacra szánt, angol nyelvű programok alapértelmezésben támogatnak. Az *ő* és *ű* betűk azonban csak a Latin-2 kódlapban találhatók meg, ezért a programokat úgy kell konfigurálni, hogy a Latin-2-t használják. A 2. táblázat áttekintést ad a magyar magánhangzók helyéről a három legfontosabb kódolásban. Láthatjuk, hogy 255 fölé csak a Unicode-os kódok emelkednek. (Knuth már a [10]-ben megemlíti pályatársa, Erdős Pál nevét. Könnyen meglehet, hogy a TeX-ben a kezdetek óta meglevő ″ ékezetet Erdős Pálnak köszönhetjük. A LaTeX T1 fontkódolása már az összes latin betűs európai nyelvet támogatja, ezért, ha a plain TeX-ből kimaradt volna a ″, akkor az

* a főszöveg bal és jobb széle közti távolság

¸ (ogonek) ékezettel együtt került volna be a LaTeX-be.) A Latin-2 nem minden programban választható, ezért gyakori anomália, hogy a Latin-1-ben azonos pozíción levő *õ* és *û* idegen betűk kerülnek felhasználásra. Amikor Márai Sándor 1950-ben, a *Halotti beszéd* c. versében azt írta, hogy „lehull nevedről az ékezet", nem gondolt arra, hogy az 1980-as évektől kezdve hazáján belül is problémát jelent majd az ékezetes betűk írása.

A magyar ékezetek általában alacsonyabban vannak, mint a nyugat-európai nyelvekben, például ö helyett ö betűt kell szedni. Bizonyos betűtípusok, például e dokumentum betűtípusa is eleve alacsony ékezetes betűket tartalmaznak.

A QuarkXPress mindenképpen a nyugat-európai kódlapot használja, így *ő* és *ű* betű gépelése csak a fontok kézi módosításával lehetséges, ami fáradságos és szakértelmet igénylő feladat. A módosítás nélküli fontok a képernyőn néha helyesen jelennek meg, de nyomtatásban rossz ékezetek keletkeznek.

– *Szóelválasztás.* [3, 23. oldal] A magyar elválasztási szabályok igen egyszerűek: bármely két magánhangzó között vagy egy magánhangzót megelőző mássalhangzó előtt el lehet választani a szót, pl. *kar-csú-ak*. Kerülendő az első betű után és az utolsó betű előtt történő elválasztás. Sajnos a fentieket bonyolítja, hogy az összetett szavakat az összetétel határán, továbbá a részszavak elválasztási pontjain lehet elválasztani, pl. *meg-érint*, de pl. a *megint* nem választható el, mert csak a szövegkörnyezetből derül ki, hogy összetett szónak kell-e tekinteni. Szó elején és végén az elválasztás tiltása az részszavakra is érvényes. Azt szeretnénk, hogy a számítógép keresse meg az elválasztási pontokat, és szúrjon be feltételes kötőjeleket (például azért, hogy csökkentse a sorkizárt bekezdésekben a szóközök szélességét). Ehhez viszont vagy az összes összetett szót tartalmazó szótárra, vagy egyéb, az összetett szavakat érzékelő eszközökre van szükség. A magyar helyesírás az idegen nyelven összetett szavak mindkét elválasztását megengedi ([12, §231, §237]), pl. *szin-kron* és *szink-ron*, de egy dokumentumon belül ezek nem keverhetők. Ha az elválasztás helyén eleve van kötőjel (pl. *nátrium--klorid*), akkor a következő sor elején az igény szerint megismételhető. [22, 43. oldal] további esztétikai ajánlásokat fogalmaz meg, melyek azonban csak külön odafigyeléssel tarthatók be. [21, 287. oldal] megtiltja nyomtatott műben a magánhangzó előtti elválasztást.

A Wordbe a *Morphologic* Helyesel elválasztóprogramja van beépítve, ami kiegészítőként a QuarkXPress-hez is megvásárolható. A LaTeX-hez készült magyar elválasztási minták leírását lásd a 3.3. alszakaszban.

– *Nem törhető szóköz.* [22, 43. oldal] [21, 287. oldal] Ma már eltörhető a sor az egybetűs szavak után (a, e, ó, ő, s), de szebb, ha az *a*-t néha átvisszük, néha pedig nem a következő sorba, így se a sorok elején, se a sorok végén nem alakulnak ki egymás fölötti *a* névelő. Uralkodók nevein (pl. IV. Béla) és összetartozó fogalmakon belül (pl. *i.e. 500*, *3.4. alszakasz*, *G. I. Jakab*, *dr. Kis Pál*, *5. kerület* első szóköze, és a dátumok szóközei) nem törhető el a sor. A jóval korábban készült [21, 286. oldal] nem engedi meg a sorvégi egybetűs szavakat.
– *Dupla többjegyű mássalhangzók.* A *tty* betűkapcsolat elválasztásnak módja magyar szövegben *ty-ty*, vagyis egy új, *y* karaktert is be kell szúrni. A dokumentumszerkesztők többségének szedési modellje ezt nem teszi lehetővé, ezért ezek a programok nem választanak el ezen a ponton. A többi kétjegyű mássalhangzónál is hasonló a helyzet, sőt a *ddzs*-nél is ez (*dzs-dzs*), de dupla *dzs*-t csak ragozott magyar szó tartalmaz: *bridzs-dzsel* (főnév). Kivételek is akadnak, például *vil-la-mos-szék*. A helyes elválasztáshoz egy kivétellistára vagy egy morfológiai elemzőre van szükség.
– *Ábécérend.* [3, 22. oldal] Az ékezetes betűk és a többjegyű mássalhangzókat tartalmazó szavak besorolása kérdéses. A pontos szabályokat lásd a 3.4. alszakaszban. Szótárak

szedésekor, továbbá könyvek tárgymutatójának és irodalomjegyzékek ábécérendbe állításakor van szükség arra, hogy a gép ismerje e szabályokat.

– *Ligatúrák.* [22, 25. oldal] A ligatúra néhány szomszédos betű összeolvadása, például f + i = fi. A legismertebbek az f-ligatúrák, ezeket a következő mondat ([23, 18. oldal]) mind tartalmazza: *Puff, oda a maffia mafla fia!* A ligatúrák még a kézi kódexmásolás idejéből származnak, használatuk akkoriban gyorsította a másolást; ma már csak esztétikai szerepük van. A magyar nyelvben a kézi szedés idején használatosak volt a *gj*, *gy* és *gz* ligatúrák (lásd az 1. ábrán [22, 25. oldal]) de ezek sajnos hiányoznak a legtöbb kereskedelmi forgalomban kapható betűtípusból, beleértve az *Adobe* OpenType fontokat is, melyek több, mint egy tucat ligatúrát tartalmaznak. (Az et-jel a kézi másolás idején még ligatúra volt: e + t = &, de mára önálló jellé nőtte ki magát. A betűtervezők e jel cifrázásával csillantják meg kreativitásukat; már e dolgozat szövegbetűjének álló (&) és dőlt (*&*) változatában is igen különböző az et-jel.) Nem minden betűtípus tartalmaz minden ligatúrát, egyes betűk olvashatósága romlana, ha mindet összeolvasnánk.

1. ábra. A *gy* betű két karakterrel (balra) és a *gy* ligatúra (jobbra)

A QuarkXPress nyugat-európai betűtípus-használata nem engedi beszúrni a ligatúrákat, Word ugyan engedi, de a helyesírás-ellenőrző hibásnak ismeri fel, a keresés pedig nem találja meg a ligatúrákat tartalmazó szavakat. A TEX-ben nincsenek ilyen korlátozások, tetszőleges ligatára létrehozható.

– *A határozott névelő.* [3, 21. oldal] A műszaki–tudományos művek sok utalást tartalmaznak önmagukon belülre, például „lásd a 45. oldalon", „az 5.7. táblázat első oszlopa", „ez következik az 1.2. szakaszban leírtakból", „lásd a B. függelékben". Egyes programok maguk tudják generálni ezeket a számokat, ha a szerző a dokumentumban címkéket és a címékre való utalásokat helyez el. Mindez mit sem ér azonban, ha a program az *a/az* határozott névelő megfelelő alakját nem tudja a generált szám vagy betű elé illeszteni. A római számokra külön figyelni kell: „a x. oldalon", de: „az x tengely".

QuarkXPress-ben ez csak kiegészítő írásával oldható meg (C nyelven), Wordben pedig a Wordmezők programozási lehetőségei nem elegendőek.

– *Toldalékolás.* [3, 22. oldal] Az utalások generált szövegrészei, főleg ha képletekre hivatkoznak, néha toldalékolt formában állhatnak: „(1)-et hozzáadva (2)-höz, majd azt kivonva (4)-ből az (5)-öt kapjuk". Ezek generálásakor meg kell valósítani a toldalékok hangrend szerinti automatikus hozzáillesztését. Elég, ha a program az egész számokat tudja toldalékolni, mert ezek a szövegrészek szinte mindig egész számra végződnek. Némi utánagondolással megállapítható, hogy minden toldalékra igaz, hogy alakja csak attól függ, hogy a szám hány nullára végződik, és hogy mi az utolsó nullától különböző számjegy. (Az viszont tévedés, hogy minden toldaléknak két (*-ból/ből*) vagy három (*-hoz/-hez/-höz*) alakja van, mert a *-val/vel* és *-vá/-vé* ragok *v* betűje a teljes hasonulás miatt megváltozik. Az is tévedés, hogy egy számhoz csak egy toldalék rendelhető, ellenpélda az *5-öshöz* és a *3-adikon*).

– *Utalások strukturális egységekre.* [19] A dokumentum strukturális egységei (a nagyobbtól a kisebb felé haladva) a részek, a fejezetek, a szakaszok, az alszakaszok (stb.) és a nevesített bekezdések. A fejezettel vagy a résszel azonos azonos szinten áll a függelék.

Az utalásnak nem része a strukturális egység neve, például a „*az 1.2.* szakaszról” utalásban a gép csak a kiemelt névelőt és a szakaszszámot generálja, magát a „szakasz” szót a szerzőnek kell bevinnie. Ha viszont egy átszerkesztés során az adott szakaszból például önálló fejezet lesz, akkor ezt az utalásokban kézzel módosítani kell. Csökkenti a hibalehetőséget, ha az egész utalást (névelő, szám, strukturális egység neve és toldalék) a gép generálja.

– *Kötőjel, gondolatjel.* [22, 30. oldal] A nyomdászatban használatos vízszintes vonalak a következők: elválasztójel = kötőjel = diviz = *hyphen* (-), gondolatjel = félkvirtmínusz = *en-dash* (–), nagykötőjel = kvirtmínusz = *em-dash* (—), a kivonás jele = subtraction signkivonás jele (−). Ezek többféle célra használatosak ([23, 20. oldal]). A kötőjel szóösszetételekben (pl. *hébe-hóba*), sorvégi elválasztásban, becslésekben (pl. *25-30* év múlva) és kötőjeles szóösszetételben (pl. *Margit-híd*) használatos. A gondolatjel fix számtartományokban (pl. *a 30–40. oldalakon*), irányokban (pl. *észak–déli*), márkanevekben (pl. *TU–154-es*) használatos. Ha egy alkotásnak több szerzője van, akkor akkor a szerzőket – mind az irodalomjegyzékben, mind a főszövegben – vékony szóközzel határolt gondolatjellel választjuk el, sortörés csak jobb oldalt lehetséges (pl. *Bolzano – Weierstrass-tétel*). Az előző mondat közbevetett gondolatot tartalmaz, melyet általában is szóközzel határolt gondolatjelek fognak közre. Az angolszász tipográfia ez utóbbi helyett a nagykötőjelet is megengedi, de a magyarban tilos nagykötőjelet használni.
– *Bekezdésen belüli idézetek.* [22, 27. oldal] Máshonnan átvett rövid szövegrészletekből, a szereplők gondolataiból vagy sajátos szóhasználatából, továbbá túlzó vagy ironikus szókapcsolatokból válik bekezdésen belüli idézet. (A bizonytalan megfogalmazást pontosítani kell, nem pedig macskakörömmel ellátni.) A záró írásjelet az idézeten belül kell elhelyezni, ha oda tartozik. Az idézetek egymásba ágyazását a következő példa szerinti jelekkel kell elvégezni: „Így gondolkozott magában: »Ha nem fogadom el a felkérést, lemondhatok a 'parkett ördöge' címről.«”. A német, a francia, az amerikai és a brit hagyomány is eltér a fentiektől, az utóbbiban például így néz ki egy kétszintű idézet: ‘“I must accept the invitation,” she thought.’ Több bekezdésen át tartó idézet esetén az angolban a folytató bekezdések elején ki szabad tenni újra a nyitó idézőjelet, míg a magyarban ez tilos. [21, 286. oldal] a csupa nagybetűs címekben macskaköröm helyett rögtön a lúdláb használatát javasolja.

  Mivel a QuarkXPress egy adott karakterpozíción várja a nyitó és a záró idézőjelet, ezért a fontok szerkesztése szükséges, hogy az idézőjelek automatikus javítása működjön gépeléskor és importáláskor. LaTeX-hel és Worddel nincs ilyen probléma.
– *Láb, hüvelyk, fok, szögperc, szögmásodperc.* [22, 27. oldal] Mindegyiknek külön jele van, melyek az idézésre használt jelektől is különböznek, például: 5′6″, $12^\circ\ 34'\ 56''$. A fok, a szögperc és a szögmásodperc között szóközt kell kihagyni [21, 291. oldal].

  A LaTeX T1 fontkódolásának a láb jele nem része (a textcomp csomagban viszont definiálva van), bár a TeX-ben használt legtöbb Type 1 font része ez a jel. Az OT1 fontkódolásnak se a láb, se a hüvelyk jele nem része. A Wordben a fenti összes szimbólum megtalálható, a QuarkXPress pedig csak a nyugat-európai fontkódolást ismeri, ezért szöveges fontokból a szögperc és a szögmásodperc jele hiányzik.
– *I.* [22, 26. oldal] A függőleges vonalra hasonlító jelek: I l 1 i ı 1. Ezek sorrendben: nagy I, kis l, normál 1-es, kis i, kiskapitális i, ugráló 1-es. Ezek nem keverendők össze.
– *O.* [22, 26. oldal] Kör alakú jelek: O 0 o o o ○. Ezek sorrendben: nagy O, normál nulla, kis o, kiskapitális o, ugráló nulla, a kör műveleti jel. Ezek nem keverendők össze.
– *Gregorián dátum.* [12] Csak az alábbi formák helyesek: 1848-03-15 (ISO dátumformátum, csak a legújabb kiadás óta); 1848. március 15. (teljes kiírt hónap); 1848. márc. 15. (rövidítve kiírt hónap); 1848. III. 15. (római számos hónap); 1848. 03. 15. (arab

számos hónap); 1848. március közepe (teljes kiírt hónap, nap nélkül); 1848. március 15-én; (teljes kiírt hónap, a nap toldalékolva); 1848 március-ában (év és hónap birtokos szerkezetben).

– *Időpont.* [22, 27. oldal] Például délután 5 óra 6 perc 7 másodperc így szedendő: 17:06:07, ugyanez 12 órával később pedig: 05:06:07.
– *Zárójel.* [21, 289. oldaltól] A zárójel nem helyettesíthető törtvonallal. A zárójelek egymásba ágyazása: ([⟨ ⟩]). Szögletes zárójelbe teendő egy sima (gömbölyű) zárójellel befejeződő kifejezés, például jogi utalás. Az irodalmi hivatkozásokat és az idézeten belüli betoldásokat is szögletes zárójelbe kell tenni. Matematikai képletekben a hierarchia a következő: { [( )] }, bár megengedett az egymásba ágyazott sima zárójelek használata is.
– *Írásjelek.* [3, 24. oldal], [22, 28–29. oldalak] A mondatzáró pont és a mondaton belüli vessző szorosan tapad a zárószóhoz; a mondaton belüli pontosvesszőt és kettőspontot, továbbá a mondatzáró kérdőjelet és felkiáltójelet pedig egy-egy keskeny szóköz választja el az utolsó szótól, például: *Megfeledkezdél a virágról?*, *Nem, nem! Hoztam: ibolyát és gyöngyvirágot; kaktuszt és fikuszt; no és persze tulipánt is.* Az angolban minden írásjel tapad a zárószóhoz, míg a francia hasonló a magyarhoz, de ott a kettőspont előtt egy egész szóközt ki kell hagyni. Ha magyar szöveget kell egy olyan programban begépelni, amely a keskeny szóközt nem támogatja, akkor magyarban a szóköz elhagyása, franciában egy teljes szóköz beszúrása a gyakorlat.
– *Az írásjelek utáni köz.* [3, 24. oldal] A magyarban, csakúgy, mint a franciában, minden írásjel után egységesen egy szóközt kell kihagyni, ha az írásjelet betű követi (kivéve a zárójel nyitó párját, mert ott a szóköz az írásjel elé kerül). Azok a jelek, melyek a szöveg felolvasásakor külön szóként hangzanak el, nem számítanak írásjelnek (ilyenek például a /, az & (ugyanaz, mint &), a * és a matematikai jelek).
– *Betűválasztás.* [22, 66. oldal] a 2. ábrán látható felosztást közli, megjegyezve róla, hogy igen durva, és hogy a betűtípus-csoportokon belül is igen nagy eltérések lehetségesek. A gyakorlatban műszaki és tudományos művek vagy a Times (barokk) vagy a Computer Modern (barokk) betűtípust használják: az előbbi ugyanis a legtöbb dokumentumszerkesztő, az utóbbi pedig a TEX alapbeállítása, és nem is kell értük külön fizetni. Mindketten igen jól olvasható betűtípusok, de a Computer Modernnek jóval vékonyabbak a vonalai, ezért a kinyomtatott lap könnyedebb összbenyomást kelt. E dolgozat szövegbetűje a Minion, ami a 20. század végének terméke, reneszánsz és barokk jegyekkel is rendelkezik. A betűválasztást segíti a gazdagon illusztrált [20], amely a betűmetszés mint művészet történetét végigkövetve bemutatja a a 20. század második felében a kézi szedésben rendelkezésre álló betűtípusokat.

2. ábra. Útmutató a stílus alapján történő betűválasztáshoz

– *Olvashatóság.* [22, 48. oldal] Bár nincs magyar vonatkozása, megemlítjük, hogy a tükörszélesség, a betűtípus, a betűváltozat, a betűméret a sortávolság, a margó és a papír színe is befolyásolják az olvashatóságot. A legjobban olvasható mű keskeny (átlagosan 40…80 leütés soronként), valamely talpas, változó vonalvastagságú, változó szélességű (*monospaced*) betűtípus álló, nem félkövér, 10…12 pontos változatával szedett, 2–3 cm margójú, nem fényes fehér vagy krémszínű papírra nyomott. Ezen belül nagy szabadsága van a tipográfusnak. Ezek indoklása a hivatkozott helyen megtalálható. Itt említjük meg, hogy a betűméret megadására használt „pont" (pt) mértékegység az *inch*-nek (in) – a tipográfiai hagyományoktól függően – kb. $^1/_{72}$-ed része (1 in = 2.54 cm). Egy 10 pontos betűben nagybetűk és a felnyúló szárú kisbetűk magassága kb. 7 pt, az ajánlott sortávolság pedig 12 pt.
– *Egalizálás* (*kerning*). [22, 14. oldal] Bizonyos betűpárok közelebb vagy távolabb vihetők egymástól, hogy a betűközök egységesnek látszódjanak, például a Tévé szó első két betűje nincs egalizálva, de a Tévé minden szomszédos betűpárja igen. Az egalizálási táblázat (*kerning table*) a font része, és betűváltozatonként eltérő. A legtöbb font nem tartalmazza a magyar ékezetes betűk egalizálási párjait, tehát a Tevét egalizálják, de a Tévét nem. Ezt általában a fontfájl utólagos szerkesztésével kell megjavítani, vagy, ha ez szerzői jogokat sért, akkor a dokumentum-előkészítő programot kell átkonfigurálni, hogy egy másik egalizálási táblázatot használjon.

  TEX-ben az ékezetes betűk egalizálása csak akkor működik, ha azok nem külön ékezetből és betűből építettek. Magyar szöveg esetén ehhez a T1 fontkódolást kell használni. A legtöbb fontból (programtól függetlenül) hiányoznak az ékezetes betűk egalizálási párjai, TEX és T1 fontkódolás esetén ezek a MagyarLATEX részét képező tfm_ligkernfix.pl Perl szkripttel pótolhatók. E szkript lemásolja az ékezet nélküli betű egalizálását. E dokumentum *Té* párja is így készült.

2.2. A folyószöveg egyéb elemei

– *Optikai sorkizárás.* [22, 21. oldal], [10, 394. oldal] Megengedett, hogy sorkizárás esetén az írásjelek és a betűk is kilógjanak mind jobb, mind bal oldalt, ha ezáltal egyenletesnek *látszódnak* a margók. Csak kevés program képes ezt automatikusan elvégezni.
– *Címsorok középre igazítása.* [22, 20. oldal], Nagy méretű betűvel szedett többsoros címek középre igazításánál megengedett, hogy az idézőjelet, a három pontot és a gondolatjelet a szedő nem teljes szélességükkel vegye figyelembe, ha ezáltal a középre igazítás szebbnek látszódik.
– *Matematikai képletek.* A fizikai mennyiségek szedésével [6, 58–63. oldalak] foglalkozik. [6, 97–104. oldalak] röviden összegzi a matematikai képletekkel kapcsolatos tudnivalókat, és [21, 251–279. oldalak] részletesen kitér a legtöbb konstrukció szedésére, amely képletekben előfordulhat. E dolgozat nem térhet ki e szabályok részletes ismertetésére, főleg azért nem, mert előbb ezeknek a mai képletszedő programok képességeivel való pontos összevetésre lenne szükség. Például [10, G. függelék] egy igen általános, és megfelelő fontok használata esetén gyönyörű képletek szedésére alkalmas modellt ír le. Ehhez képest naivak a többi hivatkozott mű *ad hoc* utasításai egy fél vagy harmad szóköz kihagyására.
– *Kémiai képletek.* [21, 279–281. oldal] A kémiai reakciókban szereplő vegyületek álló betűvel szedendők. Az indexeknek egy vonalban kell lenniük. A bonyolultabb, ábraszerű képletek szedési hagyományai a fent említett helyen találhatók.
– *Magas törtek.* Ha a szövegben törtszám szerepel, azt teljes méretben kell kiszedni (pl. $\frac{1}{2}$

és nem $\frac{1}{2}$), még akkor is, ha ez elrontja a sortávolságot. A magyar nyelvű matematikai tankönyvek és cikkek követik ezt a hagyományt.

– *Tizedesvessző.* [3, 26. oldal] A matematikai képletekben kétféle vessző használatos: az egyik felsorolásra, például $f(x, y)$, a másik pedig valós számokon belül a tizedesvesszőre, például $12{,}34$. E két jel után eltérő mértékű üres helyet kell kihagyni, ezért vagy különböző módon kell bekérni a két jelet a felhasználótól, vagy a gépnek kell megállapítani a szóköz mértékét (például úgy, hogy megnézi, hogy a következő karakter számjegy-e). A tizedesvessző helyett a műszaki–tudományos művekben a tizedespont használata terjedt el, mert például a $[-1{,}2, 3{,}4]$ intervallumot első ránézésre nem könnyű értelmezni, a $[-1.2, 3.4]$ intervallum határai pedig rögtön látszanak. A legtöbb programozási nyelv is csak a tizedespontot engedi meg.
– *Képletek elválasztása.* [3, 26. oldal] Egyes programok megengedik a sortörést (pl.; $1 + + 2 + 3 < 9$) egy szövegközi képlet közepén. Magyar nyelvű dokumentum esetén a műveleti vagy relációjelet a következő sor elején meg kell ismételni. Nem törhető el a sor a szorzás ($\cdot$) és az osztás ($/$) jele után.
– *A képleten belüli közök.* [3, 26. oldal] A tipográfia szűkebb közöket ír elő az angolszásznél, például

$$\text{angol: } a + b - c/d * y \circ x = z \qquad \text{magyar: } a + b - c/d * y \circ x = z.$$

– *Mértékegységek.* [6, 103. oldal] A fizikai mennyiség számértéke és a mértékegység között szóköz kell, hogy álljon. (E dolgozat egy másik hagyományt követ, és csak egy keskeny szóközt hagy, például 42 cm.) A mértékegységek álló betűs SI vagy egyéb alapegységek szorzataként állnak elő, például $\mathrm{A} \cdot \mathrm{h}$ (amperóra). A többtényezős nevezőt zárójelbe kell tenni, például $\mathrm{W}/(\mathrm{m}^2 \cdot \mathrm{K})$. A nevező helyett negatív kitevős mértékegység is állhat, például $\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{K}^{-1}$.
– *Megszólalások.* [22, 29. oldal] Ha egy prózai mű szereplője megszólal, szavait új bekezdésben, gondolatjellel és szóközzel kezdjük. A gondolatjel utáni szóköz sorkizárás esetén sem nyúlik. Például:

– És nem maguk tették a feljelentést az őrsön?
– De mi – mondta Bogya.

Ennek ellent mod a [3, 24. oldal]-on levű példa, ami a megszólalásokat gondolatjeles listaként hozza létre. Ez valószínűleg hibás, mert e sorok írója még csak a fenti módon bemutatott párbeszédet látott nyomtatott szövegben.
– *Ugráló számjegyek* (*oldstyle numerals*). [22, 38. oldal] A régebbi könyvek „ugráló”, az alapvonalról lelógó számjegyeket tartalmaztak, például 1984. Mivel ezek a számjegyek a legtöbb betűtípusból hiányoznak, a számítógépes dokumentum-előkészítésben visszaszorult a használatuk. Táblázatokat és tartalomjegyzéket amúgy sem praktikus ilyen számjegyekkel szedni, mert ezek váltakozó szélességűek, ezért nem igazíthatóak egymás alá. Az igényesebb fontok (pl. az *Adobe* OpenType fontok) négy stílusban is tartalmaznak számjegyeket: ugráló és álló számjegyek fix és változó szélességben. Műszaki–tudományos szövegben, főleg képleteken belül csak álló számjegyek használatosak.
– *Soregyen* (*baseline grid*). Egy mű soregyent alkalmaz, ha a főszöveg sortávolsága állandó. (A sortávolság általában 10–20%-kal nagyobb a betűméretnél.) Ez normál bekezdéseknél könnyen biztosítható, de a fejezetcímeknél, az úsztatott objektumoknál, a kiemelt matematikai képletek körül és az egyéb kiemeléseknél oda kell figyelni, hogy az őket követő szöveg újra a soregyenre kerüljön. A soregyen ellentmond a magas törtek-

re vonatkozó, a 22. oldalon található szabálynak, ezért matematikai témájú művekben eltekintenek a soregyentől. Természetesen a fejezetcím alapvonalát nem kell a soregyenre igazítani (de az utána következő szöveget már igen!), így például megengedett egy két sort elfoglaló, másfél sor magasságú cím. A soregyen természetes velejárója, hogy az oldal utolsó sora pontosan az oldal aljára kerül (kivéve, ha a következő oldal egy több sort elfoglaló címmel kezdődik).

E dolgozat nem kifejezetten matematikai témájú, mégsem alkalmazza a soregyent, mivel a készítéshez használt program kódjába több ponton kellett volna belenyúlni a kielégítő végeredmény érdekében. E sorok írójának nincs tudomása olyan műről, amely LaTeX-hel készült és soregyent alkalmazna, kivéve a több, mint 750 oldalas [24]-ben tett kísérletet, ami számos problémára hívta fel a készítők figyelmét. Ezek ismeretetése és egy javított megoldás kidolgozása ugyan a témába vág, de túlmutat e dolgozat keretein. Wordben hasonló problémák jelentkeznek: ugyan be lehet állítani állandó sortávolságot egy bekezdésen belül, a címek, többsoros kiemelése és úszó objektumok környezetében a soregyen tartásáról kézzel kell gondoskodni. A QuarkXPressben az egyes sorok alapvonalainak lejjebb vitele a soregyenrácsra való illesztés céljából automatikus, de néha kézi beavatkozásra van szükség, mert inkább feljebb csúsztatás kívánatos.

– *Címek.* [5, 71. oldal] Címen itt egy strukturális egység címét értjük, pl. fejezetcím vagy szakaszcím. Ez áll egy sorszámból (ami például az *Előszó* esetén elmaradhat), a sorszámot pont követi, majd egy széles szóköz (*en-space*) után jön a cím szövege. A szöveget sosem zárja pont! Angolszász tipográfiában szokásos a szöveget a bal margón kezdeni, és az őt megelőző sorszámot kitolni a margóra. A magyarban ez nincs megengedve.
– *Címek betűtípusa.* [3, 23. oldal] Bár régebben a cím szedésére (a méretnövelésen kívül) semmilyen formázást nem alkalmaztak, manapság a NAGYBETŰ (verzál), a KISKAPITÁLIS, a **félkövér**, a *dőlt* és a ***félkövér dőlt*** betűváltozatok is használhatók. A félkövér betűknek van azonban egy nyújtott (*extended*) változata, melyet tilos használni. (E dolgozat szövegbetűjéből hiányzik ez a változat, ezért az alábbi példa a CM betűcsaládból lett véve: **nyújtott félkövér** és **félkövér**).
– *Kiemelés a főszövegben.* [3, 24. oldal] [22, 44. oldal] A főszöveg legyen álló, nem csupa nagybetűs. A *döntött* (*slanted*) betűkön kívül bármilyen betűváltozat alkalmazható, bár a félkövér és a kiskapitális valaha idegen volt a magyar tipográfiától. A múlt század elején, amikor félkövér egyáltalán nem létezett, a kiemelést *ritkított dőlt* betűkkel oldották meg, de ma ezt már nem használják, mert hígítja az oldal feketeségét. A **félkövér** feltűnő sötét foltokat eredményez, ezért szövegközi kiemelésre nem ajánlott (de hasznos tankönyvekben, mert a tanuló figyelmét rögtön a kulcsszavakra irányítja), továbbá elterjedt leíró listák is szótárak szócikkeinek kiemelésére. Leíró listára példa található az 1.4. alszakaszban. A KISKAPITÁLIST személynevek kiemelésére használják a főszövegben és az irodalomjegyzékben, például KNER IMRE. (Színdarabokban a megszólaló személy nevét csupa kiskapitálissal kell szedni, pl. RÓMEÓ és nem RÓMEÓ.)
– *Megnevezett bekezdések* (*paragraph*). A megnevezett bekezdés a dokumentum legkisebb strukturális egysége. A szöveg a cím után általában azonnal folytatódik, lásd például a 29. oldalon. A cím után tilos pontot tenni ([5, 72. oldal]), helyette egy emelt helyzetű díszjel helyezendő el. Bár a leíró lista (lásd például a 11. oldalon) nem számít megnevezett bekezdésnek, tipográfiájára hasonló megfontolások érvényesek.
– *Behúzás* (*first line indentation*, *indentation*). [22, 40. oldal], [3, 24. oldal] Minden bekezdés, beleértve a megszólalásokat is, behúzással kezdődik. A behúzás mértéke a betűméret vagy annak kétszerese (a sortávolságot nem szabad figyelembe venni). A betűméret másik szokásos elnevezése a négyzet. [22] szerint akkor szükséges a két négyzetnyi be-

húzás, ha a tükörszélesség legalább 20 cc (ejtsd: húsz ciceró), azaz kb. 9,026 cm. [3] más forrásokra is hivatkozik, melyek 24 cc-t javasolnak küszöbértéknek. Minden bal és jobb oldali beljebb kezdés (például a felsorolásoké, lábjegyzeteké és idézeteké) a behúzás egész számú többszöröse kell legyen. A strukturális egységek első bekezdései szedhetők behúzással vagy anélkül (tehát tompán), bár a tompa szedés hátránya, hogy csúnya, ha az egység két rövid sorral kezdődik. Rövid dokumentumok, például prospektusok esetén el lehet tekinteni a behúzástól, ekkor viszont a bekezdések közti függőleges teret meg kell növelni. A cím utáni első sor állhat behúzással és tompán. [21, 286. oldal] szerint a cím utáni egysoros bekezdéseknek és megszólalásoknak és az őket követő többsoros bekezdésnek is tompán kell állniuk, és csak ezután következhet az első behúzott bekezdés.

– *A bekezdés utolsó sora* (kimenetsor, *break line*). [3, 24. oldal] Legyen a behúzás szélessége b, a tükörszélesség t, a két bekezdés közti függőleges tér h, a főszöveg betűmérete pedig m. A bekezdések utolsó sorának hosszára (u) az alábbi korlátozások érvényesek: $b > 0$ esetén $b \leq u \leq t - b$ vagy $u = t$. $b = 0$, $h = 0$ esetén $m \leq u \leq t - 2m$. Ha a főszöveg balra van igazítva, akkor $b = 0$, $h = 0$ esetén $u \leq 3/4 \cdot t$-nek is teljesülnie kell ([22, 46. oldal]). $b = 0$, $h > 0$ esetén vagy $t = u$, vagy a $b = 0$, $h = 0$ esetén fennálló korlátok érvényesek. Ezek a korlátozások csak a bekezdés sortöréseinek variálásával, és a felsőbb sorokban a szóközök széthúzásával tarthatók be. [3] a fentieknek ellentmondó szabályokat is idéz, melyek főleg m együtthatójában térnek el az itt említettektől. [21, 286. oldal] a $^3/_2 \cdot b \leq u \leq t - b$ szabályt tartalmazza.

  Wordben nem lehet kézzel sem kihúzni egy sorkizárt bekezdés utolsó sorát a sor végéig; QuarkXPressben ez lehetséges, de a fenti szabályok betartására kézzel kell ügyelni. LaTeX-hez Donald Aresenau javasolt beállításokat, melyek kis példákon automatikusan biztosítják a helyes tördelést, de nem világos, hogy mely csomagokkal milyen módon akadnak össze.

– *Számozott felsorolás.* Bal oldalt [3, 24. oldal], jobb oldalt a [5, 78. oldal] által előírt forma látható.

I. Tudományterületek
  2. Informatika
    c) Mesterséges intelligencia
      δ) Ágensek
        E) Mobil ágensek

I. Tudományterületek
  2. Informatika
    C) Mesterséges intelligencia
      d) Ágensek
        ε) Mobil ágensek

  E dokumentum a bal oldali mintát követi, de kihagyja a római számokat a hierarchia tetejéről. A fenti formázásoktól tilos eltérni: mind a kiemelést, mind a kis-és nagybetűket, mind a számot lezáró jelet pontosan úgy kell elhelyezni, mint ahogy a mintában szerepel. Ha a felsorolás egy eleme többsoros, akkor a további sorok vagy az első sor szövegével (nem a címkéjével) egy vonalban, vagy – régebbi szokás szerint – közvetlenül a bal margón kezdődnek. Az elemek között nem hagyható függőleges tér, de az egész felsorolás alatt és fölött, továbbá a szintek között igen. A jel és az elem közti vízszintes távolság a szóköz szélességének kétharmada legyen. Ha minden listaelem egysoros, akkor a legfelső szintű címkéknek a bal margón kell kezdődniük, különben pedig a legfelső szintű elemek szövegének épp behúzásnyira kell lenniük a bal margótól. A további szintek minden esetben egy behúzásnyira kezdődnek jobbra az előző szintnél.

– *Számozatlan felsorolás.* Hasonló a számozotthoz, de számok helyett azonos jelek szerepelnek egymás alatt. A jeleket (kivéve a felső gondolatjelet) kissé meg kell emelni, és a főszövegénél kisebb betűfokozattal kell szedni. Az ajánlott hierarchia a következő:

- Tudományterületek
  - Informatika
    - Mesterséges intelligencia
      - Ágensek
        - Mobil ágensek

Egyéb tekintetben a számozott listákról elmondottak érvényesek.

– *A bekezdések közti tér.* [3, 24. oldal] Főleg szakkönyvekben dívó angolszász hagyomány a bekezdések között üres függőleges helyet kihagyni. Ez a magyarban tilos, kivéve, ha a dokumentum rövid és a tipográfus a behúzás elhagyása mellett döntött. Ez a beállítás csak akkor alkalmazható, ha a dokumentum amúgy is tartja a soregyent. Ellenkező esetben a bekezdések közötti teret egy hajszálnyival meg lehet növelni, hogy az oldal utolsó sora a lap alján végződjön.

– *Címrendszer.* [22, 84. oldal] A címek betűmérete a hierarchiában lefelé haladva csökken. Ezen belül a tipográfus szabad kezet kap a betűtípus és a betűméret választásában. Műszaki–tudományos művekben hierarchikusan számozzák a strukturális egységeket, például: „3. fejezet||Európa” (itt || soremelést jelöl), „1.2. Magyarország”, „1.2.5. Idegenforgalom”. A számot mindig pont zárja (ellentétben a [22, 84. oldal]-on mutatott példával), a pont és a címet pedig egy gondolatjel szélességű köz választja el. A címsor alatt kisebbnek kell lennie a kihagyásnak, mint fölötte. A cím nem kerülhet oldal vagy hasáb aljára, legalább két normál sornak követnie kell. Példa utalásokra: „első rész”, „1. fejezet”, „1.1. szakasz”, „1.1.1. alszakasz” (a pont, a szóköz és a kisbetűk használatára kell figyelni). A tartalomjegyzék címeinek szedésében is megmutatkozik a hierarchia, de ennek nem kell utánoznia a főszövegen belüli címeket.

  A LaTeX végtelenre növeli a `\clubpenalty`-t minden cím után, így biztosítja, hogy a címet követő bekezdésben az első sor után nem következhet oldaltörés, tehát tényleg legalább két normál sorral folytatódik az oldal. Ugyanezzel a módszerrel három normál sor biztosítása nem lehetséges.

– *Kimenetoldak.* [22, 81. oldal] A kimenetoldal a fejezet utolsó oldala vagy egyéb olyan oldal, amely nem telik meg teljesen. Irodalmi művekben szokás, hogy a fejezet első oldala nem a lap tetején kezdődik. Ekkor a kimenetoldalon legalább annyi szövegnek kell lenni, hogy fedje az első oldal tetején levő kihagyást. Tudományos művekben arra kell vigyázni, hogy többhasábos kimenetoldal esetén a hasábok egyenlő magasságúak legyenek a kimenetoldalon.

  LaTeX esetén az `\enlargethispage` parancsnak adandó argumentumot próbálgatással lehet meghatározni. `QuarkXPress`-ben a szövegdoboz magasságát a kimenetoldalon kézzel kell beállítani, Wordben pedig az adott oldalra vonatkozó oldalbeállításokat kell módosítani.

– *Hosszú verssorok.* [22, 47. oldal] Bár ilyenek műszaki–tudományos művekben nem fordulnak elő, a teljesség kedvéért megemlítjük, hogy ha egy vers sorai nem férnek el egy sorban, és ez egyéb tipográfiai eszközökkel (pl. a tükörszélesség növelése, betűméret csökkentése) sem oldható meg, akkor a túllógó szavakat a következő sorban balra igazítva kell szedni, szóelválasztás nélkül. A túllógás bal oldali behúzása egységesen akkora legyen, hogy a leghosszabb túllógás éppen a jobb margóig érjen.

– *Függőleges tér kiemelések körül.* A kiemeléseket (listák, képletek, úszó objektumok) a főszövegtől elválasztó függőleges tér általában kisebb, mint angolszász dokumentumokban. A soregyent tartó dokumentumokban szokás a felsorolások körüli kihagyás teljes mellőzése, vagy egy egész sor kihagyása. Egyéb dokumentumokban kb. fél sornyi kihagyás az általános.

2.3. A folyószövegen kívüli részek

– *Táblázatok.* A műszaki könyvek táblázatainak ajánlott tipográfiáját és a felmerülő problémák megoldási lehetőségeit [6, 121–144. oldalak] igen részletesen tárgyalja. Itt csak annyit említünk, hogy az oszlopcímek nagybetűvel kezdendők, és a címekben a mértékegység a mennyiség neve után, attól vesszővel és szóközzel elválasztva áll. [6] szinte összes táblázatában van függőleges vonal, ami kisebb, egyszerűbb, nem kizárólag numerikus táblázatok esetén mellőzhető – e dolgozat is ezt teszi.
– *Ábra címe, képaláírás.* [6, 110. oldal] Az ábra címe az ábra alatt kell, hogy szerepeljen. A címet az ábra száma előzi meg ponttal és szóközzel elválasztva (lásd e dolgozat bármelyik ábráját, például a 19. oldalon). Ha az ábrához magyarázat is tartozik, azt a cím alatt, kisebb betűfokozattal kell feltüntetni. A magyarázat természetesen nem kerül bele az ábrák jegyzékébe. Az ábrán látható szöveg a főszövegnél kisebb betűfokozatú, de minden ábrában és táblázatban azonos legyen. A címet tilos ponttal lezárni ([21, 303. oldal]), még akkor is, ha több mondatból áll.
– *Táblázat címe.* [6, 142. oldal] A táblázat címe a táblázat fölött szerepel. A címet a táblázat száma előzi meg ponttal és szóközzel elválasztva, vagy pont nélkül egy sorral följebb. Ha a táblázathoz magyarázat, megjegyzés vagy lábjegyzet tartozik, azt a táblázat alatt, a címnél kisebb betűfokozattal kell feltüntetni. A magyarázat természetesen nem kerül bele az ábrák jegyzékébe. A táblázat szövege a főszövegnél kisebb betűfokozatú, de minden ábrában és táblázatban azonos legyen. A címet tilos ponttal lezárni ([21, 303. oldal]), még akkor is, ha több mondatból áll.
– *Lábjegyzetek*[*]. [5, 116–118. oldal][**]

1. Csillagok és arab számozás egyaránt használható a lábjegyzet jelölésére.
2. A csillagos számozás minden oldalon újrakezdődik, az arab számozás pedig fejezetenként (*chapter*).
3. A lábjegyzetjel szóköz nélkül követi a szót, amire vonatkozik.
4. A lábjegyzet külön mondat, tehát nagybetűvel kell kezdeni, és általában ponttal kell befejezni.
5. Ha egy műben a szerkesztő és a szerző is helyez el lábjegyzeteket, a szerző lábjegyzeteit arab számokkal, a szerkesztőjét pedig csillagokkal érdemes jelölni.
6. A lábjegyzetek szövege és a főszöveg között elég egy üres sort hagyni, vonalra nincs szükség (bár nem hiba, ha van).
7. A főszöveg és egy hosszú lábjegyzet előző oldalon félbemaradt részének folytatásával induló lábjegyzetblokk között vonalat kell elhelyezni (melynek hossza a tükörszélesség harmada vagy negyede legyen).
8. Rövid lábjegyzetek egymás mellé, egy sorba tördelhetők.
9. Táblázat vagy ábra lábjegyzetei közvetlenül alatta, ne az oldal alján jelenjenek meg.
10. Ha nagy betűmérettel szedett szövegrészt látunk el lábjegyzettel, a lábjegyzetjel ne legyen nagyobb a normál betűmérethez tartozónál.
11. A lábjegyzet kezdődhet beütéssel (azaz első sora beljebb kezdődik) vagy listaszerűen (a lábjegyzetjeltől eltekintve az első és a második sor egymás fölött kezdődik).
12. A főszövegen belüli lábjegyzetjel előtt és után nincs szóköz, az oldal alján található lábjegyzetjel után pedig egy rövid szóköz van.
13. Ha túl sok csillagos megjegyzés kerülne egy oldalra, akkor más jelre kell áttérni, például: * ** *** † †† †††.

[*] Másnéven lapalji jegyzet, angolul *footnote*.
[**] Ha semelyik oldalra sem kerül túl sok lábjegyzet, számozás helyett csillagok is használhatók. Többsoros lábjegyzet nem a margón folytatódik.

14. [22, 39. oldal] szerint az oldal alján levő lábjegyzetet jelző szám nem csökkentett méretű, pont és szóköz következik utána. Ez ellentmond [5, 116–118. oldal]-nak.
15. Alul az első csillag a behúzástól jobbra kezdődjön, de a további csillagok már balra nyúljanak ([21, 308. oldal]).

– *Összefüggő művek élőfeje.* [5, 195. oldal], [22, 86. oldal] Nem minden mű rendelkezik élőfejjel: az oldal alján található sorszám (lehet középre vagy kívülre igazítva) nem igazi élőfej (és nem is élőláb), és a szedéstükör méretébe sem számít bele. Azon művekben, melyekben van élőfej, az oldalszám minden esetben kívülre kerül, az élőfej szövege pedig vagy középre vagy belülre. Az élőfej alatti vonal opcionális. Az élőfej betűtípusa megegyezik a szövegbetűjével, esetleg annak variációival (kisebb fokozat, nagybetűsítés, dőlt betű, vagy kiskapitális), de a betű semmiképpen sem lehet döntött. A bal oldali élőfej magasabb szintű címet tartalmaz a jobboldalinál (e dolgozatban például bal oldalt szakasz címek, jobb oldalt pedig alszakaszcímek olvashatók). A bal oldali élőfej az adott oldalon kezdődő legalsó egység, a jobboldali pedig a legfelső egység nevét tartalmazza.
– *Élőfej szótárak, lexikonok esetén.* Az oldalszám középre kerül, a szélekre pedig címszavak: a bal oldal bal szélére az előző oldal befejezetlen címszava, vagy – ha ilyen nincs –, akkor az oldal kezdő címszava kerül, a jobb oldal bal szélére az oldal kezdő címszava, a jobb szélekre pedig az oldalon elkezdett utolsó címszó (lásd [7] élőfeje). Egyes lexikonokból az élőfejek belső címszavai hiányoznak, ekkor az oldalszám kerül a helyükre.
– *Üres oldal.* [3, 24. oldal] Az üres oldalakon oldalszám sem szerepelhet.
– *Páros és páratlan oldalak.* Egy kinyitott folyóirat vagy könyv a kötéstől balra levő oldalai párosak, a jobbra levők pedig páratlanok. [5, 23. oldal] leírja, hogy a mű mely egységeit kell új oldalon, és melyeket új páratlan oldalon kezdeni. Ez utóbbiak előtt lehet, hogy egy teljesen üres (még oldalszámot sem tartalmazó) páratlan oldalt kell kihagyni.
– *Könyv oldalszámozása.* [5, 191. oldal] A magyar hagyományok szerint (a könyv kinyitása után) a borítóval átellenes oldalon, vagyis a szennycímoldalon kezdődik, arab 1-es sorszámmal (az előzéklapok nem kapnak sorszámot). Az első néhány oldalon nem található oldalszám. Egy tipikus könyv esetében ezek ([5, 31. oldal] alapján) a szennycímoldal (1), a sorozatcímoldal (2), a címoldal (3) és a copyrightoldal (4). A tartalomjegyzék az 5., az első számot tartalmazó oldalra kerül. A tartalomjegyzék után az előszó és a köszönetnyilvánítás szokott következni, és csak ezután indul a főszöveg (ami általában a bevezetéssel kezdődik). Az amerikai gyakorlat ezzel szemben a főszöveget 1-től, a szennycímoldalt pedig római i-től kezdi számozni ([5, 191. oldal] felsorol egyéb szokásokat is). Ha a könyv számítógéppel készül, mindkét séma könnyen megvalósítható, ezért a magyar hagyományok követése ajánlott.
– *Cikkek oldalszámozása.* A szerző 1-es arab számmal kezdje a saját cikkét. A szedő majd átszámozza, mikor már megvan a cikk végleges helye a folyóiratban.
– *Tárgymutató.* Műszaki könyvek tárgymutatójának összeállításával és annak tipográfiájával [6, 209–227. oldal] foglalkozik. Az ott említett rendezési szabályoktól a mai gyakorlat a következőkben tér el: a betűhalmaz-elmélet (vagyis hogy a címszó besorolásakor szóközök nem számítanak) az általános, de személynevekben számít a szóköz, továbbá a többszíitű címszó (például a *csomagok*on belüli *babel*) egyes szintjei lexikografikusan kerülnek besorolásra; az alsóbb szinteken használható a ~ jel, ennek besorolási értéke a címszó eggyel feljebbi szintje, például a *tipográfia* szinten alatti mutatók ~ja besorolási értéke *mutatók tipográfiája* (gépi besorolásnál az utolsó ékezet lemaradhat). A mutatók rendezéséről [5, 161–169. oldal] részletesen ír.

A legfontosabb vizuális különbség az angol és magyar mutatók között, hogy a címszó és az első oldalszám között a magyar mutatóban nincs vessző. További különbségek adódnak az eltérő ábécérendből (lásd a 3.4.1. alalszakaszban), például a kezdőbetűk szerint eltérően kell csoportosítani (magyarban van *E, É* és *Sz* csoport stb.).

– *Helyesírás-ellenőrző.* Manapság már nélkülözhetetlen segítség a helyesírás-ellenőrző a gépelési és egyéb betűhibák javításában.

A Word a dolgozat írásának idején legfrissebb verziójába épített magyar ellenőrző a *Morphologic* Helyes-e? programjának egy régi, lebutított és kis szótárú változata. Az egyébként igen nagy tudású program külön termékként, kereskedelmi formalomban kapható. A Helyes-e? QuarkXPress-es verzióját is külön meg kell vásárolni. A LaTeX források ellenőrzésére használatos három ingyenes ellenőrzőről és azok mindennapi használatáról részletesen tájékoztat a 3.6. alszakasz.

– *Szinonimaszótár.* A programba integrált szinonimaszótárak és a tezauruszok segítik a választékos fogalmazást.

A Wordhöz a *Morphologic* Helyesebb szoftverét külön meg kell vásárolni, a másik két programhoz nem ismeretes integrált szinonimaszótár.

Példa megnevezett bekezdésre ♦ A bekezdés szövege a címmel azonos sorban folytatódik. Az ilyen strukturális egységek címét bekezdéses címnek is nevezik.

2.4. A vizsgált programok összehasonlítása

A 3. táblázat az 1.5. alszakaszban bemutatott három programot hasonlítja össze a 2. szakaszban ismertetett szempontok alapján. A LaTeX oszlop a magyar.ldf régi, 1.4-es verziójával, a diplomatervezés során kifejlesztett MagyarLaTeX szoftvergyűjtemény nélkül készülő dokumentumokat írja le, a MagyarLaTeX oszlop már az e diplomatervben készített eszközök használatát is megengedi. A Word oszlopban a Microsoft Word legújabb, magyar nyelvű verzióját jelöli. Az összehasonlítási szempontok a magyar és az angolszász tipográfiai különbségeire koncentrálnak.

Az egyes cellákba írt rövidítések az alábbiak:

- **-** A programban nem hozható létre olyan formai elem, melyre ez a szabály vonatkozik.
- **0** A program nem alkalmas e szabály betartására, csak hibás mű készíthető vele.
- **k** A programban ugyan be lehet tartani ezt a szabályt, de minden egyes előfordulásakor erről *kézzel* kell gondoskodni. A másoktól kapott dokumentumot figyelmesen át kell nézni, és a szabályt minden egyes helyen érvényre kell juttatni.
- **e** A program *e*gyszerűen beállítható úgy, hogy e szabály magától érvényre jusson.
- **p** A program egyszerűen beállítható, de az adott szabály betartása *p*roblémás, például azért, mert ütközik más szabályokkal, vagy csak speciális feltételek fennállása esetén tartható be.
- **h** A program beállítása úgy, hogy e szabály érvényre jusson, *h*osszadalmas és/vagy nagy szakértelmet igénylő művelet.
- **s** Létezik olyan *s*egédprogram, vagy kiegészítő stílus, ami segít a szabály automatikus betartásában. A segédprogramot általában külön kell beszerezni, és lehet, hogy fizetni is kell érte.
- **1** A program alapértelmezés szerint a magyar kívánalmaknak megfelelően működik.

Van olyan cella, ahol több rövidítés is helytálló, ekkor az összeset megadtuk, csökkenő fontossági sorrendben.

3. táblázat. Az vizsgált programok magyar szolgáltatásai

Szempont	Word	QuarkXPress	LaTeX	MagyarLaTeX
összes magyar betű	1	h	1	1
alacsony ékezetek	k	k	p	p
szóelválasztás	1	s	1	1
pottyan elválasztása	0	0	k	ks
nem törhető szóköz	k	k	k1	k1
ábécérend	0	0	0	s
ligatúrák	k	0	1	1
a/az generálása	0	0	1	1
toldalékolás	0	0	0	1
utalás strukturális egységekre	0	0	0	1
kötőjel, gondolatjel	k	k	k	k
gondolatjel szerzőpárokhoz	k	k	k	1k
idézőjelek	k	h	k	1k
a láb jele	1	1	0	0
hüvelyk, fok	1	1	1	1
szögperc, szögmásodperc	1	0	1	1
igazi kiskapitális alapfontban	0	0	1	1
ugráló számjegy alapfontban	0	0	1	1
magyar dátum	1	0	1	1
magyar dátum sok formában	0	0	0	1
magyar dátum toldalékolva	0	0	0	1p
zárójelhasználat	k	k	k	k
csúcsos zárójel alapfontban	1	0	1	1
keskeny szóköz írásjel előtt	k	k	k	1
egységes köz írásjel után	1	1	e	1
egalizálás font szerint	1	1	1	1
egalizálás módosítható	0	1	1	1
ékezetes betűk egalizálása	0	0	0	s
optikai sorkizárás	0	0	h	h
címsorok optikai igazítása	0	k	k	k
matematikai képletek	p	0	1	1
kémiai képletek	1	p	1	1
magas törtek	1	-	k	k
tizedesvessző képletben	k	-	k	1k
képletek elválasztása	0	-	1	1
extra + képlet új sorában	0	-	h	1
képleten belüli helyes közök	h	-	e	1
mértékegységek	k	k	k	k
fix szóköz megszólalás elején	0	k	k	1k
soregyen	kp	k	kp	kp
címek magyar számozása (pont)	0	-	1	1
címek magyar szedése	k	k	1h	1
betűritkítás	1	1	0	sp
díszpont bekezdéses cím után	k	k	kh	1
tompa behúzás cím után	k	k	1	1
több tompa bekezdés cím után	k	k	k	k

3. táblázat. (folytatás az előző oldalról)

Szempont	Word	QuarkXPress	LaTeX	MagyarLaTeX
a kimenetsor hosszkorlátjai	0	k	k	k
hierarchikus listaszámozás	k	kh	e	1
tér listaelemek között	k	k	e	e
tér kiemelések körül	k	k	e	e
számozatlan felsorolás	k	kh	e	1
oldaltörés tiltása cím után	e	0	1	1
hosszú kimenetoldalak	k	k	k	k
azonos hasábmagásság kimenetoldalon	kp	k	kp	kp
hosszú verssorok	k	k	k	k
képaláírás magyar szedése	k	k	1	1
táblázatcím magyar szedése	k	k	1	1
csillagos lábjegyzetek	0	k	e	1
lábjegyzetszám magyar szedése	0	k	0	1
szerző és szerkesztő jegyzetei	0	k	k	1
feltételes vonal lábjegyzetek felett	0	k	kp	1p
fix méretű lábjegyzetjel	0	k	k	1
lábjegyzet-csillag behúzása	0	k	0	0
élőfej	e	k	1p	1
lexikonok élőfeje	ep	k	e	e
üres oldal ne legyen számozva	k	k	k	k
páros és páratlan oldalak	1	k	1	1
könyv oldalszámozása	e	e	e	e
magyar tárgymutató generálása	0	-	0	s
magyar irodalomjegyzék	k	k	k	1
helyesírás-ellenőrző	1	s	s	s
szinonimaszótár	s	0	0	0

3. A LaTeX magyarítása

Az előző fejezetben összehasonlított programok közül e sorok írója azért választotta a LaTeX magyarítását, mert az szabad szoftver, tehát bárki számára elérhető mind a forráskód, mind a fejlesztői dokumentáció, továbbá a LaTeX és csomagjainak fejlesztői szívesen válaszolnak a kérdésekre. Az sem elhanyagolható szempont, hogy programozásához nem szükséges erős számítógép sok memóriával, szabad lemezterülettel vagy előírt operációs rendszerrel: egy rövid LaTeX dokumentum néhány másodperc alatt lefordul már a 10 évvel ezelőtt kapható PC-ken is. E sorok írója mások számára is hasznos kiegészítőt kívánt alkotni, és a Microsoft Word, QuarkXPress vagy egyéb nagy kereskedelmi programok fejlesztésébe való bekapcsolódásra egy Magyarországon elkészített, féléves diplomaterv keretében nincs reális lehetőség. Azonnal nyilvánvaló volt, hogy az e programok menürendszerében a tipográfusok és a szedők számára felkínált lehetőségekkel a teljes magyarítás (pl. a határozott névelő generálása) nem vihető végbe. Lehetséges választás lett volna a LaTeX helyett valamely magyar projektben való munka, például GrailTeX (`http://tipogral.hu/`), de mivel e sorok írója korábban is foglalkozott a LaTeX-hel, biztosabb volt e vonalon maradnia.

Az előző fejezetben említett szakirodalom tanulmányozása (különösen [21] tipográfi-

ai része: 251–320. oldalak) és a konzulensek véleménye alapján nyilvánvaló volt, hogy a LaTeX magyarítása óriási munka, sok igen nehezen megoldható részproblémával, de a könnyebb feladatok megoldása, az eredmény tesztelése és a LaTeX hivatalos verziójába való integrálása is túlmutat egy diplomatervezés keretein. Ezért e sorok írója azt a célt tűzte maga elé, hogy a régóta ismert, legfontosabb problémákat megoldja, az egyszerűbbekkel kezdve, továbbá bővítse a LaTeX-et néhány bonyolultabb, nehezen beprogramozható, a magyar dokumentumok készítését segítő szolgáltatással. Hogy pontosan melyek ezek, kiderül az 1.3. alszakaszból és a 3.2.3. alalszakaszból. E sorok írója fontosnak tartotta, hogy az elkészült megoldást mások is használni tudják, ezért MagyarLaTeX szoftvergyűjtemény néven feltette a webre (`http://www.math.bme.hu/latex/`). A LaTeX-et magát a CTAN-ről (*Comprehensive TeX Archive Network*) lehet letölteni, a cím `http://www.ctan.org/`, de akinek egy könnyen telepíthető disztribúcióra van szüksége, az vagy rakja fel Linux alatt a teTeX csomagokat, vagy töltse le a `http://www.tug.org/tex-live/` címről a TeX live telepítő vagy *demo* CD-jét.

A magyarítás során elkészült legnagyobb méretű és jelentőségű forrásfájl magyar.ldf (részletes leírás a 3.2. alszakaszban). Ez a babel csomagba illeszkedő makrógyűjtemény végzi el a főszöveg formai elemeinek magyarítását az idézőjelektől egészen a fejezetcímekig. Az élőfej, a lábjegyzetek, az úszó objektumok magyar megjelenését is ez a fájl állítja be, sőt hatása túllép a főszövegen, mivel a jegyzékek tipográfiájára is hatással van. Hatásköre így írható le a legpontosabban: a ábécérendbe sorolást igénylő műveleteket és az automatikus szóelválasztást kivéve minden, ami egy magyar nyelvű LaTeX dokumentum lefordításához szükséges, a magyar.ldf-ben van megvalósítva. (A helyesírás-ellenőrző például külön program, mivel az nem szükséges a fordításhoz.) A rendezés azért nem került a magyar.ldf-be, mert a TeX nem kínál fejlett string- és listakezelő nyelvi megoldásokat, ezért a szöveges információ rendezése TeX-ben csak igen körülményesen lenne megoldható, és ráadásul kb. duplájára növelné a fordítási időt. A mutatók rendezésére használt husort.pl bemutatása a 3.4. alszakaszban, az irodalomjegyzéket rendező és szedő BibTeX magyar stílusfájljának, a huplain.bst-nek ismertetése pedig a 3.5. alszakaszban található. Az elválasztási szabályokat leíró minták (`*hyph*.tex`) azért nem kerülhettek a magyar.ldf-be, mert ezekre a formátumgeneráláskor van szükség. (A formátumgenerálás a TeX-fordítást gyorsítja. Manapság a sebességnövekedés alig észrevehető, de az 1970-es évek végén még jelentős volt, és történeti okokból megmaradt a mai napig.)

A LaTeX magyarítására korábban is történtek próbálkozások, ezek közül a legjelentősebb a már említett magyar.ldf 1.4-es verziója, melynek készítésében igen sokan közreműködtek ([18] tartalmaz egy felsorolást). A diplomatervezés során elkészült új verzió is ezen a változaton alapszik, de a szolgáltatások nagy része újraimplementálásra került, hogy tágabb feltételek között működhessen, és ne akadjon össze számos népszerű csomaggal. A magyarításba beletartozik a felhasználói felület lokalizálása és magyar nyelvű leírások megjelentetése is, ezek felsorolása megtalálható a 3.1. alszakaszban – e diplomaterv nem alkotott újat ezen a téren. Az automatikus szóelválasztás megvalósításával is többen foglalkoztak (lásd a 3.3. alszakaszban), de mivel ez inkább nyelvészeti, mint informatikai ismereteket igényel, továbbá a születő megoldások megfelelő tudományos alaposságúak, és a gyakorlatban is használhatónak bizonyultak, e diplomaterv csak épít rájuk, és nem állít ki alternatívát. Bár egy magyar nyelvű helyesírás-ellenőrző fejlesztése érdekes, és igen nagy kihívást igénylő feladat, a diplomaterv témáját csak érinti. Három megoldás tölthető le ingyenesen: a kereskedelmi MSpell, a szabad szoftver MagyarISpell, és utódja, a HunSpell. Ezeket a 3.6. alszakasz részletesen bemutatja, és ismerteti a diplomatervezés során kifejlesztett ispell.pl segédprogramot is, melynek segítségével mindhárom ellenőrző használható ugyanazon a számítógépen LaTeX dokumentumokban

a szavak helyességének ellenőrzésére és gépelési hibák kiszűrésére. Korábban egyetlen LaTeX-es magyar indexrendező segédprogram létezett, ez mami, amely dokumentálatlan, és a mai szoftverkörnyezetben csak nehézkesen használható, és nem foglalkozik a 3.4. alszakaszban ismertetett szabályok többségének betartásával, ezért a diplomatervezés során kifejlesztett husort.pl indexrendező fejlesztéséhez a mami nem járult hozzá. Az interneten elérhető a magyar.xdy, az általános Xindy indexrendező magyar stílusfájlja, de ez is esetleges, elsietett megoldásokat tartalmaz, és nem a teljesség igényével készült, és a már említett szabályoknak is csak egy kis részét tartja be. Bár kiindulási alapnak a magyar.xdy jó lett volna, e sorok írója mégis úgy döntött, hogy egy teljesen új, se a makeindex-től, se a Xindy-től nem függő Perl szkriptet ír, ami a kevés függősége miatt jól hordozható. A bibliográfiai adatbázis-kezelőkhöz korábban nem volt ismeretes magyar nyelvű stílusfájl. Természetesen léteznek egyéb, nem publikált megoldások is a LaTeX magyarítására, például az egyes kiadóknak vannak belső használatra készült LaTeX-stílusfájljai. E diplomaterv nem foglalkozik velük, mivel ezek a felhasználók széles köre számára nem elérhetők.

A fentiek alapján elmondható, hogy a diplomatervben elkészített szoftverek használják (betöltik vagy segédprogramként meghívják) a létező megoldásokat, de ha egy vizsgált területen nincs kielégítő megoldás, akkor a diplomaterv teljesen újat implementál. Egyedül a magyar.ldf esetén történt 9 kB-nyi kód átvétele az előző verzióból. A fejlesztés során nem jelentettek segítséget a korábbi programok felhasználói vagy fejlesztői leírásai, hanem a magyar tipográfiai hagyományt ismertető művekhez (pl. [3, 22, 5, 21]) kellett visszanyúlni, és a szabályok értelmezése után azok betartását egy új szoftverben megvalósítani. E művek közül egyedül [3] koncentrál a LaTeX-re, [22] próbál programfüggetlen maradni, de erős kiadványszerkesztő-szemléletű (szerzője QuarkXPress-t használ), [5] is független, de csak általános könyvkiadással foglalkozik, [21] kifejezetten a műszaki könyvekre koncentrál, de még a számítógép nélküli könyvkészítés módszereit ismereteti.

3.1. Dokumentáció és felhasználói felület

Angol nyelvtudás nélkül igen nehéz a LaTeX-et használni. A TeX parancsszavak és a külső segédprogramok a kezdetektől fogva angol nyelvűek, ezek lokalizálásra még nem voltak törekvések. (Más a helyzet pl. a ConTeXt esetén, melynek parancsszavait is lefordították angolra, hollandra, csehre és még más nyelvekre, lásd a texexec program `--interface` kapcsolóját; továbbá egyes, a segédprogramok által megjelenített üzeneteket is lefordították, lásd a texutil program `--interface` opcióját.) A LaTeX-hel dolgozó szerző magyar nyelvű dokumentumot készít egy angol szoftverkörnyezetben. A teljes lokalizáció amúgy is lehetetlen volna a LaTeX csomag decentralizált fejlesztése miatt. A parancsszavak lokalizációja egyenesen káros lenne, mert egyrészt csak zavart okozna, ha egy magyar szerző nemzetközi folyóiratban szeretne publikálni, továbbá a lokalizáció által bevezetett új parancsok előre nem látható módon összeakadnának bizonyos csomagokkal. Az Omega projekt keretében folyamatban van a LaTeX szintaxisát kiváltó XML alapú bemeneti formátum kidolgozása, melyben semmi akadálya nincs a *tag*ek lefordításának, vagy egy olyan XML-szerkesztő kifejlesztésének, ami az angol *tag*nevek helyett magyar entitásokat jelenít meg. Világos azonban, hogy egy kötött, teljesen XML alapú szintaxis sosem fedheti a teljes LaTeX parancskészletet, mivel az előbbi struktúráját és a megengedett *tag*eket előre rögzíteni kell, míg az utóbbira csak időben változó, decentralizált definíció adható: „LaTeX dokumentum mindaz, amit a standard TeX-disztribúciók legfrissebb változatában található `latex` parancs lefordít, esetleg a CTAN-ről letöltött külső csomagok felhasználásával”. Fontos tervezési alapelv, hogy kell egy közös nyelv, amelyen a LaTeX csomagok megértik egymást: például az *a* csomag a *b* csomag `\c` parancsát mindig meghívhassa

\c néven, függetlenül attól, hogy a parancsszavak nyelve mi volt a *b* csomag betöltésekor. (A múlt században megtörtént eset, hogy az Excel táblázatkezelő függvényneveit is lefordították, az eredeti név megtartása nélkül, ezért az Excel makrócsomagokat *minden* nyelven külön meg kellett írni. Ezt a hibát nem szabad a LaTeX lokalizálásakor elkövetni.

A parancssori kapcsolók és a hibaüzenetek lefordítása azt a problémát veti fel, hogy egyes esetekben a LaTeX-et más programok futtatják (kapcsolókkal), és bizonyos hibaüzenetek szó szerinti megjelenésére számítanak (ilyen program például a MagyarLaTeX részét képező lafmtgen.pl). Épp ezért mindig meg kell tartani az eredeti parancssori kapcsolókat, és biztosítani kell egy olyan kapcsolót, ami a program által kiírt üzeneteket visszaállítja az eredeti (angol) nyelvre. E sorok írójának nincs tudomása olyan magyar vagy más nyelvű projektről, ami a LaTeX szoftvereinek lokalizációját tűzné ki maga elé; ez amúgy is óriási munka lenne, és kérdéses, hogy ezzel arányban áll-e a felhasználók produktivitásának növekedése. Jelenleg a haladó felhasználóról, aki opciókat használ, és megérti a hibaüzeneteket, a programok fejlesztői feltételezik, hogy amúgy is tud angolul.

A leírások, könyvek, dokumentációk, folyóiratok és konferenciák tekintetében már nem egyeduralkodó az angol nyelv. A CTAN-en számos leírás német, francia stb. változata is megtalálható, továbbá a MagyarLaTeX honlapjáról letölthető a 78 perc alatt elolvasható LaTeX-bezető ([17]) magyar fordítása, a plain TeX-ről ([2]) és a LaTeX-ről ([23]) is megjelent magyar nyelvű könyv, és a LaTeX-ről a közeljövőben jelenik megy új, több, mint 750 oldalas könyv ([24]) magyarul. Számos egyetemi kurzusnak, például a BME-n oktatott *Dokumentumszerkesztés...* -nek is van magyar nyelvű segédanyaga LaTeX-hez. Ezen művek tanulmányozásával a teljesen kezdő felhasználó is eljuthat egy igen erős szintre, a LaTeX programozásához és profi stílusfájlok írásához viszont már szükséges az angol nyelvtudás, hogy a CTAN-en elérhető angol nyelvű referenciákat és a programok forráskódjának kommentjeit is értse az illető.

A dimplomatervezés során e sorok írója nem vállalt a felhasználói felület lokalizációját érintő feladatokat. Az elkészült programok és a MagyarLaTeX egyéb programjai is – illeszkedve az általános LaTeX-es hagyományhoz – angolul kommunikálnak a felhasználóval. Természetesen a MagyarLaTeX készülőben levő, felhasználók számára íródó leírása magyar nyelvű (de a magyar.dtx-ben található fejlesztői leírás már angol), ennek egyes részletei megtalálhatók [24]-ben is.

3.2. A LaTeX babel csomagjának magyar.ldf modulja

A LaTeX-en belül az angoltól eltérő nyelv használatát a babel csomag könnyíti meg ([1]). A csomag tartalmaz egy általános keretrendszert, nyelvfüggő modulokat (`.ldf` fájlok) és nyelvfüggő elválasztási mintákat (`*hyph*.tex`). Erős korlátozások mellett bizonyos nyelvek esetén (a magyar nem ilyen) a babel használható plain TeX-hel is. Az elválasztási mintákról a 3.3. alszakaszban található leírás, ez az alfejezet a magyar.ldf-fel, a babel magyar nyelvet támogató moduljával foglalkozik. A magyar.ldf a babel legújabb, 2001-ben kiadott 3.7-es verziójához készült, régebbi verziókkal nem működik.

Egy `.ldf` fájl az adott nyelven való szedést lehetővé tevő összes makrót tartalmazza. Szinte minden `.ldf` fájlban szerepelnek a generált feliratok (pl. „Tartalomjegyzék”), az elválasztási minták kiválasztása, a szóelválasztás egyéb paramétereinek, például a szó elején és végén történő elválasztásának a letiltása, az aktuális dátumot generáló parancs, a nyelvben használt speciális szimbólumokat létrehozó parancsok, és az ugyanerre a célra szolgáló aktív karakterek.

A diplomatervezés legnagyobb feladata a magyar.ldf kifejlesztése volt. A fejlesztés alapjául szolgáló 20 kB-os 1.4-es verzióból a tervezés során született meg a több, mint 190 kB-

os 1.5-ös változat. Az alábbiakban csak a tervezési és implementációs szempontból érdekes új szolgáltatásokat ismertetjük. Az összes új szolgáltatásról a 3. táblázat két jobb szélső oszlopának összevetésével tájékozódni, továbbá a 3.2.3. alalszakasz is felsorolja ezeket a szolgáltatásokat aszerint rendezve, hogy mely betöltési opcióval aktiválhatók. A megjelenés alatt álló [24] részletes felhasználói leírást tartalmaz mind a betöltési opciókról, mind a szolgáltatásokról.

A magyar.ldf alábbi részei lettek az 1.4-es verzióból lényeges változtatás nélkül átvéve: a `captions=` opció által magyarított feliratok, a `hutoday=` opcióval engedélyezhető egyszerű dátumgenerálás, az úszó objektumok aláírásában a magyar számozást előidéző makrók (lásd a `tablecaptions=` és `figurecaptions=` opcióknál), a LaTeX alap dokumentumoszályainak (book.cls, article.cls, report.cls, letter.cls) élőfejében (`\ps@...`) levő számozás magyarítása (lásd a `classmod=` opciónál), ugyanitt a `\chapter` és `\part` strukturális egységeket nyitó cím szedését végző parancsok magyarítása. Az átvett részek összmérete 9 kB, a magyar.ldf fennmaradó több, mint 180 kB-ja e sorok írójának új és eredeti munkája (természetesen ehhez sok ötletet merített a régi magyar.ldf-ből, de más `.ldf` fájlokból és egyéb LaTeX csomagokból is).

A programozási nyelv a magyar.ldf esetén természetesen a TeX beépített makrónyelve, kiegészítve a LaTeX formátumban és a babel keretrendszerban definiált parancsokkal. Ez a makrónyelv számos meglepetést tartogatott: az egyszerűnek látszó feladatok (pl. egy kiszedendő szöveg első betűjének megkeresése a határozott névelő megállapításához) igen szövevényes kódot eredményeztek, míg másokat (pl. a matematikai térközök csökkentése) a várakozással ellentétben igen egyszerűen meg lehetett oldani. Sajnos nem az egyszerű megoldások domináltak, és e sorok írója sokszor órákat töltött el egy egyszerű karaktercserélő vagy prefix kód beolvasó rutin implementálásával, amely C nyelvben néhány perc alatt lekódolható lett volna. Igazán újszerű algoritmust a magyar.ldf nem tartalmaz, pl. a névelők, a ragozás és a dátum generálása mind feladhatók egy kezdő programozáskurzus első félévében. 1998 és 2004 között a sürgető igény ellenére talán azért nem vállalkozott senki e feladatok megoldására, mert a TeX makrónyelvére való lefordításuk nemcsak sok türelmet, hanem egy szokatlan kódolási stílus és gondolkodásmód elsajátítását igényli. Sajnos nem ismeretes olyan könyv vagy egyéb leírás, amelyet elolvasva egy haladó LaTeX-felhasználó (*power user*) beléphet a TeX alacsony szintű programozásának szürreális világába. Ebből a szempontból [10] tanulmányozása a legcélravezetőbb, a bátrabbak pedig végigolvashatják a LaTeX kernel (latex.ltx) forráskódját vagy ennek dokumentált változatát (`.dtx` fájlok a CTAN `macros/latex/base` könyvtárában), továbbá a standard dokumentumosztályok forráskódját (pl. book.cls) is. A programozói gondolkodásmód megalapozása és felépítése, továbbá az alapvető implementációs minták és az ügyes trükkök tematikus bemutatása hiányzik ezekből a művekből. E sorok írója fokozatosan, kísérletezéssel szerezte meg a magyar.ldf megírásához szükséges ismereteket. A tanulási folyamat során nélkülözhetetlenek voltak más LaTeX csomagok forrásai (gyakran úgy kapott segítséget, hogy az összes telepített LaTeX csomag forrásában rákeresett néhány kulcsszóra). A magyar.ldf nem készülhetett volna el, ha a LaTeX és kiegészítői nem lennének szabad szoftverek.

3.2.1. Betöltési opciók és kompatibilitás

Az új magyar.ldf biztosít ún. betöltési opciókat, melyek a magyar.ldf betöltésekor jutnak érvényre, és befolyásolják a magyar – és néhány esetben egyéb – nyelvű szöveg megjelenését az egész dokumentumban. A betöltési opciók fő célja, hogy a tipográfus választhasson azokon a területeken, melyekről a magyar hagyomány nem rendelkezik

egyértelműen. Az opciók további célja, hogy a magyar.ldf kompatíbilis maradjon az előző verziókkal (ha nem adunk meg opciót, akkor kompatibilitási módban töltődik be, lásd lent a `defaults=over-1.4` alapbeállítást).

Az opciók a `\def\magyarOptions{`*opciók*`}` vagy a `\PassOptionsToPackage{`*opciók*`}` hívással adhatók meg, ahol *opciók* egy vesszővel elválasztott, *kulcs=érték* elemekből álló opciólista. E dolgozat az alábbi opciókkal készült: `defaults=hu-min`, `postdescription=diamond`, `labelenums=hu-d`, `hyphenation=huhyphn`.

A magyar.ldf korábbi verziója híres volt arról, hogy igen sok csomaggal összeakadt (a legfontosabbak: az *AMS* osztályok, a graphics, a hyperref csomagok és a tételszerű környezetek számos csomagja) vagyis hogy a dokumentum nem fordult le, ha a magyar.ldf és a csomag is be volt töltve; esetleg lefordult ugyan, de nem a specifikációnak megfelelő dokumentum keletkezett. Az összeakadások oka az, hogy a LaTeX és a csomagok közös névteret használnak, továbbá nem egy jól definiált és pontosan dokumentált interfészen kommunikálnak egymással, hanem a másik forrásában található makrók felülírásával igyekeznek érvényre jutni. (Más lehetőség nincs, a TeX architektúrája és a központosított tervezés hiánya csak ezt teszi lehetővé.) Ha két csomag egyidejűleg írja felül a LaTeX egy parancsát, akkor a végeredmény függ a csomagok betöltési sorrendjétől, de az is lehet, hogy az egyéb, fel nem térképezett függőségek miatt az adott parancs egyáltalán nem fog működni. Az okosabb csomagok (például a hyperref) ellenőrzik más, konkrét csomagok jelenlétét, és saját működésüket ehhez igazítják, így úszva meg az összeakadást – de a probléma megoldása nem ez, hanem egy összeakadás-mentes architektúra kidolgozása lenne, amely bővelkedik interfészekben és kampó eljárásokban.

A magyar.ldf okos csomagként viselkedik: módosításait csak akkor lépteti érvénybe, ha a módosítás igénybe vételéhez szükséges csomagok be vannak töltve. Az időzítés is fontos: egyes felüldefiniálásokat azonnal, másokat a prembulum végén, és megint másokat pedig a magyar nyelv minden aktiválásakor végez el. Az okos viselkedésre jó példa a `theoremtitle=` opció által szabályozott tételstílus. Ez akkor működik a tételszerű környezetet létrehozó parancsokat felüldefiniáló theorem, ntheorem és amsthm csomagokkal és azok nélkül is, mivel mindegyikük jelenlétét érzékeli. Elképzelhető azonban, hogy a jövőben egy új csomag jelenik meg, amelyre a magyar.ldf nincs felkészülve, és összeakadnak. Ekkor a felhasználónak döntenie kell, hogy a magyar.ldf-et vagy a másik csomagot tölti be. Az 1.5-ös verzió opciókezelése lehetővé teszi, hogy ne kelljen az egész magyar.ldf-et nélkülözni, hanem betöltési opciókkal ki lehessen kapcsolni a problémás részt. (Természetesen ekkor a magyar tipográfiai hagyományok automatikus követése sérülhet, de legalább lefordul a dokumentum.) Ha tehát a felhasználó összeakadást érzékel, akkor kikapcsolja a m magyar.ldf összes szolgáltatását (a `defaults=safest` alapértékeit választva), és ha a probléma nem jelentkezik, a sorra visszakapcsolja az egyes opciókat. Ezzel a módszerrel lokalizálni tudja a hibát, és meg is tudja kerülni, anélkül, hogy LaTeX-szakértőhöz kéne fordulnia.

Összeakadást nem csak a közös névtér és a parancsok szándékos felüldefiniálása okozhat, hanem az aktív karakterek is, megoldásokkal együtt lásd a 3.2.4. alalszakaszban.

3.2.2. Alapbeállítások

Nem várható el minden felhasználótól, hogy ismerje és tudatosan használja az összes opciót (főleg azért, mert a következő verziókban új opciók megjelenése várható), ezért a magyar.ldf több alapbeállítás-készletet is tartalmaz, melyekből a felhasználó kiindulhat, és csak ezekhez képest kell majd egy-egy opciót átállítania.

Fontos volt az átállás megkönnyítése, vagyis hogy az 1.4-es verzióhoz készült doku-

mentumok az 1.5-ösben is leforduljanak; a magyar.ldf alapbeállításai (`defaults=over-1.4`) épp ezt szolgálják. Ez persze nem jelenti azt, hogy a lefordított dokumentum ugyanúgy fog kinézni: az 1.5-ös verzió alapbeállításai néhány szempontból közelebb állnak a magyar tipográfiai hagyományokhoz, mint az 1.4-es változaté. Ezek az eltérések csak a legszükségesebb, fontos és látványos szempontokhoz tartoznak, melyek csak kevéssé befolyásolják a dokumentum terjedelmét. Ha a felhasználó egy régi dokumentum lefordításához kinézetben is az 1.4-es verziót szeretné, akkor a `defaults=compat-1.4`-et érdemes választania. (Nem érdemes viszont a régi magyar.ldf-et betölteni, mert az összeakad sok csomaggal.)

A `defaults=safest` készlet a magyar.ldf összes szolgáltatását kikapcsolja. Tesztelésre és kompatibilitási hibák lokalizálására használatos. A `defaults=prettiest` beállítás minden új szolgáltatást és az angol tipográfiától való vizuális eltérést ömlesztve zúdít a felhasználóra. Használata nem javasolt, mivel bekapcsolja a potenciális kompatibilitási problémákat okozó opciókat is. Új dokumentumok készítéséhez a `defaults=hu-min` ajánlott: ez mindenben igyekszik a magyar tipográfiai hagyományokat követni, de ehhez az alap LaTeX beállításain a lehető legkevesebbet módosít, a lehető legkevesebb makrót írja felül stb., épp ezért használata egyszerre biztonságos, és a kimenet is megfelelő.

A készletek közül a szerző tudásszint, kompatibilitási igény és a hagyományok követésének igénye alapján kiválaszthatja a neki megfelelőt. A kínált készletek a következők:

`defaults=over-1.4` ♦ Ez az alapértelmezés. Az a magyar.ldf 1.4-es verziójának tipográfiájától csak kevéssé, néhány, de fontos ponton tér el. Az új, más csomagokkal össze nem akadó parancsokat elérhetővé teszi, de egyiket sem hívja meg automatikusan. Kijavítja az előző verzió legfontosabb megvalósítási hibáit. Célja, hogy az összes, az 1.4-es verzióhoz készült dokumentum leforduljon vele, és csak kevés, a sor- és oldalhatárokat általában nem érintő látható eltérés legyen az 1.4-es verzióhoz képest. Ha ehelyett a szerző célja az, hogy a dokumentum a magyar tipográfiai szabályokat minél inkább kövesse, akkor helyette a `defaults=hu-min` ajánlott. Ez készlet azért alapértelmezett, mert azok számára készült, akik semmit nem tudnak a magyar.ldf jelentős fejlődéséről az 1.4-es és az 1.5-ös verzió között. Ők ezzel a készlettel egy kompatíbilisebb és kicsit szebb eredményt adó magyar.ldf-et kapnak, anélkül, hogy bármit változtatniuk kéne az általuk készített dokumentumok forráskódján.

`defaults=compat-1.4` ♦ Nem változtat (lehetőség szerint) a magyar.ldf 1.4-es verziójának tipográfiáján, de kijavít számos megvalósítási hibát és más LaTeX csomagokkal való összeférhetetlenséget. Ezt ajánljuk azoknak, akik egy régi dokumentumot szeretnének vizuális változás nélkül az új magyar.ldf-fel újrafordítani. Nincs teljes egyezés, a kompatibilitás a magyar.ldf-nek csak az opciók által befolyásolt részeire terjed ki. Elrejti az 1.4-es verzió után definiált parancsokat is (pl. `\told`, `\emitdate`). Az alábbi hívás az 1.4-es és az 1.5-ös magyar.ldf esetén is majdnem ugyanazt a kinézetet eredményezi: `\def\magyarOptions{defaults=compat-1.4} \usepackage[magyar]{babel}`.

`defaults=safest` ♦ Szinte mindent kikapcsol, hatása olyan, mintha a magyar.ldf-et be se töltöttük volna. Hibakeresésnél érdemes ezt választani. Tehát ha váratlan hibaüzenetet kapunk, vagy ha arra gyanakszunk, hogy a magyar.ldf összeakad valamely másik csomaggal, akkor váltsunk át `defaults=safest`-re, és ha a probléma továbbra is fennáll, akkor nem a magyar.ldf-fel volt baj.

`defaults=prettiest` ♦ Minden új szolgáltatást bekapcsol, a magyar tipográfiához a lehető legjobban illeszkedik, bár még nem felel meg neki teljesen. A magyar.ldf fejlesztése során keletkezett új szolgáltatásokat ez az alapbeállítás aktiválni fogja. Vigyázat! Az itt bekapcsolt szolgáltatások esetleg összeakadhatnak más csomagokkal.

`defaults=hu-min` ♦ Az alapbeállításoktól annyiban tér el, hogy nem az 1.4-es verzióval való kompatibilitásra figyel, hanem az opciókat úgy állítja be, hogy azok ne mondjanak ellent a magyar tipográfiai alapelveknek. E dolgozat is a `=hu-min`-t használja. A dolgozat írása közben számos opció összeakadására fény derült, és a magyar.ldf ezen hibái ki lettek javítva. (Ennek ellenére előfordulhat, hogy a keletkező dokumentum egy olyan ponton eltér az alapelvektől, melyet még nem szabályoz opció.) Ha a szabályok választást tesznek lehetővé, akkor ez az alapbeállítás a legegyszerűbben megvalósítható és legkevesebb kompatibilitási problémát előhozó értékek közül válogat. Ha a magyar.ldf még nem felel meg egy szabálynak, de egy későbbi fejlesztés után már megfelel, akkor a változást ez az alapbeállítás aktiválni fogja. A magyar tipográfiát érintő opciók és jelenleg beállított értékeik a következők: `activespace=safe`, `amstocnumskip=\enskip`, `frenchspacing=yes`, `captions=hu`, `classmod=yes`, `figurecaptions=hu`, `footnote=huplain`, `frenchspacing=yes`, `hyphenmins=22`, `labelenums=hu-d`, `labelitems=hu`, `longcaption=centered`, `mathhucomma=fix`, `mathbrk=fix`, `postpara=diamond`, `postsubpara=star`, `postdescription=dot`, `sectiondot=safe`, `tablecaptions=hu`, `theoremtitle=hu`, `titles=\enskip`.

3.2.3. Az összes opció leírása

Bár ennek a dolgozatnak nem célja, hogy felhasználói leírást adjon a magyar.ldf-hez, mégis felsoroljuk az összes betöltési opciót, mivel ezekből teljes kép kapható arról, hogy a magyar.ldf-ben mi van megvalósítva. (A felhasználói leírás [24]-ben megtalálható.) Az alábbiakban minden opció után az az érték található, amelyet az opció akkor kap, ha őt `defaults=over-1.4` alapértelmezett alapbeállítás mellett kihagyjuk a `\def\magyarOptions{···}`-ből. Az opció jelentésének ismertetése után az alapérték és az egyéb lehetséges értékek hatásainak részletes ismertetése következik. Az értékeket az opció nevétől (a kulcstól) = jellel kell elválasztani, ezért az értékek a magyarázatban e jellel kezdődnek. Az opciók többségénél csak néhány előre megadott értékből lehet választani; az ettől való eltérést mindig jelezzük.

Vannak opciók, melyek egy-egy parancs definiálását szabályozzák. Ha értékük `=yes`, akkor az adott parancsot a magyar.ldf betöltésekor (újra)definiálja, `=no` érték esetén nem, `=weak` érték esetén pedig csak akkor definiálja a parancsot, ha az a magyar.ldf betöltésekor még nem létezett.

`accents=defaulthigh` ♦ Alapértékének nincs hatása, `=low` értéke a babel.def-ben definiált `\umlautlow` parancsot használva az ö és más betűk dupla pontos ékezetét viszi lejjebb, a betűhöz közelebb. Az `\umlautlow` parancs a `\lower@umlaut` makrót hívva ékezet tetejét a betű tetejétől 0.45 ex magasságba helyezi, ahol 1 ex az *x* betű magassága az adott betűtípusban. (Ezt úgy éri el, hogy a `\fontdimen5\font`-ot ideiglenesen átírja `\ht0` − 0,45 ex-re, ahol `\ht0` az ékezet magassága, ahogy eredetileg az *x* betűre kerülne.) Az opció `=high` értéke némileg feljebb viszi az ékezetet. A három pozíció, `=low`, `=defaulthigh`, `=high` sorrendben: ö, ö, ö. Bár a magyar tipográfia a `=low`-t javasolja, ennek csak OT1 fontkódolás mellett van hatása, továbbá a magyar szavak automatikus elválasztása T1 fontkódolást követel meg, és mivel az elválasztás a fontosabb, ezért inkább tartózkodjunk ennek az opciónak a használatától. Bizonyos betűtípusok, például e dokumentum betűtípusa is eleve alacsony ékezetes betűket tartalmaznak.

A TeX bármilyen betűre bármilyen ékezetet fel tud rakni, például P̋ és ą (több ékezet egymásra rakása nem mindig lehetséges). Ehhez az `\accent` primitívet Az OT1 fontkódolás csak a 2×26 angol betűt és az ékezeteket tartalmazza. A T1 fontkódolásban ezzel szemben szerepelnek az ékezetes betűk, mert az az automatikus elválasztási algo-

ritmus (technikai okokból) nem boldogul épített ékezetes betűkkel. (A T1 fontkódolás többek között ezzel a céllal jött létre.) Sajnos emiatt az ékezet magasságát T1 fontkódolás esetén csak új fontok készítésével lehet befolyásolni. Szerencsére a TeX támogatja a virtuális fontokat, tehát az új, alacsony ékezetes betűtípust nem kell újra megrajzolni, hanem készíthető a már meglévők transzformálásával. (Ez azért is előnyös, mert a keletkező program a jó minőségű, *hint*-elt Type 1 betűtípusokra építhetne, mint például a *BlueSky* fontjai és a CM-Super fontok.) Mivel a TeX-es betűtípusok különböző méretekben vannak megtervezve, ezért számuk igen magas: a virtuális fontokat ezért nem kézzel, hanem egy program írásával kéne létrehozni. Jelenleg ilyen program még nem létezik. A plain TeX-hez a TypoTeX kiadóban az 1990-es évek elején készült OM fontcsalád alacsony ékezeteket tartalmaz – ezen fontok LaTeX-es használata nem javasolt, mert számos jel és külföldi ékezetes betű hiányzik belőle.

`active=safe` ♦ Tegyen-e a magyar.ldf aktívvá egy karaktert? Az aktív karakterekről a 3.2.4. alalszakaszban olvashatunk részletesen. `=none` értéke nem tesz aktívvá karaktert. `=onlycs` sem aktivál karaktert, de bevezeti a `shorthandcs=` opcióban megadott nevű parancsot, amely ugyanazt nyújtja (hosszabb gépelés árán), mint egy aktívvá tett karakter. `=safe` aktívvá teszi a `activeprefix=` opcióban megadott karaktert, és a `shorthandcs=` opcióban megadott parancsot is. `=problematic` hatása megegyezik `=safe`-vel, de ezt máshogy, egyéb csomagokkal valószínűleg összeakadva éri el (a magyar.ldf 1.4-es verziója működött így).

A `=safe` kezdetben (a `\begin{document}` előtt) és nem magyar nyelvű szövegrész esetén, kompatibilitási okokból a karakter aktívságát megszünteti. Ez persze hiábavaló, ha betöltjük például a frenchb.ldf-et, amely a fordított aposztrófot végig aktívnak hagyja. Probléma esetén érdemes a babel-t minél később betölteni.

`activeprefix=babelopt2` ♦ Kiválasztja, hogy melyik karakter legyen aktív. `=none` értéke egyetlen karaktert sem tesz aktívvá, `=grave` a fordított aposztrófot (`‘`), `=acute` a sima aposztrófot (`’`), `=quotedbl` az írógép idézőjelet (`"`) aktiválja. `=babelopt2` az `’`-et teszi aktívvá, ha a babel `\usepackage[activeacute,···]{babel}` paranccsal lett betöltve, egyébként pedig `’`-et. `=babelopt3` az `’`-et aktiválja, ha a babel `\usepackage[activeacute,···]{babel}` paranccsal lett betöltve, a `‘`-et aktiválja `\usepackage[activegrave,···]{babel}` esetén, egyébként pedig az `"`-et. Az alapértelmezés megegyezik az 1.4-es magyar.ldf viselkedésével, amely mindig a `‘`-ot aktiválta. Az aktív karakterek használatát részletesen tárgyalja a 3.2.4. alalszakasz.

`activespace=none` ♦ Legyen-e egy rövid szóköz kihagyva minden kettőspont, pontosvessző, kérdőjel és felkiáltójel előtt? A magyar tipográfia megköveteli ezt a kihagyást, melyet a `=safe` érték meg is valósít a fenti négy karakter aktiválásával. Az alapértelmezett `=none` érték nem hagy ki helyet, mivel az aktiválás előre nem látható kompatibilitási problémákhoz vezethet. Elérhető még a `=problematic` érték, amely a `=safe`-et valósítja meg olyan összeférhetetlenségi problémákkal, ahogy a a magyar.ldf 1.4-es verziója tette (volna).

`afterindent=unchanged` ♦ Kell-e szakaszcím után az első sort beljebb kezdeni? Az alapérték ezt a dokumentumosztályra bízza, a `=force-yes` érték mindenképpen beljebb kezd (`\indent`), a `=force-no` pedig a margón kezd (`noindent`). A magyar tipográfia mindkét fajta kezdést megengedi.

`amslevelfix=fix-ams` ♦ Megjavítsa-e a magyar.ldf az ábrák jegyzékét (`\listoffigures` és egyéb jegyzékeket *AMS* dokumentumosztályok (pl. `\documentclass{amsart}`) esetén, hogy ne legyen *Overfull* `\hbox`... hibaüzenet. `=fix-ams` csak akkor javítja meg, ha egy *AMS* dokumentumosztály van betöltve, `=fix-all` mindenképpen megkísérli a javítást, `=unchanged` nem javít.

Az opció nem kötődik szorosan a magyar nyelvhez.

amspostsectiondot=no ♦ Bizonyos *AMS* dokumentumosztályok (amsproc, amsbook és amsart) ponttal fejezik be a strukturális egységek címsorait, amit a magyar tipográfia megtilt. Ha az opció értéke nem unchanged=, akkor eltünteti ezt a pontot.

amstocnumskip=\enskip ♦ Előírja (a \tocsection és hasonló parancsok átdefiniálásával), hogy *AMS* dokumentumosztályok magyar nyelvű bejegyzéseinek számait a tartalomjegyzékben mekkora kihagyás válassza el a fejezet nevétől. Tetszőleges LaTeX kód megadható. A magyar tipográfia az \enskip-et javasolja az *AMS* által használt \quad helyett. Ha üres értéket adunk meg, akkor nem nyúl a \tocsection-höz.

amstocnumlang=all ♦ Alapértéke beállítja, hogy az amstocnumskip= opció a nem magyar nyelvű fejezetek tartalomjegyzék-beli címsoraira is vonatkozzon. A =hu érték csak a magyar nyelvűekre alkalmazza az amstocnumskip= opciót.

az=weak ♦ A határozott névelőt (*a/az*) automatikusan kirakni képes \az parancsot és az őt használó \aref és egyéb parancsokat definiálja, a részleteket lásd a 3.2.9. alalszakaszban. Egyéb értékei: =no és =yes.

captions=hu ♦ Ha értéke nem =unchanged, akkor a magyar.ldf a járulékos fejezetneveket és egyéb feliratokat magyarul jeleníti meg. Például a \bibname makrót *Bibliography*-ról átírja „Irodalomjegyzék"-re, és a képaláírásokban sem *figure* jelenik majd meg, hanem „ábra".

chapternumber=unchanged ♦ Ha értéke =huordinal, akkor a fejezetszámok magyarul betűzve jelennek meg, például „harmadik fejezet". Ha az érték =Huordinal, akkor nagybetűvel kezdődik, például „Negyedik fejezet". Az alapérték nem változtat a számozáson, ami a LaTeX osztályok többségében arab számokkal történik. Lásd még a hunumbers= opciónál.

classmod=yes ♦ A magyar.ldf ismer néhány LaTeX osztályt (ezek: book, report, article, letter), melyek viselkedését képes módosítani. Az ismert osztályokban a \part és \chapter parancsok, a tartalomjegyzékek és a fejlécek tipográfiáját módosítja, ha classmod=yes-t adunk meg. classmod=unchanged esetén módosítás nem történik. Bár ismeri letter-t is, a levél fejlécét és egyéb formai elemeit nem a magyar szabályok szerint helyezi el, csak a címsort módosítja. A fejléceket – technikai okokból – nem módosítja toclang=unchanged opció esetén. Ajánlott még a partnumber=Huordinal használata, mert ekkor 1.\ rész helyett Első rész fog megjelenni a \part parancs hatására. Az élőfejet a \ps@headings parancs definiálja, ezért a magyar.ldf eintézi, hogy az ő definíciója lefusson a preambulum végén. Ha a tipográfus a classmod= opció mellett saját élőfejet szeretne, akkor például book.cls esetén a \ps@headings helyett a \@@magyar@book@psheadings parancsot kell a preambulumban felüldefiniálnia.

defaults=over-1.4 ♦ Az alapbeállításokat jelöli ki. Célszerű a többi opció előtt megadni. További értékei: =compat-1.4, =safest, =prettiest. A részleteket lásd fent.

dottedtocline=fix ♦ Alapértéke a \@dottedtocline parancs átdefiniálásával engedélyezi, hogy a tartalomjegyzékek pontozott sorainak végén az oldalszám balra túllógjon a számára fenntartott \@pnumwidth távolságon. Az =unchanged érték nem változtat a \@dottedtocline alap LaTeX-parancson.

Az opció nem kötődik szorosan a magyar nyelvhez.

emitdate=weak ♦ A dátumok magyar helyesírás szerinti megjelenítését végző, több formából választani engedő és a ragozást is támogató \emitdate parancsot definiálja. Egyéb értékei: =no és =yes. Lásd még a hutoday= opciót.

Az \emitdate[*toldalék*]{*formátum*}{*dátum*} hívás felismeri a *dátum*-ot, és megjeleníti az előírt *formátum*-ban, esetleg *toldalék*-kal látva el. *toldalék* a \told parancs

+-a után megengedett tetszőleges toldalék lehet, tipikus értékei: e és a+an. Az edik+e (1848. március 15-ödike) helyett e javasolt.

A \emitdate az ilyen formák egyikében megadott *dátum*-okat érti meg: \today, É-H-N (ISO dátumformátum), É/H/N (LaTeX csomagok keltezésének formátuma), É/H/N:X:Y (LaTeX alap \today parancs formátuma), É.N.H (keverék formátum), É. hónap N (magyar \today), Hónap N, É (amerikai \today, LaTeX osztályok \today formátuma). A dátum formátum szerinti beolvasásához a TeX-be épített makrókifejtés egyszerű mintaillesztését használja. Ez a módszer hibakezelésre nem alkalmas, ezért a fentiektől eltérő szintaxis esetén a működés nincs definiálva (lehet, hogy egy értelmetlen hibaüzenet kapunk, de lehet, hogy a LaTeX fordítás innen teljesen elromlik). A $0 \ldots 49$ közötti évszámokhoz 2000-et, az $50 \ldots 99$ közöttiekhez pedig 1900-at hozzá. A dátum végén lehet egy extra pont, amit figyelmen kívül hagy. A szóközök számítanak.

A \emitdate *formátum* argumentuma a kimeneti formátumot adja meg. A magyar helyesírásban által megengedett összes formátumot (a–h) tudja generálni: 1848-03-15 (a: ISO dátumformátum); 1848. március 15. (b: teljes kiírt hónap); 1848. márc. 15. (c: rövidítve kiírt hónap); 1848. III. 15. (d: római számos hónap); 1848. 03. 15. (e: arab számos hónap); 1848. március közepe (f: teljes kiírt hónap, nap nélkül); 1848. március 15-én; (g: teljes kiírt hónap, a nap toldalékolva); 1848 március-ában (h: év és hónap birtokos szerkezetben, a magyar.ldf még nem oldja meg a hónapnév automatikus ragozását, ezért itt a kötőjel helytelen).

extras=yes ♦ A magyar nyelv aktívvá válásakor a babel számos beállítást módosít az \extrasmagyar makró futtatásával (és amikor egy másik nyelv lesz aktív, akkor előbb a \noextrasmagyar makró fut le). A magyar.ldf betöltődésekor a fenti két makróba építi bele saját nyelvspecifikus beállításait. Az opció =yes alapértéke engedi érvényre jutni ezeket a beállításokat, a =no érték pedig letiltja őket.

Az extras=no használata egyedül hibakeresésre ajánlott, mert épp a magyar.ldf fő célját hiúsítja meg. Hibakeresés esetén érdemes a extras=no-t is tartalmazó defaults=safest alapbeállításokból kiindulni.

figurecaptions=hu ♦ Az ábrák aláírásának fejlécét állítja be. Az alapérték „5.6. ábra” formátumú fejlécet eredményez, az =us érték „ábra 5.6”-ot (angolul ez „Figure 5.6” lenne, nagybetűvel), az =unchanged nem változtat a beállításokon.

footnote=yes ♦ A lábjegyzetek megjelenítési stílusát szabályozza. =yes értéke elérhetővé teszi a \footnotestyle parancsot, mellyel a 3.2.7. alalszakaszban leírt módon testreszabható a lábjegyzetek megjelenítése. =huplain értéke ezen felül még a magyar tipográfiának megfelelő, arab számokat használó stílust állít be, =starplain értéke pedig csillagokat használó stílust állít be, ami szintén megfelel a magyar tipográfiának. Az =unchanged érték nem változtat a lábjegyzeteken, és a \footnotestyle parancsot sem definiálja.

frenchspacing=unchanged ♦ =yes értéke az írásjelek utáni szóköz szélességét a magyar tipográfiának megfelelően egyenletesre állítja (\frenchspacing), amíg a magyar nyelv aktív. =no értéke a magyar nyelv aktívvá válásakol az írásjeltől függő változó szóközszélességet ír elő (a . *?* *!* : ; , írásjelek után lesz a normálnál szélesebb szóköz, az írásjeltől függő mértékben). Az =unchanged érték nem változtat a beállításokon.

hang=weak ♦ A szimbólummal kezdődő, függő bekezdések nyitására szolgáló \hang parancsot definiálja. Egyéb értékei: =no és =yes. A \hang paranccsal az itt láthatóhoz hasonló felsorolások hozhatók létre, például \hang{$\circ$ } után a sor elején „○ ” jelenik meg, és a bekezdés további sorai ennek jobb oldalán kezdődnek.

hunnewlabel=yes ♦ A generált oldalszámok és fejezetszámok névelővel ellátását segítő \hunnewlabel segédmakrót definiálja, a részleteket lásd a 3.2.9. alalszakaszban. =no

értéke nem generál \hunnewlabel-eket, =only-hu értéke pedig csak akkor generál, ha a \label parancs hívásakor a magyar nyelv aktív.

hunumbers=yes ♦ Az alapérték a nem túl nagy egész számokat magyarul kibetűző \@hunumeral és egyéb parancsokat definiálja. (A =no érték pedig kihagyja a definíciót.) A parancsok $-9999 \ldots 9999$ közötti egész számokat betűznek ki tőszámnévként (pl. ezerkilencszázkilencvenkilenc) és sorszámnévként (pl. kétezer-negyedik), mindezt kifejthető módon, tehát a parancsok használhatók fejezetek (lásd a chapternumber= opciót) és részek (partnumber=) számozására. A TeX kifejthető konstrukcióiban lehetséges egész számok összehasonlítása, továbbá string összefűzés és darabolás, de az aritmetikai műveletek hiányoznak, ezért a \@hunumeral nagy vonalakban így működik: először összehasonlításokkal megállapítja, hogy hány jegyű a szám, ezután az első jegyet stringdarabolással kinyeri, és kiírja betűvel, majd a maradék, immár rövidebb szám kiírása következik.

Nincs elvi akadálya a 9999-nél nagyobb (de 2^{31}-nél kisebb) egész számok generálásának, de az ekkora számok betűvel való kiírása nagyon ritka, különösen fejezetszámokban, ezért nem lett implementálva.

hutoday=yes ♦ Ha értéke =yes, akkor a dátumok magyar megjelenítéséért felelős parancsokat definiálja, =no érték esetén pedig nem módosít az alapbeállításokon. A szóban forgó parancsok: \today, ami a \maketitle hatására megjelenik; \ondatemagyar és \ontoday, melyek az -án/-én raggal ellátotott \today-nek felelnek meg, de a babel sehol nem használja őket; és végül \SafeToday, amely a LaTeX kernelnek a babel által felüldefiniált \today parancsával egyenértékű. (Megjegyezzük, hogy a fenti parancsok a LaTeX indításának időpontját tükrözik, melyet a \year, \month, \day és \time változókból olvasnak ki, $\texttt{\time} = 60 \cdot \text{óra} + \text{perc}$. A magyar nyelvű dátummegjelenítésnek egy gazdagabb, ragozást is megengedő formája az \emitdate parancs, lásd az emitdate= opciónál.

hyphenation=huhyph3 ♦ Előírja, hogy a TeX milyen szabálykészlet szerint válasza el a magyar szavakat. =huhyph3 értéke az 1998 óta klasszikusnak számító 3-as verziót használja, amely kevés összetett szót tud helyesen elválasztani, =composite értéke a huhyphc.tex 2002-ben készült 4-es verzióját (vagy újabbat) használja, amely már az összetett szavak többségét helyesen választja el, az idegen szóösszetétel-határokat is követi (pl. szin-kron) [12, §231 és §238]) szerint készült. A =phonetic érték a [12, §224])-et követő, az idegen szóösszetételeket (pl. szink-ron) figyelmen kívül hagyó, a huhyphf.tex-ben található 4-es verziót (vagy újabbat) használja. A =huhyphn a 2003-ban készült, az idegen szóösszetételeket (pl. szink-ron) figyelmen kívül hagyó, a huhyphn.tex-ben található mintákat használja. A =none érték letiltja az automatikus szóelválasztást. A hyphenation= opció csak a magyar szavak elválasztására van hatással. Az elválasztásról részletesen ír a 3.3. alszakasz.

hyphenmins=22 ♦ Ha értéke két számjegy, akkor a bal illetve jobb oldali számjegy azt adja meg, hogy a magyar nyelvű szavak bal illetve jobb oldalán legalább hány karakter közé nem szabad automatikusan elválasztójelet tenni. A babel és a LaTeX alapbeállítása 23, a magyar.ldf-é pedig 22, tehát például „Aliz” a bal oldala miatt nem választható el, „Sára” viszont igen, mivel mindkét oldalon megvan a két karakter. (A „fáraó” szó a jobb oldala miatt csak egy helyen választható el, az „évei” szó egyik elválasztásai pontja a bal, a másik pedig a jobb széléhez van túl közel.) Az elválasztásról további részletek olvashatók a 3.3. alszakaszban.

A hyphenmins= a babel-en keresztül nyelvváltáskor a beépített TeX \lefthyphenmin és \righthyphenmin számlálókat módosítják. Ezekhez nem érdemes kézzel hozzányúlni, mert nyelvváltáskor visszaíródik a nyelvre jellemző érték.

`labelenums=unchanged` ♦ A `=hu-a` érték az `enumerate` környezet egyes szintjeinek számozási stílusát állítja be a magyar tipográfiának megfelelően (például a külső `enumerate` címkéje nagy római szám); a `=hu-A` érték hasonló, de egy másik magyar szokáson alapul, melyben a nagy latin betűs címkék is helyet kapnak; az alapérték pedig nem változtat a címkéken. A `=hu-d` érték ugyanaz, mint `=hu-a`, de a legfelső szinten levő nagy római számot a legalsó szintre száműzi.

A magyar.ldf öt számozott szintet definiál, de az alap LaTeX-ben a számozott és számozatlan listák maximum 4-4 szintűek lehetnek, tehát összesen 8 mélységbe lehet eljutni. Ezen az `enumerate` környezet definíciójában levő `\ifnum\@enumdeph>3` teszt enyhítésével segít ez az opció: a `\expandafter\ifx\csname labelenum\romannumeral\the\@enumdepth\endcsname\relax` tesztje enged például 5 mélységbe lemenni, ha a `\labelenumv` parancs definiálva van.

`labelitems=unchanged` ♦ A `=hu` érték az `itemize` környezet egyes szintjeinek címkestílusát állítja be a magyar tipográfiának megfelelően (például a külső `itemize` címkéje „–” lesz), az alapérték pedig nem változtat a címkéken. Lásd még a `labelenums=` opciót.

`longcaption=justified` ♦ Azt szabályozza, hogy az ábrák és táblázatok egy sornál hosszabb aláírásai hogyan jelenjenek meg. Az alapérték sorkizárt bekezdésként jeleníti meg az aláírást, a `=centered` középre igazítja (mint az egysorosakat), a `=centernewline` középre igazít, de a „42. ábra.” szöveg után új sort kezd, az `=unchanged` érték meghagyja az alapbeállításokat.

`mathbrk=define` ♦ A `=define` alapérték a matematikai szimbólumok sortörését támogató `\MathBrk` parancsot definiálja, a `=fix` érték ezen felül még a magyar tipográfiai szokásokhoz igazítja néhány matematikai bináris operátor és bináris reláció sortöréssel kapcsolatos viselkedését. Az `=unchanged` érték nem definiál és nem módosít semmit. A `=fix` érték együttműködik a nath és az amsmath csomagokkal, de kockázatos használni, mert esetleg más csomagokkal összeakadhat.

A magyar tipográfia a főszövegen belüli belső képletekben a bináris relációk és a bináris operátorok körül (a $\cdot$ szorzásjelet és / perjelet kivéve) megengedi a sortörést, de sortörés esetén az operátort illetve relációt a következő sor elején – mint az $1 + + 2 + 3 < 10$ formulában is – meg kell ismételni. Erre szolgál a `\MathBrk` parancs, amely egyetlen szimbólumot vár paraméterül, ami vagy egyetlen karakter (pl. +), vagy egy parancs (pl. `\cup`). A `\MathBrk` csak akkor elérhető, ha a magyar.ldf-et megfelelő `mathbrk=` opcióval töltöttük be. A `\MathBrk` parancs csak akkor módosítja a szimbólumot, ha a magyar nyelv aktív, a `\MathBrkAll` parancs viszont mindig. A kézi ismétlés is lehetséges, például az = jel helyett `=\nobreak\discretionary{}{\hbox{\(=\)}}{}`-et kéne gépelni. (Itt a $ helyett azért szerepel `\(` és `\)`, hogy a nath csomag által átállított kategóriakódokkal is működjön, a `\nobreak` pedig megakadályozza az = jel utáni ismétlés nélküli sortörést.)

A szorzásjel körüli elválasztás megakadályozható `\nobreak\cdot\nobreak` gépelésével `\cdot` helyett, vagy a `\cdot` parancs átdefiniálásával: `\DeclareRobustCommand\cdot{\nobreak\mathchar"2201\nobreak}`. A perjel körül alapból nem választ el a LaTeX, de ha `\slash`-ként visszük be a perjelet, akkor helyette `\nobreak\slash\nobreak`-et kell írni, de egyszerűbb simán `/`-t gépelni.

Ha a magyar.ldft a `mathbrk=fix` opcióval van betöltve, akkor a szorzásjel és a perjel után sosem töri el a sort, és az összes [17]-ben felsorolt bináris relációt ($<$, $>$, $=$, : és 43 db olyan, ami paranccsal érhető el) és bináris operátort ($+$, $-$, $*$ és 37 db olyan, ami paranccsal érhető el) megdupláz. Ez az opció a minden matematikai képlet elején újradefiniálja az egykarakteres operátorok jelentését, ami kicsit lassítja a

LaTeX-et, és kompatibilitási problémákat okozhat. A nath és amsmath csomagokkal az együttműködés tökéletes.

`mathhucomma=define` ♦ Alapértéke definiálja a `\HuComma` parancsot, amely matematikai módban egy tizedesvesszőt (*ordinary* kategória) vagy egy vessző formájú írásjelet (*punctuation*) helyez el, attól függően, hogy a parancsot számjegy követi-e. A kétféle vessző között az a különbség, hogy a tizedesvessző után nincs vízszintes kihagyás. Ha az opció értéke `=unchanged`, nem definiálja a `\HuComma` parancsot, a `=fix` érték pedig a definíción felül a , matematikai jelentését módosítja `\HuComma`-ra, így a tizedestörtek az `3{,}14` helyett egyszerűen `3,14`-nek gépelhetők (a tizedesvessző legpontosabb alakja: `\mathchar"013B`, az írásjelé pedig `\mathchar"613B`, jól látható, hogy csak az első, a kategóriát és ezzel a térközt meghatározó számjegyben különböznek). Matematikai módban a vessző az aktív nyelvtől függetlenül vesszőt szúr be (az őt követő karaktertől függően írásjelet vagy tizedesvesszőt, de sosem pontot). Lásd még a `mathreal=` opciót. Megjegyezzük, hogy a TeX programozásakor megadandó távolságmennyiségekben a tizedespont és a tizedesvessző egyaránt elfogadott.

Ha az opció aktív, akkor a „$F_i(x,y) = y^i + 1{,}3x,\ x,y \in A,\ i = 1,\ 2{,}3,\ \ldots$” formula egyszerűen, fölösleges kapcsos zárójelek nélkül begépelhető: `$F_{i}(x,y)=y^i+1,3x,\ x,y \in A,\ i=1,\ 2,\ 3,\ldots$`. Ahhoz, hogy a `\␣` parancs a képlet közepén el tudja törni a sort, a nath csomag betöltése szükséges (egyébként pedig `\allowbreak` hívásokat kell elhelyezni a képletben).

`mathfactorial=define` ♦ A `=define` alapérték a matematikai faktoriálisjelet helyes térközzel beszúró `\factorial` parancsot definiálja, a `=fix` érték ezen felül még átírja a felkiáltójel képleten belüli jelentését, `\factorial`-ra. Az `=unchanged` érték nem definiál és nem módosít semmit. A nath csomag eme opciótól függetlenül helyes térközzel helyezi el a faktoriálisjelet. A `=fix` érték együttműködik a nath és amsmath csomagokkal, de kockázatos használni, mert esetleg más csomagokkal összeakadhat. `=yes` érték esetén a faktoriális igen egyszerűen bevihető, csak a perjel előtt van szükség kapcsos zárójelekre, például `$x=a!+n!k!+(c+d){!}/e!$` → $x = a! + n!\,k! + (c + d)!/e!$. (Az előző példában figyeljük meg a szorzótényezők közötti szóközt, ami `mathfactorial=yes` nélkül hiányozna.) Ha csak az alapértelmezés szerinti `mathfactorial=define` opció aktív, akkor `!` helyett `\factorial`-t érdemes gépelni, hogy a térközök helyesek legyenek. A nath (CTAN-ról letölthető) csomag a `mathfactorial=` opció használata nélkül is a helyes térközzel szúrja be a faktoriális.

Ha az `activespace=` és a `mathfactorial=` opciókat együtt használjuk, akkor a felkiáltójelre matematikai képletben a `mathfactorial=`, szövegben pedig az `activespace=` van hatással.

Az opció nem kötődik szorosan a magyar nyelvhez.

`mathmuskips=unchanged` ♦ `=hu` értéke a matematikai képletekben levő térközökben a magyar tipográfia szűkebb matematikai térközeit alkalmazza (`\thickmuskip=4mu␣ plus2mu minus4mu`, `\medmuskip=2mu plus1.5mu minus2mu`, `\thinmuskip=3mu`; lásd még a 23. oldalon levő kiemelt egyenletben), `=latex` érték a LaTeX alapbeállításait erősíti meg (`\thickmuskip=5mu plus5mu`, `\medmuskip=4mu plus2mu minus4mu`, `\thinmuskip=3mu`), míg az `=unchanged` érték nem változtat a térközök beállításán. A magyar térközök anomáliája, hogy a keskeny térköz szélesebb a közepesnél, lásd még [3, 26. oldal]-on.

`mathreal=weak` ♦ A képletekben a tizedespontot tizedesvesszővel helyettesítő `\MathReal` parancsot definiálja. Egyéb értékei: `=no` és `=yes`. Lásd még a `mathhucomma=` opciót.

A `\MathReal` parancs a paraméterül kapott matematikai képletet úgy jeleníti meg,

hogy minden pontot tizedesvesszőre cserél, de csak akkor, ha a magyar nyelv aktív. A paraméterben a pontokat ki kell írni, tilos makróba rejteni. A `\MathReal` képleten belül és azon kívül is működik. Például: `\MathReal{1.25+2+3.75=7}` → $1{,}25 + 2 + 3{,}75 = 7$; valós számok: 45, $-56{,}07$; real numbers: 45, -56.07; itt a számok egységesen `$\MathReal{45}$`-ként és `\MathReal{-56.07}`-ként lettek begépelve.

`mond=weak` ♦ A párbeszédekben új szereplő megszólalását bevezető `\mond` parancsot definiálja (`=no` és `=yes`). A `\mond` egyszerű definíciója lehetne `\par--\hbox{ }`, melyben a `\hbox` megakadályozza a gondolatjel utáni szóköz nyúlását. A magyar.ldf-ben egy hosszabb definíció szerepel, amely beállítja, hogy a megszólalás pl. `\paragraph` után is a sor elején kezdődjön. Néhány esetben csúnya lehet a tompán, behúzás nélkül szedett megszólalás strukturális egységek elején, a behúzás biztosítható a `\mond` elé írt `\csname @afterindenttrue\endcsname` nem túl szép hívással.

`openqq=maybedown` ♦ Azt adja meg, hogy a ‘‘ és ’’ jelek hogyan viselkedjenek. (Ha az `activeprefix=` más karaktert jelölt ki aktívnak, akkor a fenti két parancs első karaktere értelemszerűen megváltozik). Az `=unchanged` érték nem kezeli parancsként a fenti karaktersorozatokat, tehát ‘‘ valószínűleg egy angol dupla nyitó felső idézőjelet fog eredményezni (“) – feltéve, hogy beírásakor az aktuális fontban van ilyen ligatúra; a ’’ pedig valószínűleg két angol szimpla idézőjelet (”) jelenít meg. Az `=up` érték ‘‘-re angol dupla nyitó felső idézőjelet (“), ’’-re pedig ennek alsó változatát („) adja, a `=down` érték ennek épp fordítottja: ‘‘-re alsó („) és ’’-re felső (“). Az opció hatása természetesen csak akkor érvényesül, ha a magyar nyelv aktív.

A `=maybedown` alapérték `active=none` esetén `=unchanged`-del, egyébként `=down`nal ekvivalens. Ennek első látásra talán meghökkentő mellékhatása, hogy a forrásban ‘‘-nek gépelt jel alsó idézőjelként jelenik meg. Ez az 1.4-es magyar.ldf-fel való kompatibilitás miatt van így, továbbá azért, mert pl. a CM fontokban nincs ligatúra a magyar nyitó idézőjelnek („), de így ligatúra nélkül is begépelhetők ‘‘-ként.

Új dokumentum írásakor az javasolható, hogy az `openqq=` alapbeállításon ne változtassunk, és az idézőjeleket a `\textqq` paranccsal helyezzük el (lásd a `textqq=` opciónál).

`partnumber=unchanged` ♦ Hasonló a `chapternumber=`-höz, de a részek (`\part`) és kötetek számozására vonatkozik, például „Ötödik rész”.

`postdescription=unchanged` ♦ Mint a `postpara` opció, de a `description` környezetre van hatással.

`postpara=unchanged` ♦ A `\paragraph` címsora és a szöveg közti kihagyást illetve a díszpontot állítja be. `=unchanged` értéke nem változtat az osztály beállításain, a `=quad`, `=enskip`, `=em` és `=empty` különböző mértékű kihagyást írnak elő, díszpont nélkül; a `=bfdot`, `=dot`, `=diamond` értékek pedig különféle díszpontokat eredményeznek kihagyással. Az opciók pontos jelentését lásd a magyar.ldf forrásában definiált `\magyar@post@...` parancsoknál.

`postsubpara=unchanged` ♦ Abban különbözik a `postpara` opciótól, hogy a `\subparagraph` parancsra van hatással.

`refstruc=weak` ♦ A dokumentum strukturális egységeire hivatkozó `\refstruc` parancsot definiálja, a részleteket lásd a 3.2.11. alalszakaszban. Egyéb értékei: `=no` és `=yes`.

`safest=no` ♦ Történeti okokból maradt meg, használata kerülendő. `=no` értékének nincs hatása, `=yes` értéke pedig `defaults=safest`-tel egyezik meg.

`sectiondot=safe` ♦ Bizonyos *AMS* dokumentumosztályok (amsproc, amsbook és amsart) ponttal fejezik be a strukturális egységek címsorait, amit a magyar tipográfia megtilt. Ha az opció értéke nem `=none`, akkor eltünteti ezt a pontot is. A `=problematic` hatása megegyezik `=safe`-évl, de ezt máshogy, egyéb csomagokkal valószínűleg

összeakadva éri el (a magyar.ldf 1.4-es verziója működött így).

`shorthandcs=\shu` ♦ Annak a parancsnak a nevét írja elő, amely az aktív karakterrel (lásd a 3.2.4. alalszakaszban) egyenértékű. Ha értéke `=none`, akkor nem lesz ilyen parancs. Ettől eltérő csak akkor adható meg, ha nem `active=none` van beállítva. Ha egy parancs van megadva (ajánlott), pl. `shorthandcs=\shu`, akkor az eldobja az első paraméterét, így például a `\shu&|` és `‘|` ugyanazt fogja eredményezni (feltéve, hogy `‘` az aktív karakter). Ha viszont a `shorthandcs=` értéke egy backslash nélküli név, akkor nincs ilyen eldobás: tehát például `shorthandcs=susu` esetén a `\susu|` és `‘|` jelenti majd ugyanazt.

A kétfajta parancs közül az első paraméterét eldobó ajánlott, mert ezáltal se a `po‘ttyan`, se a `po\shu‘ttyan` szóba nem kell szóköz, míg a `po\susu␣ttyan`-ba kéne.

`shortrefcmds=yes` ♦ `=yes` értéke a magyar.ldf 1.4-es verziójával való kompatibilitás miatt alternatív, háromkarakteres neveket definiál a névelővel hivatkozó parancsoknak: `\aref` → `\azr`, `\Aref` → `\Azr`, `\apageref` → `\azp`, `\Apageref` → `\Azp`, `\acite` → `\azc`, `\Acite` → `\Azc`. A `=no` érték e hárombetűs parancsokat nem definiálja.

`suggestions=yes` ♦ Alapértéke angol nyelvű figyelmeztető üzeneteket jelenít a preambulum befejeztével, ha a magyar.ldf-et hiányosan vagy hibásan próbálják használni. A figyelmeztetést kikapcsolja a `=no` érték. Figyelmeztetés jelenik meg akkor, ha a szerző elfelejti betölteni a `t1enc` csomagot (mert ez szükséges az ékezetes betűket tartalmazó szavak elválasztásához), ha nem tölti be az inputenc csomagot, vagy szokatlan bemeneti kódolást választ (tipikusan rossz kódolás a `latin1`, mert ekkor hullámos *õ* és kalapos *û* kerül a kimenetre; ajánlott kódolás a `latin2`), ha nem találhatók a felhasználó által kért elválasztási minták, ha a `\def\magyarOptions` vagy a `\PassOptionsToPackage` túl későn, a magyar.ldf betöltése után lett kiadva (és emiatt nem fejtette ki hatását), vagy ha a varioref csomag hibás verziója a `magyar` opcióval lett betöltve (lásd a `varioref=` opciónál).

`tablecaptions=hu` ♦ A táblázatok aláírásának fejlécét állítja be. Az alapérték „5.6. táblázat” formátumú fejlécet eredményez, az `=us` érték „táblázat 5.6”-ot, az `=unchanged` nem változtat a beállításokon.

`textqq=weak` ♦ Az idézőjeleket (macskakörmöket és egyéb jeleket) generáló `\textqq` parancsot definiálja, amely magyar hagyomány szerinti egymásba ágyazott idézéseket generálja, ha a magyar nyelv aktív (pl. `\textqq{a\textqq{b\textqq{c}d}e}` → „a»b’c’d«e”), egyébként pedig az amerikai angolt: “a‘b’e”. Az opció egyéb értékei: `=no` és `=yes`.

`theoremtitle=hu` ♦ Tételszerű környezetek címsorainak a formátumát határozza meg. Az `=unchanged` érték nem változtat az alapbeállításokon, tehát a dokumentumba valószínűleg az angol tipográfia szerinti *Theorem 1.2* kerül. A `=hu` érték megcseréli a sorrendet és elhelyezi a pontokat: a tételszám után pont és szóköz következik, majd a környezet neve (pl. tétel, lemma, bizonyítás), megint egy pont, és végül szóköz jön, például: „1.2. tétel.” vagy „3. lemma.”.

A magyar.ldf, ha `theoremtitle=hu` opcióval töltjük be, a magyar tipográfiai szabályoknak megfelelően szedi a tételszerű környezetek címeit. A magyar.ldf a theorem, ntheorem és amsthm csomagok jelenlétében is helyesen működik. E csomagok detektálását a preambulum végén végzi, hogy ne függjön a végeredmény a csomagok betöltési sorrendjétől. A theorem és ntheorem csomagok lehetőséget biztosítanak különböző tételstílusok definiálására. A magyar.ldf `magyar-plain` néven hoz létre stílust, melyet alapértelmezésben aktívvá is tesz. Nyelvváltáskor a tételstílusok nem változnak, az eredetihez `\theoremstyle{plain}` paranccsal lehet visszatérni. Az amsthm csomag használatakor a `\swapnumbers` parancstól függetlenül jó sorrendű tételfejlécet kapunk.

Az amsthm csomag lehetőséget biztosít számozatlan tételszerű környezet definiálására (a \newtheorem* paranccsal), ez is magyar sorrendben fog megjelenni.

A magyar.ldf korábbi verziója ugyan magyar tételszámokat állított be, de ezek összeakadtak a fenti csomagokkal.

titles=\enskip ♦ A strukturális egységek címeinek (pl. \section) szedésekor a pont és a cím közötti kihagyást előidéző parancsot adja meg. Ha =unchanged az értéke, akkor nem változtat az osztály beállításain, és a kihagyás valószínűleg túl nagy (\quad) lesz. *AMS* dokumentumosztályok esetén a címek megjelenését a tartalomjegyzékben az amstocnumskip= opció szabályozza, *AMS* nélkül pedig az \l@section és hasonló makrók (lásd még a dottedtocline= opciót). A korábbi verzió a címek szedését a \@ssect és \@sect makrók felüldefiniálásával végezte, de ez összeakadt a az *AMS* osztályokkal, ezért ez 1.5-ös verzió a \@seccntformat parancsot írja felül a főszövegben, a \numberline parancsot pedig a tartalomjegyzékben. A \numberline a táblázatok és ábrák jegyzékére is hatással van, de ez nem baj, mert ott is szükség van a pontra. Ha a szám eleve pontra végződik, a \numberline a régi pontot előbb eltávolítja az alább definiált \stripdot{*tokenek*} segítségével.

```
\def\stripdot#1{\expandafter\stripdot@lowb\stripdot@lowa#1\ennyi.\ennyi}
\def\stripdot@lowa#1.\ennyi{#1\ennyi}
\def\stripdot@lowb#1\ennyi#2\ennyi{#1}
```

A \stripdot csak akkor működik, ha az argumentum nem tartalmazza az \ennyi tokent.

toclang=good ♦ Alapértéke módosítja a jegyzékek (pl. .toc, .lof és .lot kiterjesztésű fájlok) generálását, hogy minden bejegyzés elé egy \select@language parancs kerüljön, így a megjegyzés a saját nyelvével legyen szedve, így például működjön a tartalomjegyzékbe kerülő címekben is a magyar aktív karakter. Az aktív karakterekről további információt lásd a 3.2.4. alalszakaszban. Az =unchanged érték hagyja érvényesülni a babel hibás implementációját, a =badfloat és =badall értékek pedig a régebbi magyar.ldf-fel kompatíbilis, de hibás megvalósításokat léptetnek életbe. A babel maga is tartalmaz ilyen törekvéseket, de ezek időzítés szempontjából egyáltalán nincsenek átgondolva, így például megzavarja őket, ha egy, a főszövegtől különböző nyelvű úsztatott ábra kerül az oldal tetejére. A magyar.ldf-ben alkalmazott javítás remélhetőleg bekerül a babel következő hivatalos verziójába.

told=weak ♦ Az egész számok toldalékolását végző \told parancsot definiálja, a részleteket lásd a 3.2.10. alalszakaszban. Egyéb értékei: =no és =yes.

varioref=hu ♦ Az alapérték megjavítja magyar nyelv esetén a varioref csomag nyelvfüggő parancsait, például \reftextfaraway, hogy a \vref és \vpageref parancsok magyar hivatkozásokat generáljanak. Például \vref{pdi:tovabbfejlesztesi} eredménye „ 4.2 a 86. oldalon” lesz. A generált hivatkozások használják az \az és \told parancsokat, például „a 30–50. oldalakon” és a „3.4-től 5.6-ig” oldalalemek generálásakor. Az =unchanged érték nem változtat semmit. A varioref csomag 2001-ben készült 1.3c verziója hibás, ezért tilos a magyar opcióval betölteni, mert minden hivatkozásra TEX hibaüzenetet ad. A hiba oka az, hogy a sehol sem definiált \aza parancsot próbálja használni, és az ennek hiányakor lefuttatott hibakezelő rutin értelmetlen üzenetet jelenít meg. A magyar.ldf figyelmeztet a varioref csomag hibás betöltésére, emellett automatikusan javítja is a hibát.

3.2.4. Az aktív karakter

A babel terminológiájában az aktív karakter egy olyan parancs, amely nem backslash-sel kezdődik, ezáltal a szerző egy karaktert megspórol a gépelésnél. A magyar.ldf alapér-

telmezett aktív karaktere a ‘ (fordított aposztróf). Fő funkciója, hogy lehetővé teszi az utána következő dupla többjegyű mássalhangzó mentén a szóelválasztást: például a `fa¬‘ttyúsor-t` csak *fattyú-sor*-ként választja el a TEX, de `fa‘ttyúsor` esetén *faty-tyú-sor* lesz az elválasztás. Aktív karakter nélkül az alábbi hosszadalmas kódot kéne gépelni: `fat¬\nobreak\discretionary{y-}{}{}\nobreak\hskip\z@skip tyúsor`. A magyar.ldf régi verziója nem tartalmazta a `\hskip`-et, ezért a TEX nem találta meg a *tyú-sor* elválasztást. Az aktív karakter az összes érintett mássalhangzót támogatja ([18]): *locs-csan*, *pogy-gyász*, *har-kály-lyal*, *meny-nyei*, *poty-tyan*, *rizs-zsé*, *edz-dzünk*, *bridzs-dzsel*. A *dzs* betű érzékelése csak az 1.5-ös verziótól valósul meg. A *ddzs* és a *ddz* közötti különbségtétel miatt négy karaktert előre kell tekinteni az aktív karakter után, melyre – nem túl elegánsan – felhasználható a `\futurelet` primitív négyszeri hívása; ha csak a `dd` után derül ki, hogy nem `z` következik, akkor már nem lehet két karaktert visszalépni, hanem ki kell szedni az eddig beolvasott `dd`-t. Az aktív karakter az utána következő betűt – sajnos – makróargumentumban kapja meg, tehát elnyelődik egy szóköz és egy kapcsos zárójel-pár. Így például a `‘t␣{t}y` és a `‘tty` ugyanazt eredményezi.

Felmerül a kérdés, hogy miért van szükség az aktív karakterre, miért nem tudja a TEX a dupla többjegyű mássalhangzókat elválasztani. A válasz az, hogy a TEX automatikus elválasztó rutinját úgy alkotta meg a TEX írója, hogy egyedül feltételes kötőjel beszúrására legyen képes, nekünk viszont egy extra *y*-t is be kéne szúrni a fenti esetben (az automatizálás további lehetőségeit lásd a 3.2.5. alalszakasz. A szerzők gyakran megfeledkeznek az aktív karakterről – szerencsére az elválasztási minták (lásd a 3.3. alszakaszban) olyanok, hogy ekkor sem keletkezik hibás elválasztás. Az aktív karakter használatának fura mellékhatása, hogy két értelmetlen szó keletkezik (fent `fat` és `tyúsor`), melyet a generált elválasztási minták valószínűleg rosszul választanak el, tehát ekkor egyedül a huhyph3.tex választ el jól (lásd a 3.3. alszakaszban), például a `verembepo‘ttyantott` (egyébként értelmetlen) szó két darabját a huhyphn.tex így választja el: `ve-remb-epoty¬-tyan-tott`. A dokumentum helyesírás-ellenőrzése is egyszerűbbé válna, ha nem lenne aktív karakter.

Az aktív karakter egyéb parancsfunkciókat is ellát, például az *Euler – Fermat tétel* közökkel határolt gondolatjele így vihető be: `‘--`. Érdemes megemlíteni még a *nátrium-klorid*ot (melynek elválasztásakor a sor elején is kötőjel jelenik meg), itt a forrásban kötőjel helyett egyszerűen `‘|` gépelendő. A `‘-` aktív parancs hatására olyan kötőjel szúródik be, melynek mindkét oldalán el lehet választani a szót (a sima `-` két oldalán a TEX nem választ el automatikusan) Az aktív karakter egyéb funkcióinak leírását lásd [24]-ben.

A fő aktív karakteren kívül a magyar.ldf egyes írásjeleket is aktívvá tesz (*? ! : ;*), mert csak így szúrható be eléjük a magyar tipográfiában előírt köz automatikusan. Tehát a szerzőnek (`activespace=yes` betöltési opció mellett) csak `hogyan?`-t vagy `hogyan␣?`-t kell gépelnie, és a kérdőjel automatikusan távolabb kerül: *hogyan ?*. A fenti írásjeleknek matematikai módban speciális jelentésük van, ezért ott nem szúródik be eléjük köz. Ha egy teljes szóközt szeretnénk hagyni az írásjel előtt (mint a fenti felsorolásban), akkor pl. `?` helyett `␣\string?` gépelendő (a `␣\null?` a teljes és az extra szóközt is kihagyja). Hasonló a helyzet a „(!)” esetén, ami így írandó: `(\string!)`. A megnövelt köz automatikusan létrehozható lenne a `\catcode‘!=13 \edef!{\unskip\kern.1em \¬string!}` definícióval. (A `\string` használata itt elkerüli a végtelen rekurziót, az `\edef` pedig a `\string`-et a definíció idején fejti ki, ezzel gyorsítva a definiált makrót.) Ennél a magyar.ldf intelligensebb, mivel matematikai módban az `activespace=` opció nem módosítja a faktoriális helyét, és a karakterek csak akkor aktívak, amikor a magyar az aktuális nyelv. A babel karakteraktiválásából fakadó előny, hogy az aktív karakterek szabadon helyezhetők strukturális egységek címeibe, továbbá az `\az`, `\ref` és `\told` parancsok

argumentumába.

Az aktív karakter a más csomagokkal való összeakadás egyik fő oka. Egy karakter aktívvá tétele a magyar.ldf után betöltött csomagokra van hatással: elronthatja a makróargumentumok mintaillesztését, vagy olyan helyen is makrókifejtést okozhat, ahol a csomag tervezője nem számít rá (például a `\catcode‘t11␣` hívásban a `‘t` string a *t* betű karakterkódját jelenti, és semmi köze a `‘tty` dupla kétjegyű mássalhangzóhoz – az aktív karakter mégis így értelmezi, és ez hibához vezet (a hibaüzenetben nem utal arra, hogy a hiba a magyar.ldf aktív karaktere miatt keletkezett; ezt a felhasználónak ki kell találnia). A `\string` inaktiválja a karakter egy példányát, ezért a fenti példa így javítható: `\catcode\string‘t11␣`. Sajnos erről a csomagok készítői általában megfeledkeznek, ezért a magyar.ldf a lehető legkésőbbre halasztja a ‘ karakter aktiválását, és nyelvváltáskor inaktiválja. Elhelyez továbbá egy `\catcode\string‘‘12` sort az `.aux` fájlba, hogy e fájl betöltésekor a karakter inaktív állapotban induljon. Ezek a módosítások megszüntették a legfontosabb összeakadásokat (graphics és hyperref csomagok). Elképzelhető, hogy egy jövőbeli csomaggal a ‘ karakter mégis összeakad, ezért a magyar.ldf lehetővé teszi a " vagy a ’ aktiválását a ‘ helyett (`activeprefix=` opció), továbbá elérhetővé teszi az aktív karakter szolgáltatásait az `\shu` paranccsal (`shorthandcs=` opció) aktiválás nélkül. Az alapértelmezés a biztonságos aktív karakter és a `\shu` egyidejű engedélyezése. A magyar.ldf védekezik a már betöltött csomagok által aktívvá tett karakterek ellen: rögtön a fájl elején 12-re változtatja a következő karakterek kategóriáját: `‘ ! & + - = | ; : ’ " ? /`, a fájl végén pedig visszaváltoztatja a kategóriákat. Emiatt a magyar.ldf akkor is jól működik, ha a `! ? ;` és `?` karaktereket aktiváló frenchb.ldf már be van töltve.

Volt korábban egy rövid időszak, amikor a magyar.ldf-fel készült dokumentumok csak hibaüzenettel fordultak le, mert a magyar.ldf aktiválta a ‘ karaktert, ami zavart okozott az `.aux` fájl beolvasásakor. (Ezt megoldotta a fenti javítás az `.aux.` fájl elején.) A magyar.ldf korábbi verzióiban például a hyperref, a graphics és a graphicx csomagokat a babel csomag előtt kellett betölteni, hogy a magyar.ldf ne aktiválja túl korán a ‘ jelet. A túl korai aktiválás igen misztikus hibákat idéz elő, a hibaüzenetből a képzetlen felhasználó nem tudja meg, hogy az aktív karakterrel van probléma, sőt, például a hyperref csomag esetén a dokumentum lefordul, de téves lesz a strukturált tartalomjegyzék (*outlines*, *bookmarks*) a PDF fájlban.

A magyar.ldf 1.5-ös verziója többféle megoldást is ad az aktív karakter okozta problémákra. A szerző az `active=` betöltési opció `=none` értékével teljesen letilthatja az aktív karaktert (ami megoldja ugyan az összes kompatibilitási problémát, de csökken a funkcionalitás); az `=onlycs` értékkel pedig az aktív karakter helyett `\shu` parancsot definiálja (tehát például `po‘ttyan` helyett `po\shu‘ttyan` gépelendő: nincs kompatibilitási probléma, teljes a funkcionalitás, de többet kell gépelni). Az `activeprefix=` betöltési opcióval szabályozható, hogy a ‘ jel helyett a ’ vagy a " legyen aktiválva. Mivel ezeket is használják más csomagok, az `activeprefix=` nem növeli a kompatibilitást, csak választási lehetőséget biztosít a szerzőnek. Ide tartozik még a már bemutatott `active=` is.

A magyar.ldf új verziója akkor is csökkenti a kompatibilitási gondokat, ha van aktív karakter. Ennek része, hogy a karaktert csak a `\begin{document}` kiadásakor (vagyis a preambulum végén) aktiválja, így a babel után betöltött csomagokra az aktív karakter nincs hatással, továbbá a karakter aktívsága megszűnik, amint más nyelvet (pl. az angolt) választunk a dokumentum szövegén belül. Sajnos a megoldás nem terjed ki a babel csomag által támogatott többi nyelvre, így például ha a preambulumban `\usepackage[french,magyar]{babel}` szerepel, akkor a ‘ jelet a frenchb.ldf betöltése túl korán aktiválja – ez nem a magyar.ldf hibája, hanem tervezési hiba a babel-ben. A magyar.ldf a babel által kínált módnál precízebben ügyel arra, hogy a tartalomjegyzékben szereplő

címekben is lehessen az aktív karaktert használni; további információ a `toclang=` opció leírásában.

3.2.5. Dupla többjegyű mássalhangzók elválasztása aktív karakter nélkül

Ligatúrákkal és speciális elválasztási mintákkal elvileg megoldható például a *tty* kettőzött betű elválasztása (ty-/ty). A továbbiakban az egyszerűség kedvéért feltesszük, hogy csak a *tty*-ről van szó, és utána magánhangzó következik (tehát az elválasztás lehetséges). Az alábbi egyszerű megoldás sajnos nem helyes: a `\patterns{t1ty}` minta és egy `t + - → ty-` speciális ligatúra felvétele. A ligatúra ugyanis *fut-szalad* szóban is aktiválódik, és *y* szúródik be a kötőjel elé, továbbá a *botcsinálta* szóba a sima elválasztási minták az összetétel határán feltételes kötőjelet helyeznek el, és a ligatúrával együtt sor végén ezt kapjuk: *boty-/csinálta*. A TeX ligatúraprogramozási nyelve igen egyszerűsített, de nagyobb fáradsággal és sok karakterpozíciót elfoglalva létrehozhatók benne környezetfüggő ligatúrák, vagyis kettőnél hosszabb stringek automatikus transzformációi. Arra azonban vigyázni kell, hogy a feltételes kötőjel a tőle jobbra álló karakterrel sosem alkot ligatúrát, tehát a `\patterns{tt1y}` alapú minta eleve reménytelen, mert a második sor elejére kerülő *y* elé már lehetetlen beszúrni a *t* betűt. Marad a `\patterns{t1ty}` alkalmazása megfelelő virtuális fontokkal és ligatúrákkal. Az alábbi megoldás kipróbálható az aer10.vf virtuális font ligatúraprogramjának (`LIGTABLE`) módosításával (a konverzióra a vftovp és a vptovf használhatók). Az elkészült program így fest:

```
(LABEL C t) (/LIG/> C - C M) (STOP)
(LABEL C M) (LIG    C - C L) (STOP)
(LABEL C L) (LIG/>  C s C -) (STOP)
```

Ez ekvivalens az alábbi C nyelvű függvénnyel:

```
void atalakit(char *s) {
  for (;*s!='\0';s++) {
    if (s[0]=='t' && s[1]=='-')      {memmove(s+2,s+1,strlen(s));*++s='M';}
    else if (s[0]=='M' && s[1]=='-') {memmove(s+1,s+2,strlen(s+1));*s='L';}
    else if (s[0]=='L' && s[1]=='s') { *s='-'; }
  }
}
```

Úgy tekinthetjük, hogy a TeX először az elválasztási minták alapján elhelyezi a feltételes kötőjeleket, majd meghívja minden résszóra az `atalakit` függvényt, úgy, hogy a feltételes kötőjelek a résszavak végére kerülnek. (Ennél kicsit bonyolultabb a helyzet, mert a feltételes kötőjel csak akkor kerül be a szóba, ha tényleg a szó közepén törik meg a sor.) Így a *pottyan* szóból *pottyan* marad sor közepén, és *potL/tyan* lesz megtörve, a *fut-szalad* szóból pedig (mivel a kötőjel nem feltételes) *fut-szalad* vagy *fut-/szalad* lesz. A virtuális fontot úgy kell kialakítani, hogy az itt *L*-lel jelölt karakter helyén egy *y*- alakú betű álljon. A fenti megoldás hátránya, hogy se elválasztási mintákkal, se kivételekkel nem tudja biztosítani, hogy a *póttyúk* „szó” elválasztásánál ne kerüljön be *y*, de a szó mégis el legyen választva; a felhasználónak minden előfordulásnál `pot\-tyuk`-ot kell gépelnie. Hasonló probléma van a *kút- és gödörásás* kifejezéssel, ennek első szavát `kűt-{}`-ként kell bevinni, ellenkező esetben egy fölösleges *y* jelenik meg.

A hátrányok tehát a következők: minden betűtípushoz egy virtuális fontot kell generálni összesen két speciális karakterrel (`M` és `L`), továbbá az elválasztás még mindig nem teljesen átlátszó a szerző számára, mert néha `{}`, néha pedig `\-` jelet kell gépelnie. Sőt, a megoldás rosszabb, mint a magyar.ldf mostani, aktív karakteren alapú módszere, mert ha a fenti speciális jeleket elfelejti kitenni a szerző (például azért, mert a szöveget egy másik

programból importálja), akkor fölösleges betűk jelennek meg, viszont az aktív karakteres megoldásban a betűk mindig helyesek maradnak, akkor is, ha a szerző nem használja az aktív karaktert. Ezek alapján az itt ismertetett módszer nem ajánlott, és a TeX elválasztási és ligatúramodellje nem is teszi lehetővé általánosabb megoldás kifejlesztését. Egy igazán a korrekt megoldás kiterjesztené ezeket a modelleket, és igénybe venne egy szótövező és szóösszetétel-felbontó programot is. Ehhez viszont a TeX forrásába kéne belenyúlni, ami kívül esik a magyar.ldf keretein.

Megjegyezzük, hogy a németben is van egy hasonló probléma a *ck* elválasztásakor, például a *Brücke* szó elválasztott alakja *Brük-/ke*. Erre sem született még kielégítő megoldás.

3.2.6. Az utalások feloldása

Bár az utalások feloldásának nincs közvetlen magyar vonatkozása, ismeretét feltételezik a későbbi alalszakaszok.

Ha a dokumentum utalásokat tartalmaz (pl. lásd az 5.6. szakaszt, a 42. oldalon), ezeket csak akkor lehet feloldani, ha az utalás célpontjának (más néven: címke) pontos helye ismert. Mivel a TeX az elejétől a vége felé haladva fordítja le a dokumentumot, nincs lehetőség előre- vagy visszanézésre. A LaTeX az alábbi trükköt veti be a probléma feloldására: az `.aux` fájlba futása során feljegyzi a már elért célpontok pontos helyét (strukturális egység száma és oldalszám). Mire végigfut, már feljegyezte az összes célpontot, de maguk az utalások még csupa kérdőjelből állnak. Végül egy figyelmeztető üzenetet jelenít meg a terminálon, hogy még egy futtatásra szükség van. A következő futtatás elején betölti az `.aux` fájlt, és az utalásokat *előző* futtatás célpont-helyei alapján generálja. Ez nem baj, mert ezek a helyek általában nem változnak a két futtatás között, de ha mégis megváltoznának, akkor a LaTeX figyelmeztet, és le kell futtatni harmadjára (és lehet, hogy még többször…), mire beáll az az állandósult állapot.

Nem adható felső becslés arra, hogy a LaTeX-et hányszor kell futtatni. Sőt, léteznek patologikus esetek esetek is. Az alábbi példa sosem állandósul, hanem két állapot között oszcillál:

```
\documentclass{article} \begin{document} \setcounter{page}{9}
\vsize=2\baselineskip \hsize=137pt \font\f=cmr10 \f \parindent=0pt
  \pageref{w} i x x x x x x x x x x x
  aaaaaaaaaaa , , , , , , , , , , , w\label{w}
\end{document}
```

Ennek az a magyarázata, hogy a $k+1$-edik futtatásban a 9. oldal elején levő `\pageref{w}` 10-et generál (mert a `w` a 10. oldalon volt a k-adik futtatásban). Igen ám, de a k-adik futtatásban a `w` kifér a 9. oldalra, tehát a generált (10) és a tényleges (9) oldalszám nem stimmel, új futtatásra van szükség. A $k+2$-edik futtatás 9-es oldalszámot generál (mert az előző futtatásban ez a tényleges oldalszám), de ekkor a dokumentum legelső sora laza (*loose line*) lesz, vagyis szélesek lesznek a sorkizárást végző szóközök. Emiatt a TeX tördelő rutinja a második sornak nem engedi meg, hogy szoros (*tight line*) legyen, vagyis ott is megnöveli a szóközt, ezért a `w` átcsúszik a harmadik sorba, de az már a 10. oldalra kerül. Tehát a generált (9) és a tényleges (10) oldalszám megint különbözik, újabb futtatásra van szükség. A kör bezárult.

3.2.7. Csillagok a lábjegyzet élén

A csillagos lábjegyzetek* jeleinek generálásához szükség van egy olyan függvényre, amely n db csillagot jelenít meg. Bármilyen általános célú programozási nyelvben egy ilyen függvény elkészítése alapfeladat – de TeX esetén ez kihívást jelenthet, ha a makrót kifejthetőre (*expandable*) kívánjuk megírni ([10, 212. oldal] tételesen felsorolja, hogy mely nyelvi konstrukciók kifejthetőek – egy kifejthető makróból csak ilyeneket szabad hívni). A csillagok generálásának problémáját már [10, 373. oldal] is felveti, de nem ad kifejthető megoldást. A [8]-ban közölt szellemes trükk viszont megoldja a problémát: a `\romannumeral`-nak $1000 \cdot n$-et kell megadni, és ez épp n db `m` betűt generál. Az `m` betűk csillagokká alakíthatók egy egyszerű makróval. Íme a megoldás (az `\mfstar` makró a `\c@foootnote` számláló értékének megfelelő `m` betűvé fejtődik ki):

```
\def\mtostar#1{\if#1m*\expandafter\mtostar\fi}
\def\mfstar{\expandafter\mtostar\romannumeral\number\c@footnote000A}
```

A fenti `\mtostar` makró törzsében levő `\expandafter` nélkül a rekurzív hívás a `\fi`-t, és nem a következő `m` betűt kapná az argumentumába.

3.2.8. Oldalanként újból egyetlen csillag

A magyar hagyományok szerint a csillagos lábjegyzet-számozást oldalanként 1-től (vagyis egyetlen csillagtól) újra kell kezdeni – mint ahogy e dolgozat lábjegyzeteiben is látható. Ez a szabály nem egyeztethető össze a TeX oldalkészítő modelljével. A modell a következő: a bekezdések sorai (és egyéb elemek, például kiemelt matematikai képletek és szövegközi táblázatok) egy függőleges listába listába kerülnek. A lista akár másfél–két oldal magasságúra is megnőhet, mielőtt a TeX eldönti, hogy hol törje el az oldalt. Az oldaltörésnél figyelembe veszi az eddig felgyülemlett úszó objektumokat (ábrák, képek, lábjegyzetek, széljegyzetek) is, és mindent összevetve egy bonyolult algoritmussal ([10, 110. oldal]) olyan töréspontot választ, melyben a függőleges közöket kevéssé kell széthúzni, és a törésnél fellépő büntetőpont is alacsony. A lábjegyzet is úszó objektum, tehát jóval *azelőtt* el kell készíteni, hogy tudnánk, hányadik oldalra is fog kerülni (nem tudhatjuk biztosan, hogy az n-edikre vagy az $n+1$-edikre, esetleg még későbbre kerül-e, mert az oldaltörési algoritmus még nem kezdett az aktuális bekezdés környezetével foglalkozni). Mivel nem tudjuk mely lábjegyzet mely oldalra kerül, arra sem tudunk válaszolni, hogy az aktuális lábjegyzet vajon az oldal első lábjegyzet-e, tehát újra kell-e kezdeni a számozást.

A szükséges információt utalások létrehozásával szerezhetjük meg: minden, szövegen belüli lábjegyzetjel mellé egy címkét hozunk létre (lásd a 3.2.6. alalszakaszban), és a lábjegyzet felépítése előtt a csillagokat számoló `\c@footnote` számlálót lenullázzuk, ha az `.aux` fájl tanúsága szerint az előző futtatásban ez a lábjegyzet volt az első az adott oldalon. Ezt teszi a magyar.ldf, amikor a `\footnotestyle{starplain}` beállítás aktív. Implementációs nehézség volt a LaTeX lábjegyzet-létrehozó kódját úgy módosítani, hogy ez lehetőleg ne ütközzön más csomagokkal. E sorok írója azt a megoldást választotta, hogy a `\c@footnote` megnövelésekor lefutó kampó (*hook*) eljárás elejére vette fel a feltételes nullázást. (E kampó eljárások eredeti célja a LaTeX-ben, hogy például új szakasz kezdésekor az alszakaszszámláló lenullázódhasson.) A magyar.ldf-beli megoldás együttműködik a footmisc, ftnright és más csomagokkal.

Felmerülhet a kérdés, hogy a címkét magát hogyan lehet helyes oldalszámmal létrehozni, ha az oldalszám nem ismert a címke készítésének pillanatában, mivel az adott

* mint ez is

oldallal az oldaltörési algoritmus még nem kezdett el foglalkozni. A válasz ([10, 226. oldal]): a címke készítése során a `\write` parancs egy egysoros `\newlabel` bejegyzést ír az `.aux` fájlba, amely a strukturális egység számát és az oldalszámot tartalmazza. A `\write` parancs által kiírandó szöveget a TeX csak az oldal kiadásakor (*shipout*) fejti ki: ekkor az oldaltörési algoritmus már lefutott, és az argumentumban levő `\thepage` parancs helyére a helyes oldalszám kerül. Arról, természetesen, külön kell gondoskodni, hogy a strukturális egység száma azonnal kifejtődjön, tehát pl. egy szakasz a saját számát kapja, és ne az oldal kiadásakor aktuális magasabb értéket.

3.2.9. A határozott névelő generálása

A műszaki–tudományos művek bővelkednek a dokumentumon belüli utalásokban. A magyar nyelvű generált utalásokhoz, mint például „lásd a 45. oldalon" és „az 5.7. táblázat első oszlopa", szükség van az *a/az* határozott névelő generálására. A magyar.ldf az egyszerűség kedvéért csak egész számokra szorítkozik. Ha az utalás több számból áll, csak az elsőt kell figyelembe venni. Minden szám *a* névelőt kap, kivéve az 5-tel kezdődő pozitív számok, továbbá azon 1-gyel kezdődő pozitív számok, melyekben az 1-es utáni számjegyek száma osztható hárommal (pl. *az ezerötödik*, *az egymillió*, *az egymilliárd-kétszázezerből* stb.). A szabály viszonylag egyszerű, és az alábbi C nyelvű függvény meg is oldja a névelő generálását: `char* aaz(int i) { return i==1 || i==5 || i>49 && i<60 || i>49-9 && i<600 || i>999 && aaz(i/1000) ? "az" : "a";}`. A betűvel kiírt szavakra ez a szabály: a (nagybetűsítés és az ékezetek eltávolítása után) A, E, I, O, U betűkkel kezdődő szavak, továbbá az F, L, LY, M, N, NY, S, SZ, R, X, Y betűket jelölő szavak kapnak *az* névelőt, a többieknek *a* jár. (A magyar.ldf korábbi verziója az LY, NY és SZ szavakat nem vette figyelembe.) A magyar.ldf-ben a határozott névelőt megállapító kód kb. 5 kB-ot foglal. Ez a meglepően nagy terjedelem részben a TeX kevéssé hatékony programozhatósága miatt van, részben pedig azért, mert az argumentumban kapott tokenlistát speciális környezetben kell kifejteni (ékezetek nélkül, nagybetűsítve, a latin ligatúrák (pl. œ) konvertálásával, a parancsszavak és a kapcsos zárójelek figyelmen kívül hagyásával, a római számokat arabbá konvertálva, hiányzó utalási célpontok esetén sem hibát okozva). E speciális környezet felépítése és az utána következő nehézkes karakterenkénti olvasás tölti ki az 5 kB-ot. A TeX programozásában elmélyedni igyekvőknek érdemes áttanulmányozni a forráskódot, mert számos alaptrükköt alkalmazva a TeX makrókifejtési mechanizmusával egy egyszeri végigolvasású véges fordítást valósít meg.

A felkínált parancsok a következők: `\az{`*tokenek*`}` speciális környezetben (`\edef`-fel) kifejti a *tokenek*et, és az első egész szám vagy szó alapján generálja a névelőt, egy nem törhető szóközt, majd szedi a *tokenek*et. Az első szám vagy szó előtti jeleket figyelmen kívül hagyja. Az `\aref`, `\acite` és `\apageref` parancsok a megfelelő `a` nélküli utalóparancsok névelős változatai, például `\aref{oda}` és `\az{\ref{oda}}` ugyanazt eredményezi (csak a magyar.ldf új verziójában). Egyenletekre való hivatkozás is lehetséges, például `\aref({egy})` vagy `\az{(\ref{egy})}` szintaxissal (ez utóbbi csak az új verziótól). E parancsoknak léteznek nagybetűvel kezdődő változatai, melyek *A/Az* névelőt generálnak. `AZ` nem generálható (még a `\MakeUppercase` és `\uppercase` segítségével sem), és az `\az` makró és társai – technikai okokból – nem kifejthetők. A magyar.ldf új verziójának névelőgenerálása a fent felsorolt új szolgáltatásokon felül stabilabb az elődjénél, jobban viseli a speciális karaktereket, továbbá parancsai elhelyezhetők mozgó argumentumban.

A római számoknak is ugyanazt a névelőt kell kapniuk, mint arab megfelelőiknek. Sajnos a magyar.ldf (neve ellenére) nem érti a magyar nyelvet, ezért nem tudja érzékelni, hogy a `\az{x.\ oldalon}` hívásban római számmal, a `\az{$x$ tengely}` hívásban

pedig betűvel kezdődik (és mindenképpen betűnek veszi). Szerencsére utalások esetén megoldható a probléma, mert minden a számozás TeX számlálóregisztereinek valamelyikéből van generálva, és csak azt kell elérni, hogy a generáláskor a római számot eredményező `\romannumeral` helyett is az arab számot adó `\number` parancs fusson le. E módosított lefutás eredményét a magyar.ldf egy `\hunnewlabel` paranccsal helyezi el az `.aux` fájlba; a `\hunnewlabel` hasonló a LaTeX `\label` parancsa által elhelyezett `\newlabel`-höz, de az előbbi arab számot tartalmaz ott, ahol az utóbbi rómait. (Ha a `\newlabel`-ben nincs római szám, akkora `\hunnewlabel` nem is generálódik.) Az `\az` parancs belsejében levő utalóparancsok a névelő megállapításához mindig a `\hunnewlabel`-ben tárolt értéket veszik figyelembe, de a szedéshez a `\newlabel`-belit. Nem triviális az oldalszám helyes generálása: a 3.2.8. alalszakasz utolsó bekezdésében leírtak szerint azt szeretnénk, hogy a `\thepage` számértéke csak az oldal kiadásakor legyen figyelembe véve, de a `\thepage`-ben megbúvó formázást azonnal át szeretnénk írni `\number`-re. Ehhez először a `\def\romannumeral{\noexpand\noexpand\noexpand\number}` definíciót kell kiadni (és hasonlóakat a `\number`, `\@alph` stb. parancsokra), majd jöhet az `\protected@edef`. Ha a `\thepage` értéke `\@roman\c@page`, akkor ez `\noexpand\number\c@page`-dzsé fejtődik ki. Ezt a `\protected@edef` is kifejti, és a keletkező `\number\c@page`-et a `\write`-nak adja, tehát a végső kifejtés, a `\c@page` regiszter értékének felhasználása csak az oldal kiadásakor történik meg, mégpedig arab számmal. Pont ez volt a célunk.

3.2.10. Egész számok toldalékolása

Főleg tudományos művekben gyakori a számozott képletekre való toldalékolt hivatkozás. Egy túlzsúfolt példa: (1)-hez hozzáadva (2)-t és elosztva (3)-mal, és az eredményt kivonva (4)-ből kapunk egy (5)-nél nagyobb értéket, ami relatív prím (6)-hoz, ám ez (7)-ről nem mondható el. Praktikus okokból feltehetjük, hogy az utalásokban szereplő számok abszolút értékben kisebbek tízmilliárdnál. Az ilyen egész számok néhány paradigmaosztályba sorolhatók végződésük szerint (az utolsó nemnulla számjegyet és az utána következő nullák számát figyelembe véve). A paradigmaosztály egyértelműen meghatározza az összes támogatott toldalék hozzáillesztendő változatát. A *-val/vel* ragban bekövetkező teljes hasonulás miatt nem elég csak a hangrendeket megkülönböztetni. A magyar.ldf összesen 23 paradigmaosztályba sorolja a számokat. Egyjegyű számok esetén az előjelre is figyelni kell, például 2-odik, de -2-edik. Ha egy generált utalás több számból áll, akkor az utolsót kell figyelembe venni. A római számokat hasonlóan kell kezelni, mint a határozott névelő generálásánál (lásd a 3.2.9. alalszakaszban). A speciális kifejtési környezet felépítésénél is hasonlóan érdemes eljárni. Néha több toldalék is járulhat ugyanahhoz a számhoz, például 60-adszor. A magyar.ldf egy vagy két toldalékot támogat, például a 60-szor kiszedhető a `\told60+szor{}` és a `\told60+szer{}`, 60-adszor pedig `\told60+ad+szer{}` és még 3 másik hívással is. A `\told` és a `+` közé írt toldalékolandó szó tartalmazhat utaló parancsokat (pl. `\cite`) és azok névelős változatát is (pl. `\Apageref`). A 17. oldalon található „Knuth már a [10]-ben megemlíti” például így let begépelve: `Knuth már \told\acite{texbook}+ban{} megemlíti`. A hátsó toldalék az *adik*, *a*, *as*, *on*, *nál*, *ül*, *val*, *án*, *at*, *höz*, *ban*, *nak*, *ba*, *ra*, *ból*, *tól*, *ról*, *ször* ékezet nélküli alakjaiból és ezek magas hangrendű változataiból (pl. *ben*) kerülhet ki, míg a kéttoldalékos változat első eleme *as*, *ad*, *adik* vagy *a* lehet, a magas hangrendű alakok itt is megengedettek.

Az elkészített `\told` makró csak egész számokat tud toldalékolni, például hónapneveket nem. A dokumentum strukturális egységeit jelölő köznevek (pl. *egyenlet*, *ábra*, *szakasz*) toldalékolása csak részben megoldott, mert például az *ábrát* szóban fellépő magánhangzó-hosszabbodást nem kezeli, és a *-val/vel* rag hozzátétele sem megoldott. Az

egységeket jelölő nevek nem érhetők el közvetlenül a \told paranccsal, ezeket a \refstruc generálja, lásd a következő alalszakaszban.

3.2.11. Hivatkozások a dokumentum strukturális egységeire

„A dolgozat a 4. szakaszszal zárul." Az előző mondatban nem csak a névelőt, a toldalékot és az utalás számértékét, hanem a strukturális egység nevét (tehát a *fejezet* szót) is a LaTeX generálta a magyar.ldf 1.5-ös verziójában bevezetett \refstruc paranccsal. A teljes hívás ez volt: `A dolgozat \az{\refstruc{pdi:utolso-valami+vel}} zárul.` A sok hivatkozást tartalmazó műszaki–tudományos művek írásakor gyakran van ilyesmire szükség, mert a szerző nem tudja előre, hogy az utalás célpontját képező strukturális egység milyen mélyen lesz a dokumentum végleges hierarchiájában. A \refstruc parancs ezt automatikusan kikövetkezteti az utalás számértékéből. Mivel a mélységet a LaTeX nem tároja az .aux fájlban, a számérték formázását és a pontok számát lehet csak figyelembe venni, például az *article* osztályba tartozó dokumentumok esetén (ilyen e dolgozat is) az „1" szám szakaszt, a „2.3" alszakaszt, a „4.5.6" alalszakaszt, az „A" betű függeléket jelöl stb. (Az „1" jelölhetne egyenletet, táblázatot vagy ábrát is, de további információ híján a \refstruc szakasznak veszi, kivéve, ha a nameref csomag be van töltve.) A támogatott strukturális egységek a következők: rész, fejezet, függelék, szakasz, alszakasz, alalszakasz, alalalszakasz, bekezdés, albekezdés és alalbekezdés. A \refstruc nem csak a pontok számára épít, hanem a következőképpen működik: az .aux fájlból kiolvasott utalásban (pl. `4.v.VI`) az arab számokat 1-essé, a kis római számokat `i`-vé, a nagyokat rómaiakat pedig `I`-vé alakítja, majd generálja a strukturális egységek számozásait (\thechapter, \thesection stb.) a számlálóregiszterekben (\c@chapter, \c@sectiom stb.) csupa 1-esekkel a kisebb egységtől a nagyobb felé haladva. Amelyik generált számozás egyezik az utalással, annak az egységnek a neve az eredmény. A függelékeket külön kezeli, mert az \appendix parancs felülírja a \thesection vagy a \thechapter parancsot (hogy melyiket, az az osztálytól függ). Visszaírás nélkül nem tudna a függeléken kívülről a függelékbe vagy fordítva utalni.

A nameref csomag elhelyezi a strukturális egység típusát is az .aux fájlba kerülő \newlabel parancs extra argumentumaiban. Ha ez az információ jelen van, akkor a \refstruc próbálgatás helyett felhasználja, és ezáltal az ábra, táblázat, egyenlet, lábjegyzet, elem, sor és tétel egységneveket is támogatja, megkülönböztetve őket egymástól és a fenti nevektől is.

hyperref csomag \autoref parancsa is hasonló a \refstruc-hoz, de az csak angol nyelven működik, a \refstruc pedig magyarul, ha a magyar nyelv aktív, és angolul (vagyis a LaTeX belső struktúraneveivel), ha más nyelv aktív. További előnye a \refstruc-nak, hogy szerepelhet az \az parancs argumentumában, tehát az utalás elé névelő tehető, továbbá a generált nevek toldalékolhatók a \refstruc argumentumának +-szal való kiegészítésével (ekkor a címek nem tartalmazhat +-t). Bizonyos toldalékok nem jól jelennek meg, ezek listáját lásd a 3.2.10. alalszakasz végén.

3.2.12. Lábjegyzetek

A magyar.ldf mindenben megfelel a 27. oldalon található előírásoknak, és biztosítja a \footnotestyle parancsot, mellyel a tipográfus beállíthatja a lábjegyzetek kinézetét (ennek opcióit nem részletezzük, de az alábbi listában láthatók példák). Az egyes előírásokhoz fűzött megjegyzések:

1. OK, a `\footnotestyle{mark=...}` paranccsal a kívánt számozás kiválasztható.

2. OK, a `\footnotestyle{starplain}` aktiválja a `\footnotestyle{mark=stars-max}`-ot.
3. OK, a szerzőnek kell figyelni rá, nehogy a `\footnote` parancs új sorban kezdődjön. Például: `Barack\footnote{Értsd: barackszínű.} ruha`
4. OK, a szerzőre van bízva.
5. OK, az `\editorfootnote` pont ezt csinálja.
6. OK, `\footnotestyle{rule=none}` vagy `\footnotestyle{rule=one-line}`.
7. OK, egy legtöbbször jól működő megoldást valósít meg, `\footnotestyle{rule=choose}`.
8. OK, ehhez külső csomag szükséges: `\usepackage[para]{footmisc}` (ajánlott) vagy `\usepackage{fnpara}` (régebbi). A footmisc csomag a `minipage` környezeten belüli lábjegyzeteket nem teszi egy sorba, fnpara csomag viszont igen.
9. OK, lásd `\makeFootnotable` lent.
10. OK, lásd `\footnotestyle{marksize=max-normal}`.
11. OK, lásd `\footnotestyle{indent=hulist}`.
12. OK, lásd `\footnotestyle{indent=article-sp}` és `\footnotestyle{indent=hulist}`.
13. OK, lásd `\footnotestyle{mark=stars-max}`. A hetedik csillag helyett ^{*7}-tel számozás kezdődik.
14. OK, [5, 116–118. oldal] előírása érvényesül.
15. Nincs megvalósítva, listaszerű lábjegyzet esetén minden jel balra, bekezdésszerűnél pedig minden jel jobbra van a behúzástól.

Bizonyos LaTeX parancsok argumentumai mozgó argumentumnak számítanak, mivel nem csak a kiadás helyén kerülnek kiszedésre. Ilyen például egy képaláírás (`\caption` parancs, argumentuma bekerül az ábrák jegyzékébe), vagy egy fejezetcím (`\chapter` parancs, argumentuma bekerül a tartalomjegyzékbe és az élőfejbe). A mozgó argumentumokban használható parancsokra sok korlátozás van, bizonyos hívások elé a `\protect` parancsot kell elhelyezni, és egyes parancsok nem is használhatók mozgó argumentumban. Ez utóbbi csoportba tartozik a `\footnote` parancs is. A magyar.ldf definiálja viszont a `\headingfootnote`-ot, amire a korlátozás nem vonatkozik.

3.2.13. Kétszeres betöltés

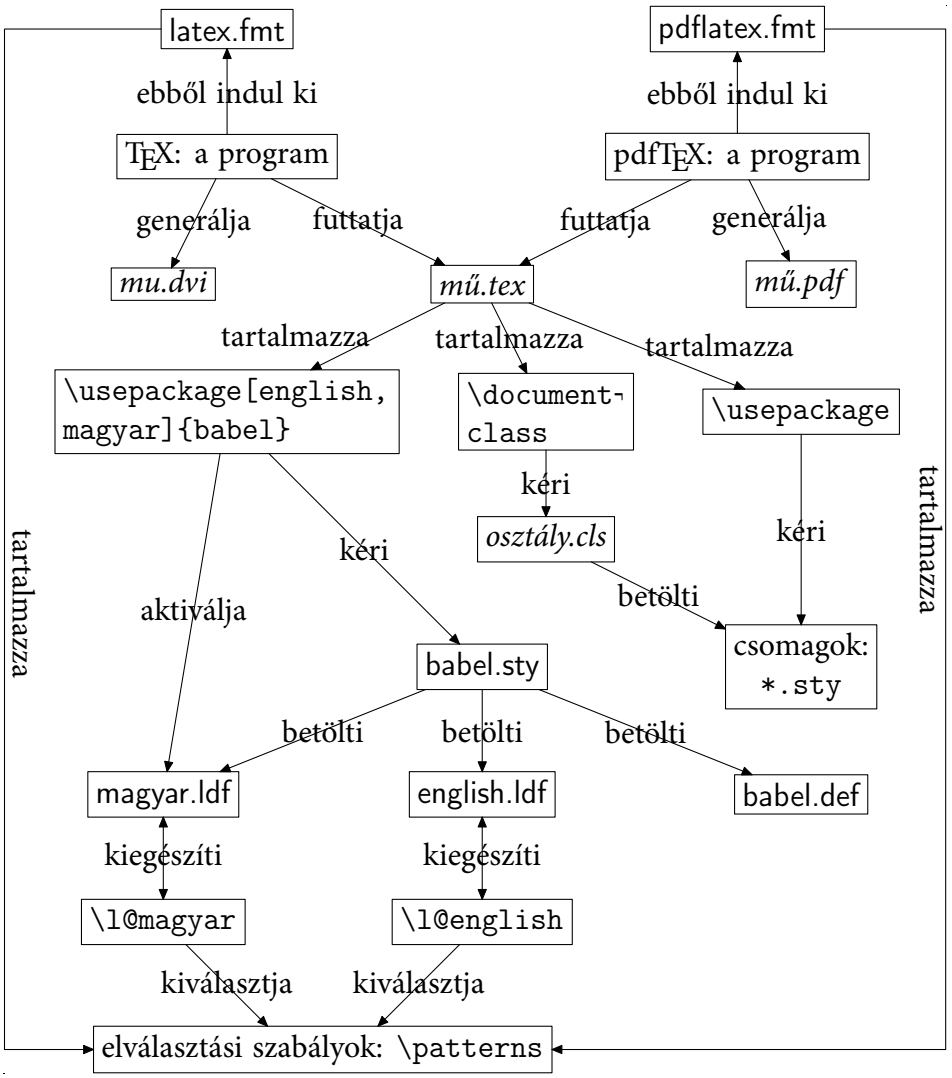
Egy dokumentumon belül a magyar.ldf-nek két példánya is használható különböző opciókkal. E szolgáltatás neve kétszeres betöltés (*kétszeres betöltés*). Ehhez a `\usepackage[hungarian,magyar]{babel}` hívással kell betölteni a babel csomagot, és előtte a `\PassOptionsToPackage{...}{hungarian.ldf}`-ben lehet a *hungarian*, és hasonlóan lehet a *magyar* nyelv opcióit megadni. A szolgáltatás nincs teljes körűen tesztelve, és biztosan nem működik minden opcióra (például `captions=`), mivel a két példány ugyanazokat a `\magyar@...` és `\@@magyar@...` kezdetű parancsneveket használja. Működik viszont a különböző `frenchspacing=` és `hyphenation=` beállítás. (Természetesen a többszörös betöltés korlátozás nélkül megvalósítható a kézi módszerrel: a magyar.ldf lemásolása új néven (de ne `hungarian.ldf` legyen), a másolatban a `magyar` string összes előfordulásának cseréje valami másra, és az új fájl nevének megadása opcióként a babel csomag betöltésekor.)

A kétszeres betöltés a magyar.ldf-en kívül más nyelv moduljában nem található meg. Valójában három- és többszörös betöltés is megvalósítható, ha a babel.sty fájl `hungarian`-t tartalmazó sorát lemásoljuk, és átírjuk a `hungarian`-t egy új névre. A többszörös betöltést a magyar.ldf úgy valósítja meg, hogy a babel csomag és a nyelvi modul közti interfészt

képező parancsnevekben (például `\extrasmagyar`, `\captionsmagyar`, `\l@magyar`) a fix `magyar` szó helyett a `\CurrentOption` értékét használja, ami megegyezik a babel csomag betöltésekor szögletes zárójelben megadott nyelvnévvel.

3.3. Automatikus szóelválasztás

A TEX az automatikus szóelválasztást az ún. elválasztási minták (*hyphenation patterns*) segítségével végzi. Az egyes nyelvekhez egymástól független minták tartoznak, sőt a magyar nyelvhez négy mintakészlet is született (áttekinti őket a a 4. táblázat, és néhány példával szemlélteti a különbségeket a az 5. táblázat). A mintákat a TEX formátumgeneráláskor veszi figyelembe, ezért a minták megváltozásakor a language.dat fájlt módosítani kell, és az iniTEX, fmtutil vagy lafmtgen.pl programok megfelelő paraméterezésével újra kell generálni a formátumot. Azt, hogy melyik nyelvhez melyik minta tartozzon, a babel csomag `.ldf` moduljai határozzák meg. Például a magyar.ldf új verziója a `hyphenation=` opcióval teszi lehetővé a választást a négy mintakészlet között; természetesen csak azokból lehet választani, melyek generáláskor belekerültek a formátumba. A formátumgenerálás, az elválasztás és a babel kapcsolatát a 3. ábra szemlélteti.



3. ábra. A formátumgenerálás, az elválasztás és a babel kapcsolata

A mintaalapú elválasztási modell teljes leírása [10, 449. oldaltól] olvasható, melyet az

alábbiakban összefoglalunk. Az elválasztási minták stringek, melyek betűket és számjegyeket felváltva tartalmaznak. Egy minta akkor illeszkedik egy szóra, ha a mintából a számjegyeket kitörölve a szó összefüggő részstringjét kapjuk. A szó elejére és végére egy pont, a minta elejére és végére, továbbá két betű közé pedig egy nulla képzelendő. Tekintsük az `á1gy`, `á1l` és `.vá2gy1á` mintákat. Ezek a `vágyálom` szóra az alábbi módon illeszkednek: $.^0\mathtt{v}^0_0\mathtt{á}^2_1\mathtt{g}^0_0\mathtt{y}^1_0\mathtt{á}^0_1\mathtt{l}_0\mathtt{om}.$; a szó azon két betű között választható el, ahol az egymás fölötti számok maximuma páratlan, jelen esetben két elválasztás lehetséges: *vágy-á-lom*. A példában az `a1gy` és az `a1l` teszik lehetővé az egyszerű szavak elválasztását magánhangzó és mássalhangzó között, a a leghosszabb minta (`.vá2gy1á`) pedig a *vágyá*val kezdődő szavakat a *vágy* után szóösszetételnek tekinti, és a mintában levő 2-es megakadályozza az egyszerű szavakhoz tartozó minták által engedélyezett elválasztást. Az összes magánhangzó-magánhangzó és magánhangzó-mássalhangzó párhoz tartalmaz 1-es mintákat, és a legfontosabb összetett szavakhoz 2-es mintákat a huhyph3.tex. A másik három magyar elválasztási mintagyűjtemény a PatGen programmal készült, amely bemenetén szavak helyesen elválasztott listáját várja, kimenetén pedig egy – a paraméterekkel szabályozható módon – optimális elválasztási mintagyűjteményt produkál. A fent ismertetett modell tömör, és megfelelő minták esetén gyors elválasztást tesz lehetővé. Átvette az OpenOffice is, a LibHnj programkönyvtár segítségével.

4. táblázat. A magyar elválasztási minták adatai

Név	Szerző	Évszám	Méret, bájt	Szófa hossza*	Op-méret*	Készítés
huhyph3.tex	Mayer Gyula	1998	23 693	5683	137	kézi
huhyphf.tex	Mayer Gyula	2002	38 358	7758	276	PatGen
huhyphc.tex	Mayer Gyula	2002	38 240	7735	282	PatGen
huhyphn.tex	Nagy Bence	2003	94 281	18104	117	PatGen

* A LaTeX formátum-generálás által kiírt statisztikából

5. táblázat. Példák a LaTeX magyar elválasztási mintáival. A hibás elválasztásokat kiemeltük

hyhyph3.tex	huhyphf.tex	huhyphc.tex	huhyphn.tex
Sha-kes-pe-a-re	Shakes-peare	Shakes-peare	Shakes-peare
fi-*zi-o*-te-rá-pi-a	fi-zio-te-rá-pi-a	fi-zio-te-rá-pi-a	fi-zio-te-rá-pi-a
szink-ron	szink-ron	szin-kron	szink-ron
meg-gyül-tet-vény	*meg-gyül*-tet-vény	*meg-gyül*-tet-vény	meggy-ül-tet-vény

TeX-ben az explicit elválasztást deklaráló legáltalánosabb hívás a `\discretionary¬{`*fel-jobb*`}{`*le-bal*`}{`*egybe*`}`, ami az *egybe* szöveget helyezi el, ha a sort a TeX nem ott választja el, különben pedig az elválasztás felső sorának jobb oldalára a *fel-jobb*, a következő sor elejére pedig a *le-bal* szöveget helyezi. Ezt felhasználva elhelyezhető például a soreleji kötőjel a *nátrium-klorid*ban: `nátrium\discretionary{-}{-}{-}klorid`. Ha azt szeretnénk, hogy a két részszót a TeX automatikusan elválassza (mint `‘|` esetén, lásd a 3.2.4. alalszakaszban), már egy bonyolultabb kód szükséges: `nátrium\nobreak-¬ \nobreak\discretionary{}{-}{}\nobreak\hskip\z@skip klorid` (ennek helyessége [10, 96–97. oldalak]-on írakból következik). A `\discretionary` használatára éles példa az e dokumentumban található szövegközi kódrészletek elválasztása a ¬ jellel; mivel a `\discretionary` nem generálható automatikus elválasztással, ezért ahhoz, hogy e jel megjelenjen, TeX makrók elhelyezik a megfelelő `\discretionary` hívásokat a kiszedendő kódrészletekben.

3.3.1. Az egyes mintagyűjtemények egyedi jellemzői

huhyph.tex vagy **huhyph3.tex** ♦ A minták 3-as verziója. Az elválasztási minták kézzel készültek, továbbá felsorol néhány közismert kivételt. Része a TEX-disztribúcióknak és a MagyarLATEX-nek.

huhyphc.tex ♦ A minták 4-es verziója. Jóval több összetett szót választ el helyesen, mint elődje. Az idegen összetett szavakat az összetétel határán választja el (pl. `szin-kron`). A minták a PatGen programmal készültek egy 70 000 db toldalék nélküli szót tartalmazó listából. Része a MagyarLATEX-nek. [13] részletesen leírja, hogy a minták készítője mely helyesírási szabályokat vett figyelembe (különös tekintettel az összetett szavak és az idegen szavak elválasztására), milyen megfontolások alapján dolgozott, és kitér a mintagenerálás technikai részleteire is.

huhyphf.tex ♦ A minták 4-es verziója. Az idegen összetett szavakat fonetikusan választja el (pl. `szink-ron`). A minták a PatGen programmal készültek egy 70 000 db toldalék nélküli szót tartalmazó listából. Része a MagyarLATEX-nek. [13] tartalmazza e minták részletes dokumentációját is.

huhyphn.tex ♦ Az idegen összetett szavakat fonetikusan választja el (pl. `szink-ron`). A minták a PatGen programmal készültek sok szótőből és sok szó toldalékolt alakjaiból (de nem az összes, hanem csak a készítő rendelkezésére álló szövegekben ténylegesen előforduló toldalékolt alakokból). Része a a MagyarLATEX-nek, és legújabb verziója letölthető a `http://www.tipogral.hu/` címről. Leírása és a másik három mintakészlettel való összehasonlítása [14]-ben olvasható (az, hogy pontosan hogyan készültek ezek a minták, igen nehezen hámozható ki a szövegből). E dolgozat is a huhyphn.tex-et használja.

3.3.2. Az elválasztást segítő fejlesztések

A fenti mintákat e sorok írója változtatás nélkül átvette, és a magyar.ldf `hyphenation=` opciójával mindegyiket kiválaszthatóvá tette. E dokumentum fordításához [14]-ben levő összehasonlítás alapján huhyphn.tex-et választotta.

Az elválasztási minták hatása lavinaszerűen akár több oldalra is kiterjedhet. Tegyük fel, hogy egy könyvet más elválasztási mintával újrafordítanak. A 96. oldalon e minták alapján lehetségessé válik egy eddig tiltott elválasztás, ezért az adott bekezdés egy sorral rövidebb lesz. Emiatt az 97. oldalra átvitt szakaszcím és a szakasz első bekezdése a 96. oldal aljára kerül, és a fejezet a 99. oldal helyett a 98-adikon fejeződhet be. Mivel minden fejezet páratlan oldalon kezdődik, ezért a következő fejezet a 101. oldal helyett a 99-ediken fog kezdődni. E fejezet oldalszámára a dokumentum egy más pontján van egy utalás, ami 101 helyett már 99; ez egy számjeggyel rövidebb, és az utalást tartalmazó bekezdés emiatt egy sorral rövidebb lesz, emiatt az oldalra több fér, a fejezet egy korábbi oldalon fejeződik be stb. Láthatjuk, hogy egyetlen elválasztás megváltozása akár 4 vagy több oldallal is megváltoztathatja a dokumentum terjedelmét.

Emiatt szükséges, hogy a dokumentum archiválásakor a fordításhoz használt összes fájl is mentésre kerüljön. (Ha verziókövető és/vagy párhuzamos fejlesztést támogató rendszert, pl. CVS-t használnak a szerzők, akkor elég, ha minden programot és stílusfájlt felvesznek ebbe a rendszerbe, és ha évekkel később újra akarják fordítani a dokumentumot, akkor a rendszerből lekérik az eredeti fordítás napján aktuális állapotot.) Nem szerencsés, hogy lényegesen különböző elválasztási minták ugyanazt a nevet kapják, ezért e sorok írója a 4. táblázatban látható fájlneveket javasolja, és azokat a MagyarLATEX-ben következetesen alkalmazza. A huhyph.tex fájl a TEX-disztribúciók része, és a másik négy

mintafájl közül bármelyik másolatát tartalmazhatja (jelenleg a teTeX disztribúcióban a huhyph3.tex található meg e néven). A magyar.ldf alapértelmezésben a huhyph3.tex-et próbálja meg betölteni, és csak, ha ez nem található meg, akkor tér vissza a huhyph.tex-re. Ez a működés segíti a képzetlen felhasználó munkáját, és a MagyarLaTeX-et tudatosan használó szerző archiválási szempontjait is figyelembe veszi.

Az elválasztási minták mindennapos használatát az alábbi szoftverek segítik:

CM-Super ♦ Az EC fontok Type 1 formátumú változata, minőségi PDF-ek készítésénél van rá szükség, ugyanis a LaTeX alapfontjai, a CM fontok magyar ékezetes betűket tartalmazó szavak automatikus elválasztását lehetővé tevő T1 fontkódolású változata csak `mf` formátumban található meg a TeX-disztribúciókban, ami PDF fájlokba pixeles Type 3 fontként kerül be, amit az Adobe Reader lassan és rosszul olvashatóan jelenít meg a képernyőn. Ennek orvoslására készítette Vladimir Volovich 2002-ben a CM-Super fontcsomagot, amely ugyanezeket a fontokat Type 1 formátumban tartalmazza, így gyorsan és szépen jelennek meg Adobe Readerben. A CM-Super fontok elkészítéséhez többek között a TEXtrace programot is használták, amely e sorok írójának egy korábbi alkotása. A CM-Super fontcsomag 50 MB-nál is nagyobb, és viszonylag bonyodalmas a telepítése. Az eredeti, legfrissebb verzió a CTAN-ról tölthető le. A MagyarLaTeX honlapján egy Debian csomag található, ami az ügyes tömörítésnek köszönhetően feleakkora, mint az eredeti (tömörítve).

lafmtgen.pl ♦ A formátumot a UNIX-os teTeX disztribúcióban automatikusan újrageneráló Perl szkript. E diplomaterv keretében készült el, része a MagyarLaTeX-nek. A kezdő vagy rendszergazdai jogosultsággal nem rendelkező felhasználókat segíti: egyszerűen megírják a magyar.ldf-et használó dokumentumot, bemásolják a megfelelő `huhyph?.tex`-et az aktuális könyvtárba, lefuttatják a `lafmtgen.pl` *dok.tex* parancsot, és a formátum előáll. A szkript a rendszergazdák munkáját is megkönnyíti, mivel a teTeX fmtutil parancsát és a language.dat konfigurációs fájl kézi szerkesztését is képes kiváltani. A lafmtgen.pl a olyan módosított formátumot generál, amely felismeri a fejléces UTF-8 kódolású forrásfájlt, és automatikusan alkalmazkodik hozzá. Ez a funkció azért a lafmtgen.pl-be került, mert csak a formátumban valósítható meg.

ccs_extract.pl ♦ A dupla többjegyű mássalhangzókat tartalmazó szavak (lásd a 3.2.4. alalszakaszban) elválasztását segítő Perl szkript. E diplomaterv keretében készült, és része a MagyarLaTeX-nek. A többjegyű mássalhangzók aktív karakterrel (`‘`) való automatikus ellátását segíti.

Ha például a `mű.tex` (esetleg többször) tartalmazza a `loccsan`, `hosszú`, `leggyakoribb` és `mennyi` szavakat, akkor a `ccs_extract.pl mű.tex` parancs hatására az alábbi sorok a `mű.tex` végéhez fűződnek:

```
%‘% ho‘‘sszú 5
%‘% le‘‘ggyakoribb 10
%‘% lo‘‘ccsan 6
%‘% me‘‘nnyi 3
```

A fenti listában a számok arról tudósítanak, hogy a fájlban hányszor fordult elő `‘`-ok nélkül az adott szó. Ezután a szerzőnek a szövegszerkesztőben a megfelelő `‘‘`-ok egyik aposztrófját törölnie kell, ha a cserét végre szeretné hajtani. Példánkban a `le‘‘ggyakoribb` változatlanul hagyandó, mivel szóösszetételről és nem dupla többjegyű mássalhangzóról van szó, a másik három szó viszont módosítandó a fájl végén: `ho‘‘sszú`, `lo‘ccsan` és `me‘nnyi`. A `ccs_extract.pl mű.tex` parancs újbóli lefuttatása $5 + 6 + 3$ aposztrófbeszúrást végez a fájl belsejében. Mostmár újra lehet fordítani a `mű.tex`-et, és a LaTeX el fogja választani a dupla többjegyű mássalhangzókat is. Ha a

dokumentum változik, akkor a ccs_extract.pl-t újra kell futtatni, a fájl végén keletkező %‘% sorokban a dupla ‘ jeleket szimplára kell cserélni, majd újra le kell futtatni a ccs_extract.pl-t. A ccs_extract.pl tehát azáltal gyorsítja a munkafolyamatot, hogy a szerzőnek nem kell a dupla többjegyű mássalhangzókat tartalmazó szavak *összes* előfordulását felkutatnia és egyenként lecserélnie.

E dolgozat írásának egy köztes fázisában a ccs_extract.pl több, mint 150 db dupla többjegyű mássalhangzókat tartalmazó szót talált, ezek közül el nem választhatók voltak: *“ttyan kulc“sszavakat kulc“sszavakra kulc“sszóra le“ggyakoribb le“ggyakoribbnak le“ggyakorlatiasabb le“ggyorsabban me“ggyőződését paranc“sszavait paranc“sszavak.* (E listában a `\string‘{}\string‘` hívással lett szedve a szavak közepén levő ‘‘ jel, mivel a magyar.ldf aktiválta a ‘ jelet.)

Az ccs_extract.pl elvárja, hogy `latin2`-es kódolásban legyenek az általa feldolgozott fájlok. Az ékezetes betűket többkarakteres formájukban nem ismeri fel. Nem okoz gondot a programnak, ha egy szóban több dupla többjegyű mássalhangzó szerepel, például *mennyiségigénnyel*.

3.4. Tárgymutató (index) készítése

[6, 226. oldal] szerint „számítógéppel is rendezhetők a tárgymutató tételei, ez azonban ma még nagyon költséges. Nem is takarítunk meg vele időt, mert a cédulára felírt tételeket le kell gépelni a számítógép kezelője részére. Igaz, nem ábécérendben, hanem ahogy sorba kiírtuk. A számítógépes kiírás azonban nem felel meg a magyar helyesírásnak, azt át kell gépeltetni, tehát az ábécérendbe rakás helyett még egyszer le kell gépelni a tárgymutatót.” A könyv 1982-es kiadása óta a dokumentum-előkészítés módszerei sokat változtak. A számítógéppel készülő művek esetén nem kell számolni plusz begépelési idővel, továbbá ma már léteznek olyan programok, melyek a tárgymutatót a magyar helyesírás szerint rendezik.

A LaTeX-es dokumentum tárgymutatója az alábbi módon készül: a szerző a forrásban az `\index` paranccsal megjelöli, hogy mely szavak kerüljenek a tárgymutatóba. Például a `\index{LaTeX@\LaTeX!\~{} telepítése}` hívás egy kétszintű bejegyzést hoz létre: a „LaTeX”-en belülre kerül a „~ telepítése” tétel. Az oldalszám pedig az `\index` parancs forrásfájl-beli helye alapján generálódik. A példabeli kukac a „LaTeX” szó két alakját választja el: `LaTeX` alapján sorolódik ábécérendbe, de a megjelenítését a `\LaTeX` parancs végzi. Az `\index` parancsok elhelyezése után a tárgymutató előállítása automatikus.

Az `\index` parancsokkal ékesített dokumentumból a LaTeX fordítás egy `.idx` fájlt készít, melyben már az oldalszámok is szerepelnek. Az `.idx` fájlt egy ún. indexrendező segédprogrammal (*index processor*) kell rendezni, ennek kimenete egy `.ind` fájl, melyet a LaTeX a következő fordításkor beépít a dokumentumba. Az indexrendező program ábécérendbe rakja és kezdőbetű szerint csoportosítja a tételeket, továbbá intervallumokat képez az egy tételen belüli oldalszámokból. (Az indexrendező program futási ideje a LaTeX-fordítás egyéb műveleteihez képest elhanyagolható, például [24] félkész, 750 oldalas változatának 35 oldalas tárgymutatóját a husort.pl egy kb. 5 évés számítógépen 10 másodperc alatt rendezte, míg a LaTeX-fordítás 3 percnél is tovább tartott.) A legfőbb különbség az egyes indexrendező programok között az ábécérend és a kezdőbetűk definiálásában mutatkozik.

Az indexrendező programok és magyar kiegészítőik a következők:

makeindex ♦ 1987-ben (!) született indexrendező program LaTeX-hez. Mai verziója csak az angol és a német nyelvet támogatja (az utóbbit is csak korlátozásokkal). Előnye, hogy gyors, megbízható, és nem kell egy új, inkompatíbilis verzió megjelenésétől tartani.

Mivel a makeindex nem a magyar ábécé szerinti helyükre sorolja az ékezetes betűket, a fenti `\index` hívás helyett ezt kellett volna írni: `\index{LaTeX@\LaTeX!@LaTeX␣ telepitese!\~{} telepítése}`. Ez sem tökéletes megoldás, mert nem foglalkozik például dupla többjegyű mássalhangzókkal, továbbá azzal sem, amikor két tétel csak az ékezetekben különbözik egymástól. Az ilyen eseteket csak a tárgymutató elkészülte után csak annak igen figyelmes végignézése alapján lehet javítani – az átnézés igen fáradságos, és ennek ellenére könnyen elkerülheti egy rossz helyen levő szó az ember figyelmét. Összegzésül: a makeindex nem ajánlott magyar nyelvű tárgymutató készítésére.

husort.pl ♦ Egy Perl szkript, mely e diplomaterv keretében készült el. Parancssori opcióit és az `\index` parancs szintaxisát tekintve – néhány dokumentált különbségtől eltekintve – kompatíbilis a makeindex-szel. A szavakat a magyar ábécé szerint sorolja be, és magyar kezdőbetűkből képez csoportokat (pl. *Sz* és *Ö, Ő*, lásd még e dolgozat tárgymutatóját).

Xǐndy ♦ Egy LISP-ben készült, teljes mértékig testreszabható indexrendező program. `.xdy` kiterjesztésű konfigurációs fájljaiban reguláris kifejezéseken alapuló cserékkel lehet a rendezés előtt elvégzendő transzformációkat specifikálni, továbbá a kezdőbetű megállapítása és az oldalszámok rendezésének módja is szabadon beállítható. Ez a rugalmasság lehetővé teszi a magyar rendezési szabályok legtöbbjének definiálását, de például az, hogy a `\index{LaTeX@\LaTeX!\~{} telepítése}` bejegyzésben a „~telepítése” ábécébeli értéke `LaTeX telepítése` legyen, nem oldható meg. A Xǐndy hátránya továbbá, hogy nem illeszkedik teljes mértékben a LaTeX-hez, és az `\index` parancs argumentumában található `\ @ ! | "` karaktereket a makeindex-től eltérően (kevésbé általánosan) értelmezi. A Xǐndy jóval lassabb, mint a makeindex, de a husort.pl-nél is lassabb. Tovább bonyolítja a helyzetet, hogy a Xǐndy-t a szokásosnál nehezebb lefordítani, és csak régi függvénykönyvtárakkal és az újabb C fordítók alapbeállítással rossz kódot generálnak.

magyarp.xdy ♦ A Xǐndy tárgymutató-feldolgozó program magyar stílusfájlja. A `make¬index4 -s lang/magyar/magyarp` hívással vehető igénybe, ha mind a MagyarLaTeX, mind az ugyanarról a honlapról letöltött Xǐndy Debian csomag telepítve van. Mivel e stílusfájl nincs teljes mértékben tesztelve, vagyis nincs ellenőrizve, hogy mindenben megfelel a magyar rendezési szabályoknak, ezért éles használata kerülendő; érdemes inkább a husort.pl-t futtatni.

A husort.pl tervezésekor e sorok írójának először arról kellett döntenie, hogy melyik programozási nyelvet válassza. Mivel külső segédprogramról van szó, amely a terminálra írt tájékoztató üzeneteken kívül csak szövegfájlokat ír és olvas, ezért számos nyelv szóba jöhetett. E sorok írójának az volt a célja, hogy a husort.pl telepítése és futtatása a felhasználónak minél kevesebb munkát jelentsen, ezért a kört szűkítette a több platformon is elérhető, közismert nyelvekre. A lefordított nyelvek előnye a termék nagy futási sebessége, az interpretált nyelvek előnye pedig, hogy nincs szükség fordításra (így tehát a fordítóprogram telepítésére sem), és a képzetlen felhasználók számára is érthető, platformfüggetlen telepítési utasítás készíthető. E sorok írója azért választotta a Perl nyelvet, mert interpretált, és több tucat operációs rendszeren elérhető, ezek közül a legtöbb UNIX-ra amúgy is telepítve van. Perl-ben kis odafigyeléssel lehetséges platformfüggetlen kódot írni (a husort.pl is ilyen), továbbá az is megoldható, hogy csak a Perl beépített függvényeit használjuk, így a Windowsos futtatáshoz elég legyen a `perl.exe` és a megfelelő `perl*.dll` (tömörített összméretük 400 kB alatt van). A Perlbe integrált reguláris kifejezés támogatás nagyban hozzájárult ahhoz, hogy a husort.pl kódja tömör, könnyen

áttekinthető legyen. (A Perl, ellentétben például a Java-val, nem kényszeríti a programozót könnyen olvasható, strukturált forráskód készítésére, ezért sokszor nehezebb egy más által írt Perl szkriptet megérteni, mint a programot nulláról akár Perlben, akár egy másik nyelven újraírni. E sorok írója a husort.pl kódját igyekezett fegyelmezetten, jól olvashatóra megírni.)

3.4.1. A husort.pl által követett rendezési szabályok

A tárgymutató tételeinek betűrendbe (ábécé-rendbe) sorolási szabályainak felállításakor a husort.pl szerzője [5, 161–169. oldal]-re és [3, 22. oldal]-re támaszkodott. Az alábbi szabálylistában „[OK]" jelzi a husort.pl által követett szabályokat, a szabályoktól való eltérést pedig [· · ·] fejti ki.

S0a. „szó"-nak nevezünk egy előre rögzített kódolású (Latin-2 vagy TEX-T1), tetszőleges (0. . . 255) karaktereket tartalmazó, tetszőleges hosszú karaktersorozatot.

S0b. Két szó azonos, ha egyenlő hosszúak, és azonos pozíción azonos karakterek állnak. Jelölés: egyik = egyik. [OK]

S0c. Ha két szó nem azonos, akkor egyértelműen szabályozva van, hogy melyik kerül előrébb az ábécében. Jelölés: egyik < másik. [OK]

S1a. A magyar nagyábécé: a á b c cs d dz dzs e é f g gy h i í j k l ly m n ny o ó ö ő p q r s sz t ty u ú ü ű v w x y z zs [OK, ha angol szavak is kerülnek a tárgymutatóba, például *typesetting*, akkor a többjegyű mássalhangzókat `\empty`-vel kell megszakítani, például `\index{t\empty ypesetting}`]

S2a. A kis- és nagybetűk besorolási értéke azonos. Pl. kín < Kína < kínál < Kovács < kovács. [OK]

S2b. Ha a két szó csak a kis- és nagybetűkben különbözik, akkor a nagybetűk előrébb kerülnek. Pl. Kovács < kovács. [OK]

S3a. Ha a szavak azonos betűvel kezdődnek, a sorrendet az első nem azonos betű határozza meg. Pl. ha < habár < habilitáció < habitus [OK]

S3b. Ha az egyik szó az másikkal kezdődik, akkor a másik kerül előre, pl. ha < habár [OK].

S4a. A magánhangzók rövid és hosszú változatai azonos értékűek (de pl. u < ü): a = á = ä < e = é < i = í < o = ó = ô < ö = ő < u = ú < ü = ű. Azaz például á-val kezdődő szó megelőzhet a-val kezdődőt. Pl. ács < acsarkodik. [OK, ha a `no_vowel_equiv` opciót választjuk]

S4b. A nyelvészeti munkákban, valamint a hosszabb terjedelmű lexikonokban a magánhangzók rövid és hosszú változatai különböző értékűek (pl. u < ú < ü < ű), de csak akkor, ha az ékezetektől eltekintve a két szó megegyezik. [OK, ha a `vowel_equiv` opciót választjuk]

S4c. Ugyanez vonatkozik az egyéb ékezetekre (mellékjelekre) is (ha például idegen szavakat veszünk fel a mutatóba). Pl. áhitat < ähnlich < ahogy. [OK. TEX-T1 esetén (ez az alapértelmezés) csak a kódolásban megtalálható betűkre, melyek az a. . . z betűk ékezetes változatai. Latin-2 esetén csak a magyar ékezetes betűk és *ä*.]

S5a. Az egyjegyű és kétjegyű (illetve háromjegyű) mássalhangzókat (c–cs, d–dz–dzs, g–gy, l–ly, n–ny, s–sz, t–ty, z–zs) nem két betűnek, hanem egyetlen, külön betűnek tekintjük. Pl. C < Curie < Czuczor < Cs < Csajkovszkij < Csehov. [OK, az idegen szavakban előforduló többjegyű mássalhangzókat is egynek veszi]

S5b. Ha latin betűs ábécére rendezzük a tárgyszavakat, akkor a sorrend a következő: C < Csajkovszkij < Csehov < Curie < Czuczor. [nem így működik helyette ld. S5a, ami gyakoribb igény]

S6. A szavak belsejében vagy a szavak végén a kettőzött mássalhangzókat két külön betűnek tekintjük (pl. ggy = gy + gy). Pl. nagz < nagy < naggyá < nagygyakorlat < nagyjából. [OK]

S7. A régies írású magyar neveket, illetve az idegen neveket és szavakat nem kiejtés szerint, hanem úgy soroljuk betűrendbe, hogy a kétjegyű betűket felbontjuk (pl. ch=c+h, ew=e+w). A mellékjeles betűket sem kiejtés szerint csoportosítjuk. Pl. Čapek < Černý < Chianti < Cholnoki. [OK]

S8a. Az összetett szavakat és a külön- vagy kötőjellel írt szókapcsolatokat úgy soroljuk betűrendbe, mint az egyszerű szavakat. Pl. munkaerő < munkaerő-gazdálkodás < munkaerő nélküli < munkaerős. [OK]

S8b. Ugyanez vonatkozik a nevekre is. Pl. Széchenyi-hegy < Széchenyi István < Széchényi Könyvtár < Széchenyi tér. [OK]

S8c. A három tagból álló nevek is akkor kereshetők vissza a leggyorsabban, ha ugyanezt a logikát alkalmazzuk. Pl. Csáth Géza < Cs. Szabó László < Nagy A. László < Nagy Albert < Nagy A. Márton < Nagy Sándor < Nagy Szabó István < Nagy Zoltán. [OK]

S8d. Azoknál a neveknél azonban, amelyek írott formája eltér egymástól, már nem használható ez a szabály. Pl. Horvát Aladár < Horvát Károly < Horvát Zoltán < Horváth Imre < Horváth Péter [OK, ha a neveket elválasztó szóközöket a hívó külön megjelöli úgynevezett „fontos szóköz”-nek, azaz pl. „Horvát Zoltán” helyett „Horvát{ }Zoltán”-t és „Horváth Imre” helyett „Horváth{ }Imre”-t gépel]

S9. A besoroláskor minden adatot figyelembe kell venni (évszám, egyéb kiegészítés). Pl. Kovács Pál < Kovács Pál{ }(1767–1812) < Kovács Pál{ }(1946–1994) < Kovács Pál fordító < Kovács Pál költő < Kovács Pál (1767–1812) < Kovács Pál (1946–1994). [OK]

S10a. A mutatókban használt névelők *nem* részei a besorolási adatnak, éppen ezért névelőket csak akkor használjunk, ha az elkerülhetetlen (pl. címek). Pl. The Television < Television, The < televízió < televíziókészülék [nem ismeri fel a névelőket, LaTeX-ben külön pozícionálni kell, pl. `\index{Television@The Television}`]

S10b. Irodalmi művek mutatójában (pl. betűrendes versmutató) viszont a névelők is számítanak. Pl. Ádventi köd < Ady < A festő halála < Alkalmi vers. [OK, amúgy is ez az alapértelmezés]

S11–S21. [5, 116–118. oldal] részletesen taglalja, hogy személyneveket milyen formában kell felvenni a tárgymutatóba. [Ezek a szabályok igen fontosak, de a nevek sorrendezését nem érintik.]

S91. S4a., S8a. és S8b. közös általánosítása: A láthatatlan (szóköz, tabulátor stb.) karaktereket és az alábbi írásjeleket: `! " & ’ ( ) , - . /:;? [ ] _ { }`, kihagyjuk az összehasonlítás során, és a magánhangzók rövid és hosszú változatait egyenértékűnek tekintjük, kivéve ha két nem azonos szó ily módon egyenlőnek találtatna. Az első esetben a sorrend: fontos szóköz < ASCII vezérlő < itt fel nem sorolt ASCII írásjel < szókezdő számjegy < betű < egyéb számjegy. A kivételes esetben a sorrend: fontos szóköz = *whitespace* < ASCII vezérlő < összes ASCII írásjel < szókezdő számjegy < betű < egyéb számjegy. A kis- és nagybetűk azonos értékűek mindkét esetben, továbbá az egyes írásjelek között sem teszünk különbséget. Ha két nem azonos szó még a második esetben is egyenlőnek bizonyulna, akkor a két karaktersorozatot binárisan hasonlítjuk össze. A használt kódolások (ASCII, Latin-2 és TeX T1) közös tulajdonsága, hogy a bináris összehasonlításnál a nagybetűk megelőzik a kisbetűket. [OK]

S92. Az előző pont folyománya a husort.pl által követett Bab < bab < Báb < báb sorrend, mivel a kis- és nagybetűk különbsége már csak akkor számít, ha a kisbetűsített alakok megegyeznek. Egy másik sorrend lehetne: Bab < Báb < bab < báb. [OK]

S93. A számjegyek azonos értékűek, kivéve, ha emiatt két nem azonos szó egyenlőnek találtatna. Ezáltal a nemnegatív egész számok jól rendeződnek. Pl. $0 < 2 < 9 < 28 < 100$. A tizedestörtek és a negatív számok rendezése nem helyes. Pl. $2 < 9 < 2.7$, és $-2 < -10$. [OK]

3.4.2. Opciók

A husort.pl nem támogatja az `.ist` kiterjesztésű stílusfájlokat, hanem helyette a `-C` parancssori kapcsoló után lehet testreszabni a husort.pl működését. A parancssori kapcsolók megadhatók a TEX dokumentumban is (`\index{|run husort.pl ...}`, sőt `\index{|run makeindex ...}` és más is megadható, ekkor nem a husort.pl végzi el a rendezést és TEX forrás generálását. Ily módon a választott indexrendező program neve a `.tex` fájlba írható, a parancssorban elég mindig husort.pl-t hívni. A `-C` parancssori kapcsolót a következő argumentumban egy szónak kell követnie, melyben a kis- és nagybetűk különbsége nem számít, és csak az angol ábécé betűi vannak figyelembe véve. Ha az így kapott szó `no`-val kezdődik, akkor ki-, egyébként bekapcsolja a megfelelő opciót. Például `circum2` és `n_OCI/rC+uM-2` egymás negáltjai. Az egyes opciók (zárójelben alapértékeikkel) a következők:

group_headings ♦ (ki) Kezdőbetűjük szerint csoportosítja a tételeket, és minden kezdőbetűt félkövérrel megjelenít. A `-s gind` kapcsoló is bekapcsolja.

aggregate_pages ♦ (be) Az `\index{`*szó*`|`*parancs*`}` bejegyzéseknél azonos *szó* azonos *parancs*ához tartozó szomszédos oldalakat intervallummá fűzi össze, ha legalább hárman vannak, például: „3, 4, 6, 7, 8” → „3, 4, 6–8”. A `-r` kapcsoló kikapcsolja.

vowel_equiv ♦ (be) A magánhangzók megítélésekor az ékezet nem számít. Ha az opció ki van kapcsolva, akkor (az első összehasonlításnál) o = ó < ö = ő < ô, és az ezen betűkkel kezdődő csoport címkéje „O, Ó”. Bekapcsolt állapotban fent végig egyenlőség van, és „O” a címke (ez felel meg az ún. általános latin betűs ábécének; [12, §16]). A `-a` kapcsoló kikapcsolja. Az opció mindkét állása helyes a magyar nyelvben: bekapcsolt állapotában nyelvészeti munkák és hosszú lexikonok sorrendjét követi, kikapcsolt állapota pedig az egyéb művekét.

circum2 ♦ (be) Az inputenc csomag nélkül (vagy az inputenc és index csomagok mellett) a LATEX az ékezetes betűket `^^`-pal kezdve írja ki az `.idx` fájlba. A fájl beolvasásakor ez az opció visszaalakítja a betűket. Használata elengedhetetlen, mert csak így biztosítható, hogy a hosszú magánhangzók (első körben) a rövidekkel egyező besorolási értékkel bírjanak.

latin2 ♦ (be) Ha az index csomag be van töltve, akkor a bejegyzések az `.idx` fájlba az inputenc csomag szerinti kódolásban kerülnek be. Ha ez a kódolás `latin2`, akkor ezt az opciót kell megadni a husort.pl-nek, hogy a `^^`-os kombinációkat Latin-2 karakterekként értelmezze. Csak a `circum2` opcióval együtt van értelme.

separate_tags ♦ (ki) Kikapcsolt állapotában az oldalszámokat körülvevő *parancs* argumentumába a vesszővel és a `---`szal épített teljes intervallum kerül, egyébként pedig az intervallum két végpontja egy-egy *parancs* argumentumába kerül.

single_symbols ♦ (be) A nem számjeggyel vagy betűvel kezdődő szavakat egyetlen, „Jelek” nevű osztályba sorolja.

shadow_untagged ♦ (be) Ha ugyanaz a tétel ugyanazon az oldalon szerepel *parancs*csal és anélkül, akkor a parancs nélkülit figyelmen kívül hagyja. Akkor használható jól, ha például `\index{piros}` és `\index{piros|textbf}` is szerepel ugyanazon az oldalon, de mivel a félkövér a kettő közül a fontosabb, csak félkövér oldalszámot kívánunk látni a tárgymutatóban. A makeindex-nek nincs ilyen opciója.

3.4.3. Tapasztalatok

A husort.pl élesben való tesztelésére e dolgozat mellett a [24] írásakor került sor. Ennek a több, mint 750 oldalas könyvnek a tárgymutatója 35 oldalnál is hosszabb volt. A próbafuttatások során a husort.pl számos hibájára fény derült, és a makeindex pontos működését (beleértve a ritka, speciális vagy hibás index-bejegyzések feldolgozási módját is) ekkor ismerte meg e sorok írója. A hibák ki lettek javítva, a különbségek csökkentek, de számos különbség megmaradt (főleg a parancssori opciók értelmezését illetően), mert e sorok írója nem szerette volna a makeindex hibáit és gyengeségeit is utánozni. A különbségek ellenére a tárgymutatót mindkét program futásidejű hiba nélkül rendezte, de a makeindex természetesen nem a magyar szabályoknak megfelelően. A husort.pl érezhetően lassabb volt, de a sebességcsökkenés nem volt zavaró, és eltörpült a LaTeX fordítási folyamat egyéb parancsai mellett. Összességében tehát a husort.pl bevált, és a tapasztalatok alapján a husort.pl a képzetlen felhasználónak is ajánlható, mert teljes mértékben kiváltja a makeindex-et, és a magyar rendezési szabályokat követi, anélkül, hogy a LaTeX dokumentum forráskódját át kéne írni.

Íme a husort.pl és a makeindex közti különbségek részletes listája:

- Csak a husort.pl támogatja a `|␣run` módot.
- A makeindex nem veszi figyelembe az oldalszám elején a szóközt.
- A makeindex nem veszi figyelembe a `-q` kapcsolót, ha `-i` is adott.
- A makeindex kimenete speciális parancsok definiálását igényli a LaTeX dokumentumban, ha `-s␣gind` kapcsolóval hívtuk (és nem a husort.pl `|␣run` módján keresztül).
- Ha a makeindex-et paraméter nélkül hívjuk, akkor azt az `-i` kapcsolónak veszi, míg a husort.pl program súgót ír ki önmagáról a konzolra.
- A makeindex nem fogadja el ugyanannak a címszónak egyidejű, különböző stílusú nyitásait, például: `\index{a|(b} \index{a|(c}`.
- A husort.pl általában a szabványos kimenetre naplóz, és csak `-i` esetén tér át a szabványos hibakimenetre, a makeindex viszont mindent a szabványos hibakimenetre ír, továbbá a makeindex az `.ilg` fájlba is naplóz.
- A husort.pl nem engedi meg a parancssori kapcsolók összevonását, tehát csak a `husort.pl -q -i` helyes, míg `makeindex -qi` is helyes.
- A husort.pl automatikusan hozzáteszi az `.idx` kiterjesztést a parancssorban megadott fájlnévhez, a makeindex ellenben először a kiterjesztés nélküli fájlt próbálja megnyitni, csak utána teszi hozzá az `.idx`-et.
- A husort.pl elfogadja a `-sgind` kapcsolót is, de csak a `makeindex -s gind` a helyes.
- Csak a makeindex-nek adható meg több bemeneti `.idx` fájl a parancssorban.
- Csak a makeindex készít munkájáról naplófájlt (`.ilg` kiterjesztéssel).
- Csak a husort.pl fogadja el az `\index{a|b|textbf}` bejegyzést (és írja ki a `\textbf{42}`-t sorszámnak, továbbá figyelmeztető üzenetet jelenít meg, hogy a bal oldai `|` jelet le kellett volna védeni: `"|`). A makeindex visszautasítja a bejegyzést.
- A husort.pl nem ír ki figyelmeztetést (*Conflicting entries: multiple encaps...*), ha ugyanazon az oldalon `\index{a|textbf}` és `\index{a|textit}` is szerepel.
- Csak a husort.pl-nek adható olyan opció, amivel az és esetén az előbbit figyelmen kívül hagyja.
- Csak a makeindex támogatja a `-l`, `-c`, `-g`, `-t` és `-p` parancssori kapcsolókat.
- A husort.pl-nek van `-C` kapcsolója, ezzel szemben a makeindex `.ist` kiterjesztésű stílusfájlokat használ.
- A makeindex nem fogadja el az `\index{a|b@c}`, bejegyzést, a husort.pl pedig figyelmeztető üzenettel elfogadja, és az oldalszámot körülölelő parancs `\b@c` lesz.

3.5. Irodalomjegyzék a magyar tipográfia szerint

Egy bibliográfiai adatbázis-kezelő program használata a következő előnyökkel jár: a bibliográfiai tételek adatait tartalmazó adatbázis megosztása több dokumentum között; csak a ténylegesen használt irodalom kigyűjtése; a tételek (bejegyzések) automatikus ábécérendbe rakása; a tipográfiai szabályok automatikusan helyes alkalmazása. A LaTeX-hez kapcsolódó szokásos bibliográfiai adatbázis-kezelő, a BibTeX ún. stílusfájlokkal tetszőlegesen testre szabható. Az első stílusfájl, ami figyelembe veszi a magyar hagyományokat, e diplomaterv keretében készült el, neve: huplain.bst.

A BibTeX maga igen régi: legfrissebb változatát (0.99c) 1988-ban írták, és azóta számos őt kiváltó megoldás született, melyek a többnyelvű bibliográfiák generálását is támogatják (pl. MLBibTeX). E sorok írója mégis BibTeX-et választotta, mivel jól dokumentált, egyszerű, része minden TeX disztribúciónak, leírása megtalálható a legtöbb, LaTeX-hel foglalkozó könyvben (például [23]-ban és [24]-ben is), és a legtöbb szerző igényeit kielégíti. Legfőbb hátránya, hogy a többnyelvű irodalomjegyzékek kezelése nehézkes, különösen akkor, ha a dokumentum nyelve nem egyezik meg a hivatkozott mű nyelvével.

A BibTeX adatbázis egy `.bib` kiterjesztésű szövegfájl, amely bibliográfiai tételekből (bejegyzésekből) áll. Bejegyzésekre számos példa található a MagyarLaTeX részét képező magyar-minta.bib fájlban. Minden tétel (*entry*) mezők (*field*) listája. Minden mezőnek van egy (szabványos) neve, és egy értéke. E dolgozatot leíró tétel például így fest:

```
@MASTERSTHESIS{pts_diploma,
  type={Diplomaterv szakdolgozat},
  author = {Szab{\'o}, P{\'e}ter}, huname=1,
  year = 2004, month = may,
  title = {Magyar nyelv{\H u} m{\H u}szaki--tudom{\'a}nyos tipogr{\'a}fia},
  address = "Budapest",
  school={Budapesti M{\H u}szaki {\'e}s Gazdas{\'a}gtudom{\'a}nyi Egyetem},
  url = {http://www.math.bme.hu/~pts/pts_diploma.pdf}, }
```

3.5.1. Célkitűzések, előzetes kutatás

A BibTeX fenti előnyei miatt különösen alkalmas LaTeX-hel készülő művek irodalomjegyzékének szedésére. Első lépésként e sorok írója összegyűjtötte, hogy magyar nyelvű művek bibliográfiájában mely szabályokat kell betartani, és megvizsgálta, hogy erre a BibTeX alkalmas-e. A szabályok [5, 121–149. oldalak]-ban megtalálhatók. [21, 311. oldal]-tól is találhatók útmutatások, de részben azért, mert ezek [5]-nél jóval régebbiek, részben pedig a diplomaterv időkerete miatt nem lettek figyelembe véve. Mivel az útmutatások egyike sem algoritmikus jellegű, ezért egyes esetekben [5] igen gazdag irodalomjegyzékének bejegyzései, mint példák lettek figyelembe véve. [5] a bibliográfiai leírások számos típusát mutatja be, e sorok írója ezek közül a csak egyetlennek, a műszaki–tudományos cikkekben előforduló leggyakoribbnak, a hivatkozott műveknek a szerzőjük szerint növekvő sorrendbe rendezett, számozott, egyszerűsített bibliográfiai leírásának generálását tűzte ki célul maga elé. Az a reménye, hogy a megfelelő angolszász stílusfájlból (plain.bst) kiindulva néhány apró módosítással a magyar tipográfiának megfelelő stílus hozható létre, részben beigazolódott: az „apró” módosítások hatására az eredeti fájl mintegy $2{,}5$-szeresére duzzadt.

A magyar és az angolszász bibliográfia az alábbi fő pontokban térnek el (az elkészült huplain.bst és a plain.bst összevetése alapján):

– A mezősorrend eltérő (pl. a kiadás évének és helyének sorrendje).

– Mást és máshogyan kell kiemelni, zárójelbe tenni stb., különösen szakfolyóiratok évfolyamának és számának szedésekor (az amsplain.bst például félkövérrel szedi az évfolyamot).
– A dátumot a magyar helyesírás szerint, magyar hónapnevekkel kell megjeleníteni.
– Az egyéb generált szövegek, pl. *szerk.*, *és mások* és *fejezet* is más, mint az angolban.
– A szerzők felsorolása eltér, magyarban például „Bujdosó Gyöngyi – Wettl Ferenc" szedendő, lásd még a 2.1. alszakaszban a gondolatjelről szóló pontot.
– A tételeket elválasztó vessző, kettőspont és pont használata eltér.
– A magyar bibliográfiában a pontot követő szóköz nem szélesebb, mint az egyéb szóközök (`\frenchspacing`).
– A szerzők kereszt- és vezetéknevének, továbbá a név egyéb elemeinek feltüntetési sorrendje eltérő. Gondolni kell arra is, hogy ugyanabban az irodalomjegyzékben szerepelhetnek magyar és angol névsorrendű szerzők is.
– Korlátozás nélkül támogatni kell a magyar ékezetes betűket, és lehetőleg a T1 fontkódolás egyéb betűit is. (A BibTeX alapértelmezésben az ékezetek kapcsos zárójeles beviteli előírásának mellékhatásaként megakadályozza a ligatúrák létrejöttét az ékezetes betűk körül.)
– A rendezési szabályoknak figyelembe kell venniük az ékezeteket.
– A mezők rendezéshez felhasznált sorrendje eltérő, például máshogy sorolandó be egy mű, ha a szerző ismeretlen ([5, 138. oldal]).
– A mű címének ábécérendbe sorolásánál a magyar *a*, *az*, *egy* névelőket sem szabad figyelembe venni.

A magyarítás során fontos szempont volt, hogy ugyanazt az adatbázist lehessen használni a huplain.bst-vel magyar nyelvű, plain.bst-vel pedig angol nyelvű dokumentumokban. Ez csak úgy valósulhat meg, ha a szerző ennek tudatában készíti el a bibliográfiai adatbázist. Az egyszerűség kedvéért a huplain.bst minden bejegyzést a dokumentum nyelvéhez igazít, függetlenül a bejegyzés nyelvétől, tehát egy könyv szerkesztőjét mindig „(szerk.)" jelöli, és sosem pl. „ , editor", még akkor sem, ha a szerkesztő és a könyv is angol. A másik megoldást, nevezetesen, hogy minden tétel az ő saját nyelvén kerül szedésre, sokkal fáradságosabb lett volna megvalósítani a BibTeX eszköztárával. A huplain.bst tulajdonsága, hogy az ábécérendbe rakást az ékezetek és kettős mássalhangzók figyelembe vétele nélkül végzi.

A BibTeX rendezési és generálási modellje teljesen nyelvfüggetlen, bár eleve az angol nyelvhez tervezték. Azt kellett megvizsgálni, hogy ékezetes betűk miként vihetők be, továbbá jól rendezi-e a magyar ékezetes betűket és a többjegyű mássalhangzókat tartalmazó szavakat. Mivel az irodalomjegyzékben a tételek rendezése nem olyan kritikus, mint a tárgymutatóban, ezért nem követünk el nagy hibát, ha az ún. általános latin betűs ábécénél ([12, §16]) is egy egyszerűbb rendezési módszert alkalmazunk: a többjegyű mássalhangzók több különálló betűnek számítanak, az ékezetek pedig figyelmen kívül hagyandók, kivéve ha két szó csak az ékezetben különbözik. Ez a sorrend biztosítja, hogy két *Bota* közé ne ékelődhessen be egy *Bóta*, de az egymás mellé kerülő *Bota*, *Bóta* és *Bötá* szavak sorrendjét nem határozza meg. Nem fenyeget az a veszély sem, hogy Siklós megelőzi Sárossyt, mivel az *i* az *a* után következik az angol ábécében. Helyesírásunk ennél pontosabb követését az adott programnyelven igen fáradságos lett volna implementálni.

Cél volt az is, hogy a huplain.bst akkor is jó kimenetet generáljon, ha a magyar.ldf nincs betöltve.

A BibTeX8 a BibTeX nyolcbites (tehát ékezetes betűket kezelő) változata. Nem része a MagyarLaTeX-nek, de a honlapról letölthető egy Debian csomag (forrással), ami az

eredeti BibTeX8 2004-ben javított, kevesebb beégetett memóriakorlátot tartalmazó és a Kpathsea könyvtárat használó változatát tartalmazza. Az eredeti a CTAN-ről is letölthető, a `biblio/bibtex/8-bit` könyvtárból. A BibTeX8 nyolcbitessége abban nyilvánul meg, hogy minden karakterkódra definiálva van az ő nagybetűs és kisbetűs változata, továbbá a karakterek ábécébeli sorrendje. Az általános latin betűs ábécé szerinti betűrendet a 88591lat.csf fájl írja le, mely egyúttal a program alapbeállítása, ezért a `.bbl` fájl egyszerűen így generálható: `bibtex8` *fájlnév.aux*. A többjegyű betűket csak elemeikre bontva tudja rendezni, és a csak magánhangzó-hosszúságban különböző párokat (pl. Toth és Tóth) az idézésükével azonos sorrendbe állítja. Mindent összevetve tehát a BibTeX8 a nagybetűsítést és a rendezést könnyíti meg; de a nagybetűsítés magyar (és a legtöbb nyugat-európai) szövegre már a sima BibTeX-ben is jól működik, a huplain.bst-ből hívott `rmacc4` függvénynek köszönhetően (lásd a 3.5.2. alalszakaszban), a rendezést pedig az előbbi és az `rmbrace.extrat1` függvény segíti, ezért itt sincs szükség a BibTeX8-ra. A BibTeX8 hátránya, hogy nem része a TeX-disztribúcióknak, és mivel a Kpathsea könyvtárat sem használja, ezért nem találja meg a saját fájljait a `texmf` fában. A fenti érvek alapján e sorok írója szándékosan mellőzte a BibTeX8-at, és a magyar bibliográfiai stílusban egyedül a klasszikus, 1988 óta változatlan BibTeX lehetőségeit használta.

3.5.2. Implementáció

A BibTeX stílusfájljainak saját programozási nyelve van, amely egy *postfix* aritmetikát használó, verem alapú nyelv. Hasonló a PostScript-hez, de jóval egyszerűbb: leírása (a CTAN-en a `biblio/bibtex/distribs/doc/btxhak.tex`) összesen 9 oldalas, és leírja mind a 37 beépített függvényt. A programozó saját függvényeket definiálhat, melyekben használhat globális string és egész szám típusú változókat, elérheti a verem két legfelső elemét, meghívhat beépített és már definiált függvényeket. A stílusfájl feladata kettős: egyrészt előállítja minden egyes tételnek az ábécérendbe sorolásához használandó string típusú kulcsát, másrészt kiszedi a tételeket (LaTeX forráskódot generálva). A 37 beépített függvény igencsak speciális célú: például van olyan függvény (az `add.period$`), ami pontot rak egy string végére, ha még nem lenne, és van olyan is, ami egy személynevet egy minta alapján megformáz (a `format.name$`; a minta adja meg a keresztnév, a vezetéknév és az egyéb elemek sorrendjét, és azt is, mit kell rövidíteni), és olyan is, ami megmondja, hogy egy string a `cmr10` betűtípussal kiszedve hány pont szélességű (századokra kerekítve), de nincs olyan függvény, ami egy string hosszát (a karakterek számát) adja vissza, és olyan sincs, ami egy logikai értéket negál. Akik már programoztak PostScript-ben, tudják, mekkora kényelmetlenséget okoz, hogy nincsenek infix operátorok, és ha ehhez a BibTeX stílusfájl nyelvének egyéb nehezítéseit is hozzávesszük (a stringek nem módosíthatók, nincs a hosszt visszaadó függvény, a veremnek csak a két felső eleme érhető el), akkor nem meglepő, hogy egy egyszerű karaktercserélő rutin, ami egy stringben az összes `a` betűt `b`-re cseréli, forráskódja eléri a 600 bájtot, és megírásához – a nyelv alapos ismeretét feltételezve és többéves programozói rutint feltételezve – fél óra is szükséges lehet. A stílusfájl-nyelvből ráadásul hiányzik a PostScript-ben meglevő névtérkezelés (*dictionary stack*), a hatékony veremkezelés, a gazdag operátorkészlet és a változtatható (*mutable*) változók kezelése is. Még egy furcsaság, hogy a BibTeX ugyan bármilyen karaktert elfogad az adatbázisban, és egy string első karakterének a numerikus kódját mindig vissza tudja adni, ennek ellenére nem képes tetszőleges karakterkódból stringet építeni, csak az ASCII tartományból. Szerencsére ez utóbbira nem is volt szükség, mert a kiszedendő `.bbl` fájlba minden ékezetes betű a hosszú LaTeX-es alakjával került generálásra, például az *ű* betű `\H{u}`-ként.

A fentiek miatt a BibTeX stílusfájljának kódolása gyakran az assembly-programozáshoz hasonlít. Mivel a stílusfájl megírása egy egyszeri feladat, ezért az ügyetlen programozási nyelv okozta hátrány még nem kerekedett felül a BibTeX előnyeivel szemben (lásd az alfejezet elején).

Az implementáció során a stringkonverziót ciklusok jelentették a legnehezebb feladatot. A megvalósítás mintájául szolgált a plain.bst-ben található `n.dashify` függvény, amely minden kötőjelet gondolatjelre cserél egy stringben. Az új, stringkonverziót végző függvények a következők: az `rmacc4` függvény az adatbázisból származó, ékezetes betűket is tartalmazó stringet konvertálja a BibTeX ékezet-formátumára, például minden á karaktert a `{\'a}` stringre cserél. Támogatja a Latin-1 és Latin-2 bemeneti kódolásokat. Az `rmbrace.accs` függvény az ékezetes és egyéb betűk BibTeX-es alakjából előállítja a LaTeX-nek szedésre továbbküldött alakot a kapcsos zárójelek eltüntetésével, például az `{\"o}` részstringet `\"o`-re, az `{\ae}` részstringet pedig `\ae␣`-re cseréli, hogy ne akadályozzák az egalizálást. Az `rmbrace.extrat1` függvény pedig egy stringből – rendezés céljából – eltávolítja néhány betű forrásából a kapcsos zárójelet és a backslash-t, például a `{\DH}` részstringet `DH`-ra cseréli. Csak a T1 fontkódolásnak a 6. táblázatban felsorolt ékezet nélküli jelei közül azokkal foglalkozik, melyeket a `purify$` beépített függvény nem egyszerűsít, például a `{\ae}` részstring `ae`-vé alakítását nem végzi el, mert ezt a `purify$` amúgy is megteszi.

Az elkészült huplain.bst a [24] könyv és e dolgozat bibliográfiájának kiszedésével lett tesztelve.

3.5.3. Tételtípusok és mezők

A huplain.bst a magyar tipográfia szerint szedi ki az összes, a plain.bst által definiált tételtípust (*entry type*). A plain.bst tételtípusai és a bennük megengedett mezők listája [23, 269. oldal]-on megtalálhatók. A huplain.bst számos bővítést tartalmaz ehhez képest, ezeket az alábbi lista álló betűvel tartalmazza. A tételtípusok és mezőik tehát:

article ♦ Egy folyóiratban (journal) vagy magazinban megjelent cikk. Szükséges mezők: *author*, *title*, *journal*, *year*. Kiegészítő mezők: *volume*, *number*, *pages*, *month*, *note*, numpages, issn, url, tetex-url, ctan-url.

book ♦ Megnevezett kiadó (publisher) által kiadott könyv. Szükséges mezők: *author* vagy *editor*, *title*, *publisher*, *year*. Kiegészítő mezők: author2, *volume* vagy *number*, numvolumes, *series*, *address* (városnév), *edition*, *month*, *note*, numpages, isbn, url, tetex-url, ctan-url.

inbook ♦ Egy kiadó által kiadott könyv egy fejezete vagy egyéb része. Szükséges mezők: *author* vagy *editor*, *title*, *chapter* és/vagy *pages*, *publisher*, *year*. Kiegészítő mezők: *type*, author2, *volume* vagy *number*, numvolumes, *series*, *address* (városnév), *edition*, *month*, *note*, numpages, isbn, url, tetex-url, ctan-url.

incollection ♦ Egy kiadó által kiadott gyűjtemenyes mű egy része. A gyűjteményes művek jellemzője, hogy van szerkesztőjük (aki az egész műért felel), és minden résznek megvan a maga szerzője. Szükséges mezők: *author*, *title*, *booktitle*, *publisher*, *year*. Kiegészítő mezők: *editor*, author2, *volume* vagy *number*, numvolumes, *series*, *type*, *chapter*, *address* (városnév), *edition*, *month*, *note*, numpages, isbn, url, tetex-url, ctan-url. Ha a rész szerzője és a mű szerkesztője megegyezik, ez `editor={self}` megadásával jelölendő (csak huplain.bst-vel).

booklet ♦ Nyomtatott és kötött kiadvány; kiadó és támogató intézmény nélkül. Szükséges mező: *title*. Kiegészítő mezők: author2, *volume* vagy *number*, numvolumes, *series*, *address* (városnév), *edition*, *year*, *month*, *note*, numpages, isbn, url, tetex-url, ctan-url.

6. táblázat. A BibTEX ékezetei és különleges betűi

ì	{\‘i}	í	{\’i}	î	{\^i}	ĩ	{\~\i}
ī	{\=\i}	i̇	{\.i}	ï	{\"i}	į	{\c i}
ǐ	{\u\i}	ǐ	{\v\i}	ő	{\H\i}	ị	{\d i}
i͡	{\b i}	i͡n	{\t\i\i}	o͡o	{\t oo}	i̊	{\r\i}
œ	{\oe}	Œ	{\OE}	æ	{\ae}	Æ	{\AE}
å	{\aa}	Å	{\AA}				
ø	{\o}	Ø	{\O}	ł	{\l}	Ł	{\L}
ı	{\i}	j	{\j}	¡	!‘	¿	?‘
þ	{\th}	Þ	{\TH}	i̧	{\k i}		
ð	{\dh}	Đ	{\DH}	đ	{\dj}	Đ	{\DJ}
ŋ	{\ng}	Ŋ	{\NG}	ß	{\ss}	SS	{\SS}

manual ♦ Technikai dokumentáció vagy szabvány, melynek nincs kiadója, és általában szerző sincs feltüntetve. Szükséges mező: *title*. Kiegészítő mezők: author2, *author*, *organization* (pl. a szabvány kódneve), *address*, *edition*, *month*, *year*, *note*, numpages, *url*, *tetex-url*, *ctan-url*.

unpublished ♦ Egy szerzővel és címmel bíró dokumentum, melyet még formálisan nem publikáltak. Szükséges mezők: *author*, *title*, *note*, numpages. Kiegészítő mezők: *month*, *year*, url, tetex-url, ctan-url.

mastersthesis ♦ Doktori vagy egyéb tudományos cím megszerzéséhez kapcsolódó értekezés, vagy diplomamunka, kiemelés (`\emph`) nélküli címmel. Szükséges mezők: *author*, *title*, *school*, *year*. Kiegészítő mezők: author2, *type* (alapértelmezett értéke „Doktori értekezés”, felülbírálható: pl. `type={Kandid{\’a}tusi {\’e}rtekez{\’e}s}`, *address* (lehetőleg városnév legyen, és ne a *school*-on belüli intézménynév), *note*, numpages, url, tetex-url, ctan-url.

phdthesis ♦ PhD vagy egyéb tudományos cím megszerzéséhez kapcsolódó értekezés. A *type* mező alapértéke „PhD értekezés”, egyébként megegyezik a a *mastersthesis*-szel.

techreport ♦ Egy oktatási vagy egyéb intézmény által kiadott jelentés, általában sorszámozott. Szükséges mezők: *author*, *title*, *institution*, *year*. Kiegészítő mezők: author2, *type*, *number*, *address*, *month*, *note*, numpages, url, tetex-url, ctan-url.

proceedings ♦ Egy konferencia teljes írásos anyaga, az előadásokhoz tartozó cikkek gyűjteménye. Szükséges mezők: *title*, *year*. Kiegészítő mezők: *editor*, *volume* vagy *number*, *series*, *address*, *month*, *organization*, *publisher*, isbn, issn, url, tetex-url, ctan-url, numpages, *note*.

inproceedings (= conference) ♦ Egy konferencián elhangzott előadáshoz tartozó közlemény. Szükséges mezők: *author*, *booktitle*, *title*, *year*. Kiegészítő mezők: *pages*, *editor*, *volume* vagy *number*, *series*, *address*, *month*, *organization*, *publisher*, isbn, issn, numpages, *note*, url, tetex-url, ctan-url, talked.

misc ♦ Ha a többi tételtípus egyike sem illik a dokumentumra. Nincs szükséges mező. Kiegészítő mezők: *author*, *title*, *howpublished*, *month*, *year*, *note*, numpages, isbn, issn, url, tetex-url, ctan-url.

A huplain.bst által definiált kiegészítő mezők:

inputenc ♦ Az *inputenc* mező lehetővé teszi, hogy a bejegyzés egykarakteres ékezetes betűket tartalmazzon. A mező lehetséges értékei: `latin1` és `latin2`, magyar tételekhez ez utóbbi ajánlott. A mező nélkül bevitt ékezetes karakterek a dokumentum bemeneti

kódolása szerint értelmeződnek, és a rendezésben sem vesznek részt, ezért nem ajánlottak. A *inputenc* csak a huplain.bst-ben működik, ezért az őt használó adatbázis nem hordozható. Az ékezetes betűk begépelésére a BibTeX az alábbi szabványos formátumot ajánlja: `{\'e}` és `{\H o}` (lásd még a 6. táblázatban) A kapcsos zárójeleket a BibTeX felismeri, így helyesen rendez és kisbetűsít ékezetes betűket is. A formátum kiszedve megtörné a TeX ligatúrákat, ezért a huplain.bst a kellő időben eltávolítja az ékezetes betűket körülvevő kapcsos zárójeleket. Ha az ékezetes betűket a táblázat szerint viszszük be, akkor nincs szükség az *inputenc* mezőre, és a LaTeX fordítás is jóval gyorsabb lesz.

Ha az ékezetes betűket több karakterrel, a 6. táblázatban megadott módon visszük be, akkor nincs szükség az *inputenc* mezőre. Az egyetlen karakterből álló ékezetes betűkhöz pedig kapcsos zárójelre nincs szükség.

Érdemes összevetnünk a fenti táblázatot azzal, ahogy LaTeX-ben lehet ékezetes betűket építeni. A legfontosabb különbség, hogy a kapcsos zárójeleket máshová kell helyezni: LaTeX-ben például az *é* betű ajánlott alakja `\'e`, az *ű* betűjé pedig `\H{u}`. A LaTeX-es alakok nem akadályozzák meg a ligatúrák létrejöttét (pl. *TéT* helyett *TéT*, és az automatikus egalizálást. A BibTeX-es szintaxis oka az, hogy a BibTeX a `{\`*parancs*`...}` konstrukciót egyetlen betűnek kezeli. Ennek hatása cikkcímek kisbetűsítésekor (`"t" change.case$`), keresztnevek kezdőbetűjének kinyerésekor (`"{ff.}" format.name$`) és az ékezetek sorbarendezés előtti eltörlésekor (`purify$`) is megmutatkozik. A BibTeX írója nem gondolt a ligatúrákra, amikor ezt a szintaxist kitalálta, ezért a huplain.bst minden mező szedése előtt meghívja a stílusfájlban definiált `rmbrace.acc` függvényt, amely leszedi a fölösleges kapcsos zárójeleket. A a 6. táblázatban szereplő `{\th}`, `{\TH}`, `{\dh}`, `{\DH}`, `{\dj}`, `{\DJ}`, `{\ng}`, `{\NG}` és `{\SS}` betűket csak a huplain.bst a két alkotó betű alapján sorolja ábécérendbe.

nocheck ♦ Ha értéke nem üres (ajánlott: 1), akkor kevesebb figyelmeztető üzenetet ír ki a BibTeX az adott tételről, például nem figyelmeztet arra, hogy *article* esetén nem adtuk meg *number*-t.

preseries ♦ Ha értéke nem üres (ajánlott: 1), akkor a könyv sorozatcíme (*series*) a cím (*title*) elé kerül.

huname ♦ A szerzők és a szerkesztők neveinek megjelenítését szabályozza. Ha nem adtuk meg, vagy értéke 0, akkor az angol névsorrendet követi (pl. „Jules Verne”); ha 1, akkor a magyart (pl. „Verne Gyula”); ha 2, akkor pedig a fordított angolt, vesszővel (pl. „Verne, Jules”); ha pedig 3, akkor a magyart, de rövidített keresztnévvel (pl. „Verne Gy.” – az adatbázisban a kettős mássalhangzó miatt így szerepel: `author={Verne, {\empty Gy}ula}`). Ajánlott az 1 illetve 2 érték megadása, attól függően, hogy magyar vagy külföldi az illető. E mező értékétől függetlenül az adatbázisban a szerző nevét angol sorrendben kell megadnunk, mint ahogy a példabejegyzésben is szerepelt, és ha nem sorolunk fel minden szerzőt, akkor ␣`and`␣`others`-szel záruljon a lista. Ezáltal az adatbázis magyar és angol nyelvű dokumentumokban változtatás nélkül felhasználható lesz.

Ha a rövidített keresztnév kettős mássalhangzóval kezdődik, akkor a kettős mássalhangzót így kell körülvenni: `{\empty...}`. A BibTeX a kapcsos zárójelekről tudja, hogy meddig tart a név első betűje. A huplain.bst átdefiniálja LaTeX `\empty` makróját, így kapcsos zárójelek nem törnek majd meg ligatúrát.

key ♦ A rendezéshez használatos kulcsot adja meg. Csak a huplain.bst sajátossága, hogy ha `key={.}` esetén a szerző (és szerkesztő nélküli művek figyelmeztetés nélkül a többi mű *elé* kerülnek címük (*title*) alapján, míg `key={.}` nélkül címük (*title*) szerint kerülnek besorolásra (eltérően a plain.bst-től, ahol a lista elejére kerülnek, ömlesztett

sorrendben), a szerzővel rendelkező művek *közé*. A figyelmeztetés letiltható *nocheck*=1-gyel.

crossref ♦ A BibTEX bejegyzések közti mezőörökítő funkcióját valósítja meg. Használata kerülendő, mert a huplain.bst nem a magyar tipográfiának megfelelően reagál rá.

talked ♦ Csak *inproceedings* esetén használható. A bejegyzésben „In" helyett „Elhangzott a… konferencián" fog szerepelni.

A fontosabb szabványos mezők magyar vonatkozású jellemzői:

year, month ♦ A dokumentum megjelenésének dátumát a *year* és *month* mezőben adhatjuk meg. A hónapot a 12 konstansból válasszuk (`jan`, `feb`, `mar`, `apr`, `may`, `jun`, `jul`, `aug`, `sep`, `oct`, `nov` vagy `dec`). Ha a hónap napja is ismert, akkor azt angol tipográfiával, a hónap neve után fűzzük, például `month = "15~" # mar`. Ne tegyünk e mezők végére pontot. Az évszámot és a többi számértékű mezőt nem kötelező idézőjelbe vagy kapcsos zárójelek közé tenni. Ne törődjünk azzal, hogy a *month*-t angolul kell megadni: a kimenetbe magyar hónap és nap kerül, a megfelelő sorrendben.

series ♦ Itt adjuk meg a sorozat címét, ha a könyv egy több kötetből álló vagy számozott sorozatból van, de mi ebből csak egyetlen kötetre hivatkozunk. Ne vegyük fel a „sorozat" szót (pl. `series = {A kultúra világa}`), ezt a huplain.bst magától hozzáfűzi. Ekkor a *series* mező tartalma a *title* mögé kerül. Ha ehelyett elé szeretnénk írni, akkor vegyük fel a *preseries*=1 értéket (csak a huplain.bst-ben). Ajánlott a *series* mellett a *volume* vagy *number* mezők egyikét szerepeltetni, mellyel megadjuk, hogy a könyv a sorozat hányadik tagja.

Ha a teljes sorozatra hivatkozunk, és nemcsak egy tagjára, akkor a *series*, *volume* és *number* mezőket ne szerepeltessük, helyettük a sorozatcímet a *title* mezőben, a sorozat köteteinek számát pedig a *numvolumes* mezőben adjuk meg.

type ♦ *mastersthesis* és *phdthesis* esetén az értekezés típusa, például `type = {kandid{¬\'a}tusi {\'e}rtekez{\'e}s}`.

inbook és *incollection* esetén a rész típusát adja meg. A *type* mező alapértéke „fejezet" (*chapter*). Az ettől eltérő értékeket angolul érdemes megadni: (*part* → rész, *volume* → kötet, *chapter* → fejezet, *section* → szakasz, *subsection* → alszakasz, *subsubsection* → alalszakasz, *paragraph* → paragrafus, *subparagraph* → alparagrafus, *figure* → ábra, *table* → táblázat), hogy ugyanaz az adatbázis angol nyelvű stílusokkal is használható legyen. A huplain.bst a fent felsorolt angol kulcsszavakat magyarra fordítja, az ettől eltérő stringeket pedig változatlanul hagyja.

chapter ♦ *inbook* és *incollection* esetén a rész címét adja meg. Ha a rész többszintű, e sorok írója a mélyülő sorrendet javasolja, vesszővel elválasztva, például `chapter =␣{Veszélyforrások, Vírusok, Makróvírusok}`.

3.5.4. A huplain.bst nyelvfüggetlen újdonságai

A plain.bst-ből hiányoznak az alábbi nyelvfüggetlen mezők:

ISBN ♦ Könyv ISBN száma, esetleg végződhet X-re. Például: `ISBN={964 379 227 4}`.

ISSN ♦ Folyóirat vagy könyvsorozat ISSN száma.

URL ♦ A tételhez tartozó webcím. Ajánlott a dokumentum preambulumában az url csomag betöltése, mert ellenkező esetben a huplain.bst a `\verb` parancsot használja, ami nem tudja eltörni a sort, ezért a hosszú URL-t a következő sorba rakja (és az előző sort balra igazítja); ez jobbra igazított sorokat is eredményezhet, ha ugyanabban a bekezdésben két ilyen speciális sortörés is szerepel. Például: `url = {http://www.¬dante.de/}`.

teTeX-URL ♦ A tételhez tartozó cím a teTeX disztribúció fájában. Általában egy DVI, esetleg egy PDF fájl neve, például `tetex-url={doc/makeindex/makeindex.dvi}`.

CTAN-URL ♦ A tételhez tartozó cím a CTAN TeX-archív fájában. Általában egy TeX forrás, egy DVI, esetleg egy PDF fájl neve, például `ctan-url = {indexing/make¬index/doc/makeindex.dvi}`.

numpages ♦ A dokumentum terjedelme, utolsó számozott oldalának (vagy az őt követő, még a fejezet részét képező utolsó oldalnak) a sorszáma. Nem tévesztendő össze a *pages* mezővel, amely kijelöli, hogy a *numpages* hosszúságú műből mely oldalakra hivatkozunk (pl. `pages=23` vagy `pages={23--45}`).

numvolumes ♦ Egy könyv köteteinek száma. Nem tévesztendő össze a *volume* mezővel, amely kijelöli, hogy a *numvolumes* darab kötetből álló könyv mely kötetére hivatkozunk (pl. `volume=2`).

author2 ♦ A mű elkészítésében közreműködő egyéb személyek, például `author2 = {¬Illusztrálta Kis Géza. A borítót tervezte Gipsz Jakab}`.

A tételek kinézetén és generált szövegrészein (csak huplain.bst esetében) változtathatunk a `\bibOverride` parancs definiálásával, melyben egyéb `\bib...` makrókat módosíthatunk. A `\bib...` makrók a következők: `\bibAnd`, `\bibEtAl`, `\bibEd`, `\bibNewBlock`, `\bibVolume`, `\bibTechRep`, `\bibInSelf`, `\bibUrll`.

Például a `@preamble{ "\def\bibOverride{\def\bibEtAl#1{ et~al.}} "}` bejegyzés hatására „et al.” fog megjelenni „és mások” helyett, ha az *author* mező ␣`and`␣ `others`-re végződik. Ajánlott a `@preamble` string-jében minél gyakrabban szóközt hagyni, mert ha nincs szóköz, akkor kb. a 76. karakternél a BibTeX mindenképpen eltöri a sort, még akkor is, ha a sortörés egy parancsszót vág ketté (!).

Lehetőség van a nevek megjelenítésének szabályozására. Ehhez a LaTeX dokumentumban egy `\nocite{f1}` hivatkozást kell szerepeltetni, az adatbázisban pedig egy ilyen bejegyzést elhelyezni: `@FMTS{f1, firstname.fmt={f.}}`. Ekkor a keresztnevek rövidítve jelennek majd meg. A vezetéknevek megjelenítése is szabályozható a `lastname.fmt` mező megadásával, de ezt nem szokás rövidíteni. Például az `author={John von Neu¬mann and Donald E. Knuth and Verne, {\empty Gy}ula}` eredménye *huname*=1 mellett: „von Neumann J. – Knuth D. E. – Verne Gy és mások”, *huname*=0 mellett pedig: „J. von Neumann, D. E. Knuth, Gy. Verne és mások”.

3.6. Helyesírás-ellenőrzés

A helyesírás-ellenőrzés nem része a LaTeX fordítási folyamatnak, ezért a szerző bármilyen programot használhat, amely sima szövegfájlokat képes ellenőrizni. Általában ajánlott a helyesírás-ellenőrzőt egy szövegszerkesztőn belülről meghívni, mert ekkor a hibás szavakat a szövegkörnyezet alapján gyorsan és nagy biztonsággal lehet javítani. A Wordben alkalmazott megoldás, hogy a hibás szavakat pirossal aláhúzza, igen felhasználóbarát és akadálytalanul alkalmazható LaTeX dokumentumok esetén. (Aki rendelkezik a Word egy jogtiszta példányával kipróbálhatja, hogy a `.tex` forrásfájlt sima szövegfájlként betölti Wordbe, lefuttatja a helyesírás-ellenőrzést, majd elmenti a változtatásokat szövegfájlként. Érdemes egy próbamentéssel rögtön a fájl betöltése után ellenőrizni, hogy a Word nem rontja-e el a soremeléseket, és nem végez-e más automatikus módosítást a dokumentumon.)

UNIX alatt a szabványos helyesírás-ellenőrző az ISpell. Ennek létezik interaktív és csővezetés módja, az utóbbihoz `-a` kapcsolóval kell futtatni. Interaktív módban a terminálon a dokumentum néhány soros részlete jelenik meg, az aktuálisan hibásnak talált szó pedig ki van emelve. A képernyő nagyobb részét a felkínált javítási lehetőségek foglalják el.

Az interaktív mód nem ajánlott (kivéve igen rövid, néhány kilobájtos dokumentumok ellenőrzésére), mert a félbemaradt ellenőrzést csak a fájl elejéről lehet újrakezdeni, továbbá az ISpell nem biztosítja a dokumentumban történő szabad navigálás lehetőségét, pedig egy-egy szó javításhoz gyakran szükség lehet a dokumentum távoli részeinek megtekintésére. Az ISpell TEX-beolvasója a `-t` kapcsolóval aktiválható, ez (LA)TEX parancsokat és néhány egyéb szintaktikai elemet nem vesz figyelembe, de sajnos a megjegyzéseken belül is ellenőriz. Nem ismeri teljes mértékben a LATEX szintaxist, például ellenőrzi a `longtable` környezet argumentumát, ami nem szöveges információ, hanem egy táblázat oszlopformátumát előíró karaktersorozat. Ha `.tex` kiterjesztésű fájlt ellenőrzünk, ez a mód magától aktiválódik.

A csővezetékes mód még ennél is barátságtalanabb, de ezt nem emberi felhasználóknak, hanem szövegszerkesztőkkel való kommunikációra tervezték. A protokoll igen egyszerű: beírunk egy sort, majd az ISpell válaszul kiír néhány sort, amelyben az általunk beírt sorban levő szavak helyességéről nyilatkozik egy-egy sorban. Több olyan, a magyar nyelvet ismerő helyesírás-ellenőrző létezik, amely beszéli ezt a protokollt:

MSpell ♦ A *Morphologic* cég Helyes-e? programjának lebutított, 2002 óta a cég honlapjáról ingyenesen letölthető Linux-os változata. (A kereskedelmi forgalomban kapható Helyes-e? jóval nagyobb tudású, és a szótára is frissebb.) Ugyanez a helyesírás-ellenőrző található meg az Office 97-ben és változtatás nélkül az Office XP-ben. A forráskód, a szótár és a fejlesztői dokumentáció nem nyilvános. Képes az ISpell helyébe lépni, de akkor is csővezeték módban fut, ha `-a` kapcsoló nélkül indítjuk. Előnye, hogy igen gyors, és a szótárfájl is kicsi (1 MB alatt van). Hátránya – a zártság mellett –, hogy nem bővíthető: a felhasználó által felvett szavakat nem tudja ragozni. [15, 72–80. oldalak] egy igen részletesen összehasonlítja az MSpell és a HunSpell képességeit, és bár nem von le végkövetkeztetést, az összehasonlításban rendre a szabad és nyílt HunSpell bizonyul jobbnak. ([15]-öt a HunSpell szerzője írta, ezért pusztán ez alapján az a két program nem hasonlítható össze objektívan.) Példaszavakat is közöl, melyek begépelésével bárki reprodukálhatja a hibás működést (a fordított esetet, vagyis amikor a program jól működik, természetesen nem lehet példaszavakkal *bizonyítani,* ekkor csak a helyes működés tényét említi meg). Az összehasonlítás rövidítve [16, 3. oldal]-on is megtalálható.

MagyarISpell ♦ Az ISpell UNIX-os helyesírás-ellenőrző program magyar modulja. Szabad szoftver, letölthető a `http://magyarispell.sf.net/` címről. 1998-ban kezdte feljeszteni Németh László Felhasználói és fejlesztői dokumentációját [15] tartalmazza. Debian rendszerekre az `apt-get install ihungarian` paranccsal telepíthető. Nem változtat az ISpell igen egyszerű, az angol nyelvhez tervezett ellenőrzési modelljén, ezért szótárfájlja igen nagy. (A modell egyik fontos korlátja, hogy bár külön kezeli a szótőt és a toldalékot, nem képes egymás után több toldalékot hozzátenni a szóhoz, így például ahhoz, hogy a *legjobbakat* szót helyesnek fogadja el, a *jobbakat* ragozott alakot is fel kell venni a szótárba. Ez a fő oka annak, hogy nagy a szótár. Megfigyelhetjük, hogy a *legjobbakat* szótöve az ISpell *jobbakat*, a HunSpell szerint viszont, helyesen, a *jó*. A szótövezéshez a HunSpell-t a `-s` kapcsolóval kell meghívni, míg a MagyarISpell-nek elég a `-a`.) A HunSpell-lel együtt folyamatosan fejlesztés alatt áll. Telepítését és használatát lásd [15]-ben. A MagyarISpell interaktív használatához az `ispell -d␣hungarian` vagy `ispell -d magyar` parancsot kell futtatni, a csővezetékhez pedig ugyanezt `-a` kapcsolóval.

HunSpell ♦ A MySpell többplatformos helyesírás-ellenőrző keretrendszer magyarított változata kiegészítve egy magyar szótárfájllal. A MagyarISpell utódja. szabad szoft-

verSzabad szoftver, letölthető a `http://magyarispell.sf.net/` címről. 2002-ben kezdte fejleszteni Németh László Felhasználói és fejlesztői dokumentációját [15] tartalmazza. Ha nem kívánjunk interaktív módban használni, akkor letöltés (`hunspell-*.tar.gz`, ahol a csillag helyére a verziószám helyettesítendő, de `-lib` nem!) és kicsomagolás után így telepíthető: a `hunspell.cxx` fájlban kommentezzük ki a `#define CURSES` sort, a `Makefile`-ból pedig töröljük a `-lcurses` és `-lreadline` szavak összes előfordulását, futtassuk a `make hunspell` parancsot, majd másoljuk a keletkező `hunspell` futtatható fájlt a `/usr/local/bin`, a `hu_HU.aff` és `hu_HU.dic` fájlokat a `/usr/share/myspell` könyvtárba. Bár a MySpell ellenőrzési modellje fejlettebb az ISpell-nél, a HunSpell még a MySpell-hez képest is tartalmaz javításokat, például minden hibás szóra megpróbál maximum 4 ékezetet felrakni, továbbá futás közben is bővíthető (ragozott alakokkal is!) a szótár. A módosítások teljes listája megtalálható [15, 23. oldal]-on. Csővezetékes, az ISpell-lel kompatíbilis használatához a `hunspell␣-d hu_HU -a` parancsot kell futtatni. Tartalmaz egy szótövezőt is (HunStem), amely `-s` kapcsolóval aktiválható (és a helyes szavakról fokozatosan lebontja a toldalékot, például a *javíthatatlanságaitoktól* végéről először a ragokat és jeleket vágja le, így *javíthatatlanság* marad, majd a képzők egyenként eltűnnek, és az utolsó kiírt sorban már csak *javít* szerepel). TEX módja a megjegyzések tartalmát (szerencsére) nem ellenőrzi. A MySpell igen lassan indul, mert az adatbázisát két, összesen 6 MB-os szövegfájlból olvassa be, és felépít egy kb. 20 MB-os, gyors keresésre alkalmas adatszerkezetet a memóriában.

A MagyarISpell szerzője jelenleg a Szószablya fejlesztésen ([16]) dolgozik, melynek célja, hogy a magyar weboldalak szövegtartalma alapján magyar szógyakoriság-szótár, majd ennek felhasználásval nyílt forráskódú morfológiai elemző program készüljön. A morfológiai elemző annyival tud többet a szótövezőnél, hogy nemcsak a tövet, hanem a hozzá járuló toldalékokat (és azok jelentését) is megmondja.

Álljon itt a fenti programok csővezetékes használatára néhány példa. A `helyedírás` „szóra” MagyarISpell esetén `& helyedírás 3 0: helyed írás, helyed-írás, helyesírás`, MSpell esetén `& helyedírás 1 0: helyesírás`, HunSpell esetén pedig `&␣helyedírás 2 0: helyesírás, helyed írás` a válasz. Az `&` jel azt jelenti, hogy a szó helytelen. Az utána következő szám pedig a felkínált javítási lehetőségek száma (ezután egy nulla és egy kettőspont jön, majd a javítási lehetőségek következnek). Számos egyéb választípus lehetséges, a teljes protokollt az ISpell leírása és a UNIX kézikönyvlapja (ispell(1)) is tartalmazza. Megemlítjük még, hogy a helytelen `észszerű` „szót” a MagyarISpell összetett szóként elfogadja (válasza: `-`), az MSpell elfogadja (válasza: `*`), és csak az HunSpell ajánl fel javítást, melyek közül a helyes *ésszerű* a második.

Létezik egy harmadik mód is, nevezzük döntési módnak, amely a csővezetékes módhoz hasonló, de csak azt dönti el, hogy egy szó helyes-e vagy sem. Gyorsabb a csővezetékes módnál, és a `-l` kapcsolóval aktiválható.

3.6.1. Integrálás Emacs-ekbe

A szerző szempontjából az a legkényelmesebb, ha a kedvenc szövegszerkesztőjéből futtatja a helyesírás-ellenőrzőt. Ez megvalósítható, ha a szövegszerkesztő csővezetéken keresztül hívja az ISpell-t vagy az ISpell protokollját beszélő fenti programokat. Az Emacs, XEmacs, Vim, OpenOffice, LyX programok és a KDE szövegszerkesztői mind képesek erre. Mivel sima szövegszerkesztők használatosak TEX dokumentumok írására, ezért az OpenOffice-t és a LyX-et azonnal kizárhatjuk. A Vim-nek csak a legújabb, 6.3-as verziójába van beépítve az ellenőrzés, a KDE telepítése túl nagy fáradság pusztán a helyesírás-ellenőrzés

kedvéért (továbbá a KDE igen erős gépet igényel), ezért UNIX alatt az Emacs és XEmacs programokat érdemes először kipróbálni.

Az alábbi, felhasználóknak szóló leírás azért került bele a dolgozatba, mert e sorok írójának kellett kikísérleteznie, mivel nem talált olyan útmutatót, amely mindhárom magyar helyesírás-ellenőrző Emacs-es használatát lépésről lépésre leírja, továbbá az alapbeállításokkal az Emacs-ek képtelenek magyar nyelvű ellenőrzésre. Mindhárom ellenőrző leírása tartalmaz néhány szót az Emacs-ről, de az ott leírtak általában hiányosak, az útmutatások egy része nem is működik a leírt módon, továbbá hiányzik belőlük egy, a mindennapi munka során kényelmes és megbízható módszer ismertetése.

E két Emacs feltételezi, hogy a helyesírás-ellenőrzőt az `ispell` paranccsal lehet futtatni; az MSpell és HunSpell programokat nem ismerik (mint ahogy a többi UNIX-os szövegszerkesztő sem). Elvileg ki lehetne választani a magyar nyelvet a program menüjéből, de ez az alapkiépítésben szállított Emacs-ekben nincs bekonfigurálva, ezért nem érdemes vele vesződni: sose válasszuk az ellenőrzőt az Emacs menüjéből, ehelyett az Emacs indítási előtt állítsuk be a `DICTIONARY` környezeti változót.

E sorok írója elkészítette UNIX-ra az ispell.pl Perl szkriptet, melyből a 3.6. alszakasz elején bemutatott mindhárom ellenőrző futtatható csővezetékes módban. A MagyarISpell választásához a `-d magyar` (vagy `-d hungarian`), az MSpellhez a `-d mspell`, a HunSpellhez pedig a `-d hunspell` kapcsolóval kell futtatni az ispell.pl-t. (Mindhárom esetben szükség van a `-a` kapcsolóra.) A `-d` kapcsoló helyett a megfelelő nyelvet a `DICTIONARY` környezeti változóban is megadhatjuk. Ha se a `-d` kapcsolót, se a környezeti változót nem adjuk meg (vagy `magyar-best` nyelvet írunk elő), akkor először a HunSpell-t, majd az MSpell-t, végül a MagyarISpell-t próbálja az ispell.pl. Ha a programot `--list` kapcsolóval hívjuk, feltérképezi a gépre telepített ellenőrzőket, `--install`-lal hívva pedig elhelyezi önmagát `/usr/local/bin/ispell` néven (és létrehoz néhány `.hash` fájlt, melyek jelenlétét egyes szövegszerkesztők ellenőrzik, például az Emacs menüjének is szüksége van e fájlokra).

Az ispell.pl további előnye, hogy használatához nem kell (rendszergazdaként) telepíteni se az MSpell-t, se a HunSpell-t. Elég, ha az `mspell` és `hunspell` parancsok elérhetők a `$PATH`-on. Az MSpell-nek szüksége van az `mhyph.lex` és `mspell.lex`, a HunSpell-nek pedig a `hu_HU.aff` és `hu_HU.dic` szótárfájlokra – ezek legyenek ugyanabban a könyvtárban, mint a megfelelő parancs.

A két Emacs-ben történő helyesírás-ellenőrzéshez először telepítsük a az ellenőrzőket; majd az ispell.pl-t másoljuk ISpell néven a `$PATH`-ra; állítsuk be a `DICTIONARY` környezeti változót; az Emacs-et vagy az XEmacs-et abból az ablakból indítsuk, melyben a változóbeállítást elvégeztük; adjuk ki a `set-language-environment` parancsot `Latin-2` paraméterrel; nyissuk meg a `.tex` forrásfájlt; ugorjunk a fájl végére (`end-of-buffer` parancs), jelöljük ki a régió végét (Ctrl-Szóköz leütése vagy `set-mark-command` parancs), vigyük a kurzort oda, ahonnan az ellenőrzést el akarjuk kezdeni, és adjuk ki az `ispell-region` parancsot. Ha hibát tapasztalunk (például *... misalignment...* jelenik meg az alsó sorban), futtassuk újra az `ispell-region` parancsot. Az ellenőrzés az ISpell interaktív változatához lesz hasonló: minden hibára a billentyűzeten kell reagálnunk. Ctrl-G leütésével bármikor megszakíthatjuk az ellenőrzést, és az `ispell-region` kiadásával folytathatjuk (onnan, ahol éppen a kurzor áll). Az igazán kényelmes megoldás, hogy a szövegszerkesztő szerkesztés közben pirossal húzza alá a hibás szavakat, sajnos nem működik megbízhatóan (a `flyspell-mode` parancs kiadása után csak néhány szót jelöl meg, és később már semmit sem húz alá, esetleg eleve el sem kezdi a javítást, vagy csak akkor javít, ha a kurzort lassan mozgatjuk; [15, 16. oldal] se ajánlja) – ez az ispell.el és a flyspell.el Emacs-modulok hibája, melyekkel számos egyéb probléma is van (pl. néha

kétszer kell kiadni a parancsot, hogy elinduljon az ellenőrzés, vagy újra kell indítani az Emacs-et, továbbá igen lassúak) – e sorok írója nem is érti, hogyan kerülhettek be ilyen kezdetleges állapotban, tesztelés nélkül az Emacs 20-as és 21-es verziójába, továbbá az XEmacs-be. A piros aláhúzást viszont megkaphatjuk, ha kijelölünk egy szövegrészt, és kiadjuk a `flyspell-region` parancsot. A szövegrész módosítása után újra ki kell adni e parancsot. Az egész dokumentum ellenőrzését a `flyspell-region` paranccsal lehet kezdeményezni, de ez kivárhatatlanul lassú (lásd a 7).

Az Emacs és az XEmacs az ellenőrzőt a `-B` opcióval futtatja, vagyis arra utasítja, hogy az ismeretlen összetett szavakat hibásnak ítélje. Ezt megakadályozhatjuk, ha az első `ispell-*` parancs kiadása előtt kiadjuk az `ispell-kill-ispell` parancsot (hogy az ispell.el betöltődjön), majd az `eval-expression` parancson belül kiértékeljük az alábbi LISP kifejezést: `(setcar (cdr (cdr (cdr (cdr (cdr (car ispell-dictionary-alist)))))) "")`.

3.6.2. A Epsilon-ból indítható ellenőrzés egyedi vonásai

E sorok írója, mivel látta, hogy az Emacs-ben igen nehézkes a helyesírás-ellenőrzés, és az alap Debian csomagokban nem található egyéb szövegszerkesztő, amely pirossal aláhúzná a hibás szavakat, házi használatra az Epsilon nevű többplatformos, programozható szövegszerkesztőt kiegészítette egy, a TeX dokumentumok ellenőrzésére használatható, az `ispell -a` protokollját használó, a hibás szavakat a szövegben megjelölő rutinnal. E kiegészítés kényelmessé tette e dolgozat ellenőrzését, de még sokat kéne fejleszteni rajta, hogy egy képzetlen felhasználó is telepíteni és használni tudja. (További nehézség, hogy az Epsilon szövegszerkesztő fizetős, az időkorlát nélküli változatnak többszáz dollár az ára, és ráadásul magán az Epsilon-on is sokat kell konfigurálni, hogy könnyen használható legyen, és az „általános elvárásoknak” megfelelően működjön.)

Az elkészült kiegészítő három fontos szempontból különbözik a hasonló megoldásoktól: egyrészt saját maga gyűjti ki a TeX forrásfájlból a szavakat (ezáltal a dupla többjegyű mássalhangzókat bevezető aktív karaktert figyelmen kívül tudja hagyni – erre a többi ismertetett szövegszerkesztő nem képes), és így az `ispell -t` hívásnál pontosabban hagyja figyelmen kívül a kommenteket és bizonyos TeX és LaTeX programozási konstrukciókat, másrészt pedig az ellenőrzendő szavakat előbb rendezi, és minden szót csak egyszer ad át a helyesírás-ellenőrzőnek, továbbá megjegyzi az ellenőrző által hozott döntéseket, ezért az újraellenőrzés már sokkal gyorsabb. Kis kényelmetlenséget okoz, de elkerüli a szövegszerkesztés lelassulását, hogy a dokumentum módosítása közben nem ellenőriz (ha a felhasználó egy már hibásnak talált szót ír be, azt hibásnak jelöli meg, de az ellenőrizetlen szavakat nem különbözteti meg a helyesektől). Az (újra)ellenőrzést a felhasználó a `pts-spellcheck` parancs kiadásával kezdeményezheti. A szavak előzetes rendezésének hátránya, hogy a tévesen különírt vagy kötőjellel írt szavakat nem jelzi (és nem javítja), mivel ezek eleve külön szóként jutnak el a helyesírás-ellenőrzőhöz.

A csővezetékes magyar ellenőrzők sebességét és memóriaigényét tartalmazza a 7. táblázat. A dokumentum e dolgozat egy köztes, kb. 210 kB méretű változata volt. A tesztgép Celeron 333 MHz-es processzorral és 128 MB SDRAM-mal bírt – ez már réginek számít, de mivel a LaTeX-es dokumentum-előkészítéshez amúgy sem szükséges új és gyors gép, a helyesírás-ellenőrzést is célszerű volt egy ilyen gépen kipróbálni. Meglepő nagyságrendi különbség van az Emacs-en kívüli és azon belüli ellenőrzés gyorsasága között, amiből azt a következtetést vonhatjuk le, hogy az Emacs ISpell-interfésze a lassú. Az Epsilon szövegszerkesztő a saját szkriptnyelvét körülbeli ugyanolyan gyorsan futtatja, mint az Emacs a LISP-ét, mégis az Epsilon-os ellenőrzés nagyságrendekkel gyorsabb, tehát az

Emacs-es lassú működést nem technológiai okokkal, hanem az interfész rossz tervezésével és/vagy megvalósításával lehet magyarázni. Az Epsilon-os megoldás azért gyorsabb a sima csővezetéknél is, mert ugyanazt a szót csak egyszer ellenőrzi.

7. táblázat. A helyesírás-ellenőrzés sebessége és memóriaigénye Celeron 333 MHz-es gépen, Linux alatt, a sebesség karakter per másodperc egységben

Program	Sebesség	Memóriaigény
MSpell* `-a`	5 637	14 MB
MagyarISpell `-t -a`	8 569	1.45 MB
HunSpell `-t -a`	4 265	21 MB
Emacs 20 HunSpell	322	
Emacs 21 HunSpell	769	
Epsilon HunSpell	16 728	
Vim HunSpell `-l`	59 024	

* Figyelmen kívül hagyja a `-t` opciót.

3.6.3. Helyesírás-ellenőrzés Vimben

A Vim honlapjáról letölthető vimspell.vim kiegészítő, amely a folyamatos és a paranccsal hívható ellenőrzést támogatja. (LA)TEX dokumentumok esetén is működik. A hibás szavakat aláhúzza, és külön paranccsal javítási lehetőségeket javasol. Legfőbb hátránya, hogy lassú, főleg akkor, ha a dokumentum sok hibás szót tartalmaz; ez a dokumentum betöltésekor, az ellenőrzés elindításakor és a folyamatos ellenőrzéskor is megmutatkozik. A folyamatos ellenőrzésre úgy működik, hogy ha a felhasználó *Insert* módban leüti a Szóköz billentyűt. A folyamatos ellenőrzéshez a 2004 áprilisi 1.84-es vagy annál frissebb vimspell.vim fájl szükséges, amely definiálja a `spell_insert_mode` változót. A Vim 6.1-es vagy frissebb verziója szükséges, mert ehhez a verzióhoz készült a szkript.

E sorok írója kidolgozott egy módszert, amellyel a vimspell.vim a gyakorlatban, a processzorok jelenlegi sebességét alapul véve, kb. 400 kB-os dokumentumok ellenőrzésére használható, vagyis nem kell túl sokat várni az ellenőrzésre. A keletkezett huspell.vim parancsfájl a MagyarLATEX része.

Az első fájl betöltésekor a vimspell.vim végigpróbálja a `spell_language_list` változóban felsorolt ellenőrzőket, ami több másodperces késleltetést okozhat. Ezen segít az ispell.pl, mert érzékeli, a Vim-es próbát, és ekkor gyorsan, az ellenőrzők tényleges meghívása nélkül tér vissza. Érdemes a példában leírt módon az ispell.pl-re bízni a gépre telepített ellenőrzők listázását, hogy ne jelenjen meg minden dokumentum megnyitásakor hibaüzenet a hiányzó nyelvről. A `spell_executable` változóban sajnos nem adható meg pl. `ispell.pl`, ezért az ispell.pl-t `ispell` néven kell a `$PATH`-on elérhetővé tenni.

A vimspell.vim azért olyan lassú sok hibás szó esetén, mert beépíti a hibás szavakat a Vim színezési (*syntax highlighting*) szabályaiba. A tesztgépen (paramétereit lásd a 78. oldalon) egy 1160 hibát tartalmazó dokumentumot HunSpell-lel 4 másodperc alatt sikerült ellenőrizni és a hibás szavakat a Vim ablakában megjelölni (ez jóval gyorsabb a 7. táblázatban látható többi programnál), de ugyanennek a dokumentumnak az ellenőrzése MSpellben már 58 másodpercig tartott, mert az MSpell 2100 hibát talált. Próbaképpen e sorok írója készített egy olyan „helyesírás-ellenőrzőt”, amely minden szót hibásnak jelez. Ez egy másodpercnél is hamarabb lefutott, és ezek után a Vim 16 percig gondolkozott a színezésen, majd *Segmentation fault*tal elszállt. A lassúságot a Vim szkriptnyelvében írt `while`-ciklus okozta, amely a megtalált hibás szavakat egyetlen reguláris kifejezéssé

alakította. E sorok írója kicserélte ezt a ciklust egy külső Perl hívásra, és 10 másodperc alatt lefutott az ellenőrzés. Sajnos azonban a sok hibát tartalmazó dokumentumok esetén hosszú reguláris kifejezés keletkezik, és ennek illesztésekor a Vim 6.1-es (pontosabban: 6.1.18-as) és 6.2-es verziója változatos hibaüzenetek közepette száll el. Ezért e sorok írója a reguláris kifejezéseket használó `syn match` parancsok egy részét lecserélte `syn keyword`-re, és ezáltal néhány másodperc alatt lefutott a minden fájlt hibásnak találó ellenőrzés (és minden szó pirossal aláhúzva jelent meg). De egy 1,4 MB méretű dokumentumra, melynek szólistája az ismétlések kiszűrésével 90 kB, már ez sem működött: a `syntax` parancs ugyan jelezte, hogy mik a hibás szavak, de a Vim nem húzta őket alá pirossal. Az a következtetés vonható le, hogy a Vim architektúrája és a vimspell.vim nem alkalmas hosszabb szövegek helyesírás-ellenőrzésére; de a vimspell.vim-ben elvégzett megfelelő javításokkal és alkalmas beállításokkal egy kb. 100 oldalas TEX forrásfájl még viszonylag kényelmesen ellenőrizhető. E sorok írójának véleménye szerint a Vim következő verzióiban elsősorban a reguláris kifejezések illesztését kéne stabillá tenni, továbbá a `system` függvény helyett párhuzamosan futtatott folyamatokkal való kommunikációt kéne biztosítani, hogy ne legyen szükség minden változtatás esetén új helyesírás-ellenőrző folyamat indítására.

4. Összefoglalás

4.1. Értékelés

E sorok írója nem bánta meg, hogy a LATEX magyarítását választotta. Ez egyrészt jó lehetőség volt számára arra, hogy a LATEX programozásában elmélyedjen, másrészt a tervezés során elkészülő magyar.ldf, husort.pl, huplain.bst és egyéb szoftverek nemcsak a LATEX-hel dolgozó szerzők és szedők munkáját könnyítik meg, hanem – remélhetőleg – növeli a LATEX-et kipróbálók körében azok táborát is, akik végül a LATEX-hel készítik el egy új művüket. E sorok írója korábbi tapasztalatai alapján meg van győződve arról, hogy műszaki–tudományos művek lehető legjobb minőségű szedésére a TEX alapú szedőprogramok a legalkalmasabbak. Ezt a meggyőződését erősíti, hogy matematikai és természettudományos könyvek és folyóiratok többsége LATEX-ben készül. (Mérnöki kiadványok esetén, melyek kevesebb matematikai képletet tartalmaznak, a LATEX nem áll előkelő helyen.) A LATEX magyarítása hozzájárul az előny fenntartásához.

4.1.1. magyar.ldf

A LATEX-közösség számára a magyar.ldf 1.4-es verziója 1998 óta elérhető. Ez a verzió már tartalmazta a határozott névelő generálásának egy kezdetleges implementációját, a generált feliratok (pl. „Tartalomjegyzék”) magyarítva voltak, tudott egyféleképpen dátumot generálni, a táblázatok, ábrák és strukturális egységek címei helyesen jelentek meg (bár ez utóbbiban a számot záró pont utáni köz túl nagy volt), bizonyos feltételek mellett magyarul jelent meg a tételszerű környezetek számozása, az élőfej is magyarítva lett (noha döntött betűvel lett szedve, és néha nem aktiválódott a magyarítás), és a dupla többjegyű mássalhangzók szedését aktív karakter segítette (de letiltotta a szóelválasztást a szó többi részében). Ezek a szolgáltatások (vö. a 3. táblázattal) még korántsem voltak elegendőek magyar nyelvű szöveg elfogadható minőségű szedéséhez, továbbá rengeteg bosszúságot jelentett a régi verzió összeakadása más csomagokkal, ezért sokan a magyar.ldf nélkülözésére kényszerültek.

Az e diplomaterv keretében elkészült verzió befejezi az 1.4-es változat csak félig megvalósított szolgáltatásait, és számos újat is bevezet, továbbá gyakorlati szempontból kielé-

gítően orvosolja az előző verzió ismert kompatibilitási problémáit. (Az ismeretlen problémák tüneteit pedig a felhasználó kezelheti olyan betöltési opciók megadásával, melyek a magyar.ldf problémás részét kikapcsolják.) Tévedés viszont az, hogy az 1.5-ös verzióval a magyarítás lezárult. A 4.2. alszakasz tartalmaz egy vázlatos listát az ismert hibákról és a megvalósításra váró szolgáltatásokról. Az 1.5-ös verzió nyújtotta megoldásokra a LaTeX-közösségnek 1998 óra várnia kellett. Bár a munka nem fejeződött be, az új verzióval már a képzetlen felhasználó is jó minőségű, a magyar hagyományok legfontosabb szabályainak megfelelő dokumentumot hozhat létre, és a magyar.ldf jobban automatizálja a dokumentumok professzionális szedését is.

A magyar.ldf-hez e sorok írója magyar.dtx néven, a LaTeX-csomagok dokumentációs módszertana szerint készített egy 40 oldalas angol nyelvű felhasználói leírást (ugyanez a fájl tartalmaz egy 65 oldalas forráskód-listát, és egy készülőben levő, jelenleg 15 oldalas fejlesztői leírást is). A magyar.dtx eljutott Johannes Braamshoz, a babel csomag karbantartójához, és a folyamatban levő egyeztetés után ez év nyarán várható, hogy az 1.5-ös verzió bekerül a hivatalos disztribúcióba.

A TeX-et Knuth eredetileg azért készítette, hogy szebb formában ki tudja adni a *Számítógép-programozás művészete* c. könyvének ([9]) második kiadását. Ez a könyv programozási (tehát műszaki) és matematikai (tehát tudományos) részeket vegyesen tartalmaz. A TeX-ben eredetileg nem volt több nyelv egyidejű támogatása, mivel Knuthnak nem volt erre szüksége a könyvében. Azóta két fontos fejlesztés született: a TeX már be tud tölteni nyolcbites betűtípusokat, vagyis egy font 256 karaktert tartalmazhat, tehát az ékezetes betűknek is van hely, továbbá a `\language` változó segítségével több elválasztási mintagyűjtemény között lehet váltogatni. (Egy további bővítés a `.tci` fájlok bevezetése: a TeX a terminálra és a fájlokba nyolcbites karaktereket is tudjon írni. Ez kevésbé jelentős, és egyes TeX disztribúciók még ma sem kapcsolják be alapértelmezésben. Az ε-TeX (ejtsd: ítekh), a TeX egy kiterjesztése támogatja ugyanazon a bekezdésen belül mindkét írásirányt: jobbra és arlab, a forrásban az utóbbi `\beginR balra\endR`-ként szerepel; ez pl. héber vagy arab nyelvek szedésekor hasznos.) Minden egyéb nyelvfüggő viselkedés TeX-makrókkal van megvalósítva: a LaTeX esetén ezeket a makrókat a babel csomag tartalmazza. A TeX egy lehetséges utódja, az Omega nyelvi szinten számos új lehetőséget biztosít, melyek jelenleg magyar nyelvű szöveg szedésére nem jobbak lényegesen a TeX lehetőségeinél (de egyes ázsiai és egyéb nyelvek esetén igen), kivétel talán, hogy a TeX (8-bites bemeneti rutinja miatt) nem képes UTF-16 kódolású Unicode szövegek beolvasására, míg az Omega igen.

A magyar.ldf tervezésének kezdeti szakaszában nem volt világos egy-egy bonyolult szolgáltatásról, hogy megvalósítható-e a TeX programozási lehetőségének és szedési modelljének felhasználásával. Bár a TeX makrónyelve Turing-teljes, mérnöki szempontból nem a legalkalmasabb általános célú programok írására, és már a tipográfiával összefüggő számos feladat lekódolása nehézségeket támaszt (lásd a következő bekezdésben). Mindezek ellenére e sorok írója mindent meg tudott valósítani a magyar.ldf-ben, amit eltervezett. A magyar.ldf elkészítése egy jó alkalom volt e sorok írójának, hogy a TeX programozási tudását bővítse és készségét fejlessze.

Az új verzió óriási fájlmérete nemcsak a sok új szolgáltatásról árulkodik, hanem arról is, hogy a TeX mennyire kevéssé hatékony: C vagy egyéb nyelvekben pofonegyszerű feladatokat TeX-ben csak többoldalas makróprogramozással lehet megoldani, és minden makróhoz egy hosszú felsorolást kell(ene) mellékelni, hogy mely bemenetekre nem működik pusztán azért, mert pl. a TeX kapcsos zárójeleket és szóközöket elnyelő mágikus tulajdonságait nem lehet kikapcsolni. A TeX makróprogramozásban létrejövő kód sok esetben kusza, hosszú, nehezen érthető és csak speciális esetben működik, a közös névtér

pedig csak növeli a csomagok összeakadásának esélyét. Ráadásul a LaTeX makrócsomagok számos hallgatólagos feltételezéssel élnek, továbbá gyakran előzetes megfontolások nélkül bírálják felül az alap LaTeX parancsokat, nem gondolva arra, hogy ezzel más csomagok működésében a fejlesztő számára nehezen diagnosztizálható zavart okoz. A fentiek mellett csoda, hogy a LaTeX egyáltalán működik. (A ConTeXt elvei ettől lényegesen eltérnek: csak egy szűk körű fejlesztői csapat írhat bele a központilag karbantartott kódba. Ez növeli a rendszer megbízhatóságát, de a nyíltság elvével ellentétes.) A magyar.ldf-nek nem kis erőfeszítésébe kerül, hogy bizonyos csomagok jelenlétét felismerve saját módosításait pont a megfelelő időben hajtsa végre, továbbá hogy az alap LaTeX parancsok közül minél kevesebbet definiáljon át, lehetőleg olyanokat, melyekhez más – ismert – csomagok nem nyúlnak. A magyar.ldf ebből a szempontból jól vizsgázott: a 750 oldalas [24] által betöltött 50 csomag egyikével sem akadt össze – de a jövőben megjelenő új csomagokról semmit nem állíthatunk. A fentieket figyelembe véve e sorok írója megérettnek látja az időt egy új, a hagyományos procedurális programozáshoz közelebb álló, mágikus tulajdonságok nélküli, névterekkel rendelkező makrónyelv kidolgozására. (Reális esélyt viszont csak akkor lát arra, hogy a TeX helyét egy másik program átveszi, ha az átmenetet zökkenőmentesen és visszafelé kompatíbilis módon meg lehet valósítani.) A jelenlegi LaTeX-et feltételezve a szerző reméli, hogy az Omega vagy az új babel a mostaninál fejlettebb és hatékonyabb keretrendszert biztosít majd a nyelvfüggő tipográfia megvalósítására – egy olyan keretet, amelyben az alkalmatlan programozási eszközökkel való küzdelem, és a kompatibilitás fáradságos és keresztbe foltozgató biztosítása helyett magára a tipográfiára lehet koncentrálni.

4.1.2. husort.pl

E sorok írója korábban egy kb. 500 karakter hosszú, előfeldolgozó Perl szkriptet használt a makeindex bemenetének átalakítására, hogy azt a makeindex már többé-kevésbé a magyar ábécé szerint rendezze. A figyelembe veendő rendezési szabályok összegyűjtése után világossá vált, hogy vannak olyan előírások, melyeket a makeindex-szel nem lehet betartani (például külön *S* és *Sz* csoport létrehozása), ezért e sorok írója egy új, a makeindex-et nem használó Perl szkript, a husort.pl fejlesztésébe kezdett. Az implementálás gond nélkül folyt. A Perl nyelv fejlett szövegmanipulációs lehetősége, különös tekintettel a reguláris kifejezések magas fokú integrálására, igencsak megkönnyítette a kódolást. A három alprogram (beolvasás: a TeX kimenetének beolvasása, a több karakteren bevitt ékezetes betűk egy karakterré alakítása; rendezés; generálás: a rendezett tételek LaTeX forrásként való kiírása) közül a beolvasást megírása volt a legnehezebb, mert a különböző LaTeX csomagok (inputenc, t1enc és index) más-más formátumú `.idx` fájlt eredményeznek, és a husort.pl-t úgy kellett elkészíteni, hogy minden variációban jól működjön. Gyakran előfordul, hogy a makeindex olyan `.ind` fájlt készít, melynek beillesztésekor a LaTeX szintaktikai hibát jelez. Ez nem a makeindex hibája, hanem azzal magyarázható, hogy a szerző eleve rossz `\index` bejegyzéseket hoz létre, melyben egyes speciális karakterek (`" ! @ | % \ { } $`) rosszul vannak levédve, és a hiba figyelmeztetés nélkül az `.ind` fájlba kerül. Ennek megelőzésére a husort.pl minden tárgymutatóbeli szó generálásakor ellenőrzi, hogy a kapcsos zárójelek, továbbá a dollárjel és a matematikai mód egyéb határolójelei helyesen vannak-e egymásba ágyazva, és figyelmeztető üzenet kíséretében kihagyja az `.ind` fájlból a hibásnak vélt sorokat. Az üzenet – a későbbi LaTeX hibaüzenethez képest – értelmes, lényegbe vágó, és pontos sorszám-információt tartalmaz.

A husort.pl teszteléséhez – az egyes funkciók ad-hoc tesztelése után – kezdetben nem állt rendelkezésre egy hosszú mű bonyolult tárgymutatójával, amelyen a husort.pl-t lefut-

tatva legalább azt ellenőrizhette volna, hogy tényleg szintaktikusan helyes kódot generál. Ez a probléma megoldódott, mert e sorok írójának birtokába került egy 1188 oldalas magyar nyelvű matematikai kézikönyv 75 oldalas tárgymutatójának forrása, továbbá e sorok írója bekapcsolódott egy 750 oldalas, a LaTeX-ről szóló kézikönyv ([24]) írásába. Ez utóbbi könyv tárgymutatója 35 oldal hosszúságú, és lefordítása során a husort.pl számos hibájára fény derült. A tárgymutató kezdetben a makeindex-szel sem fordult le, a dokumentum javítása során e sorok írójának fejében tisztázódtak azok a szabályok, különös tekintettel a speciális karakterek levédésére, melyeket a makeindex követ az `.idx` fájl beolvasásakor, és ezek alapján újra lett írva a husort.pl beolvasást végző alprogramja. Számos opció is azért került a husort.pl-be (lásd a 3.4.2. alalszakaszban), hogy a program a könyv írásakor felmerülő gyakorlati igényeket ki tudja elégíteni. E sorok írója egyetért Knuth véleményével (a husort.pl-re vonatkoztatva), ami szabad fordításban a következő: „Arra a következtetésre jutottam, hogy az az ideális, ha egy új rendszer tervezője nemcsak maga valósítja meg a tervet, de ő az első nagybani felhasználó, és ő írja az első felhasználói kézikönyvet. E négy tevékenység különválasztása nagy kárt okozott volna a TeX-nek. || Ha nem vettem volna részt teljes odaadással eme tevékenységekben, többszáz javítás és fejlesztés sosem történt volna meg, mert sosem gondoltam volna rájuk, vagy nem lettem volna tudatában a fontosságuknak."

A fejlesztés eredményeképp a makeindex-et ismerő felhasználók magyar nyelvű művük tárgymutatóját a husort.pl-lel egyszerűen, további tanulás és a magyar sajátosságokra fordított külön figyelem nélkül is elkészíthetik. Tipográfusok és haladó felhasználók számára a husort.pl opciókat és a kinézetre hatással levő, felüldefiniálható makrókat biztosít.

4.1.3. huplain.bst

Kevés szerző terheli magát azzal, hogy megjegyezze, pontosan mit kell kiemelni az egyes bibliográfiai tételtípusokban, a mezők milyen sorrendbe kerüljenek, továbbá hogy az egyes mezők után kettőspontot, pontot vagy szóközt kell-e rakni. A BibTeX stílusfájlja mindezt leírja, ezért a szerzőknek nagy könnyebbség, hogy BibTeX-hel készíthetik műveik irodalomjegyzékét. A jelentés és a megjelenítés szétválasztásán túl további előny a tételek automatikus beválogatása és rendezése: ha egy szerző felvesz egy művet a bibliográfiai adatbázisába, arra a műre bármikor hivatkozhat anélkül, hogy a részletes adatokkal foglalkoznia kéne. Sőt, a szerzők – bizonyos korlátok között – megoszthatják egymással az adatbázist, akkor is, ha különböző nyelvű műveket különböző bibliográfiai stílussal készítenek. Az adatbázisok webes megosztásának a `.bib` fájlformátum lett az egyik szabványos módja, sok műszaki–tudományos folyóirat honlapja és sok kereső adatbázisa tartalmaz BibTeX tételeket. (Bár az XML kiváló keretformátum egy bibliográfiai adatbázis létrehozásához, addig nem versenyezhet a BibTeX-hel, amíg az adatbázis tételeiből nem lehet egy TeX dokumentum irodalomjegyzékét automatikusan összeválogatni.)

A fenti előnyök természetesen csak akkor jelentkeznek, ha létezik a dokumentum nyelvének megfelelő bibliográfiai stílus. Ezt a régóta jelentkező hiányt szünteti meg a huplain.bst, amely úgy lett tervezve, hogy ugyanaz az adatbázis, bizonyos korlátozásokkal, használható legyen angol és magyar nyelvű művekhez. Ennek kulcseleme a *huname* mező, amelyet egyedül a huplain.bst vesz figyelembe, és a szerző névrészeinek sorrendjét befolyásolja. A nevet az angol sorrendben kell beírni (vagy vesszővel a magyar sorrendben), mert ezáltal más stílusok angulul, a huplain.bst pedig – a *huname*=1 értékadással – magyarul jeleníti meg. Az, hogy a huplain.bst nem ír elő egy új ajánlást az ékezetes betűk bevitelére, és ennek ellenére, a kapcsos zárójelek a megfelelő időben történő automatikus eltüntetésével eléri, hogy az ékezetes betűk körül is létrejöjjenek az egalizálás és

a ligatúrák, szintén fontos eleme az adatbázisok nyelvközi hordozhatóságának.

Nem nyílt lehetőség a huplain.bst tesztelésére és lektorálására nagy adatbázison. A [24]-hez használt adatbázis mindössze 50, e dolgozat adatbázisa pedig 24 tételt tartalmaz (e tételek szedése mind kifogástalan). A huplain.bst szerzője ennek ellenére kevés hibajelentésre számít, mert a legfőbb tipográfiai szabályokat a program betartja, és az ábécérend követése is kielégítő, és a kód egyszerűsége miatt az implementációs hibák várható száma is alacsony.

A BibTeX stílusfájl-szintaxisának és az alapfüggvények készletének a végletekig való egyszerűsítése, továbbá a *postfix* aritmetika és a különböző méretkorlátozások miatt a nyelv olyannyira elvarázsolt, hogy gyakran egy egyszerű stringmanipulációs függvény megírási ideje is félórás egységekben mérhető. Szerencsére nincsenek mágikus mellékhatások (mint például a TeX makrókifejtésében egyes szóközök és kapcsos zárójelek mágikus, kikapcsolhatatlan elnyelődése), ezért bármilyen procedurális programozási feladat több-kevesebb idő alatt megoldható BibTeX-ben. A BibTeX szerencsére be tudja olvasni a nyolcbites, ékezetes betűket is megengedő adatbázist, és a szavakat kézi manipulációval olyan formára lehet hozni, melyet a beépített sima, lexikografikus rendezés helyes sorrendbe rak. Mindent összevetve a BibTeX stílusfájlok ugyan fáradságosan készíthetők, de minden szempontból kielégítő megoldást nyújtanak egy magyar nyelvű irodalomjegyzék elkészítésére. Más a helyzet, ha azt szeretnénk, hogy minden tétel a saját nyelvének stílusában legyen kiszedve, illetve hogy ugyanaz a tétel különböző nyelven szedve a cím különböző fordításait tartalmazza: ekkor már egy többnyelvű bibliográfiai adatbázis-kezelőre van szükség, ilyen például az MLBibTeX (amely ugyan támogatja a magyar nyelvet, de tipográfiájának helyességét magyar szakértő még nem ellenőrizte).

4.1.4. Szakmai tapasztalatok

A kutatás során (főleg az általános [21] és a műszaki témájú [6] alapján) szerzett legfontosabb tapasztalat annak tudatosulása volt, hogy milyen gazdag a magyar tipográfiai hagyomány, milyen sok, a magyar nyelvű művek szerkesztése és szedése során felgyülemlett gyakorlati tudás van írásba foglalva. Ez tudás alkalmazása egyrészt segíti a szerkesztőt, hogy nagyobb hatékonysággal készüljenek a művek, másrészt biztosítja az olvasó számára a mű olvashatóságát, kereshetőségét és jó áttekinthetőségét. A rengeteg szabály, előírás és ajánlás teljes körű betart(at)ására e sorok írója a diplomaterv keretében nem vállalkozott, mivel ezek egy része már elavult (és több magyar tipográfust kéne szakértőnek felkérni, hogy pontosan mi változott és mi nem), továbbá egy ekkora munka túlmutat egy diplomaterven. A szabályok betartásán a tipográfusnak, a szerzőnek, a szedőnek és a szedőprogramnak együtt kell fáradoznia, a legokosabb program sem képes a képzetlen felhasználó által elkövetett hibák mindegyikét korrigálni, és a programok – bár a TeX bekezdéstördelési modellje ellenpélda erre – nem képesek az esztétikai alapon döntéseket hozni, főleg ritkán előforduló, egyedi esetekben. Épp ezért e sorok írója a kifejlesztett szoftverekben csak az automatikusan betartható tipográfiai előírások követésére szorítkozott, és igyekezett minél több szabályt ilyenné tenni (például a `\textqq` és `\MathReal` parancsok bevezetésével).

Lényeges tapasztalat volt még a TeX és a hozzá kapcsolódó formátumok, csomagok és szedőprogramok lazán csatolt együttműködésének megfigyelése. Szinte minden egyes szoftvernek más a szerzője, és nincs olyan szervezet, amely ezeket koordinálná, az együttműködést lehetővé tevő interfészeket definiálná, és nagystílű kutatási és fejlesztési terveket szőne. Minden fejlesztő a maga mozaikját beilleszti TeX-es szoftverek széles körképébe (a CTAN-be), a TeX felhasználói és más fejlesztők számára; a mozaikdarabok

nincsenek ráerőltetve a felhasználóra: abból építi saját művét, amelyikből szeretné. A TEX-disztribúciók azért születtek, mert egy átlagos felhasználó már nem látja át, hogy várhatóan mire lesz szüksége, és túl nehéz lenne fejben tartani a programok telepítésének pontos lépéseit, hogy azok egymással együtt tudjanak működni. A legegyszerűbb felrakni egy TEX-disztribúciót, és a CTAN-hez csak abban a ritka esetben fordulni, ha az adott szoftver nem része a disztribúciónak. Fontos különbség azonban e disztribúciók és a központilag fejlesztett szoftverek *release*-ei között, hogy a disztribúció készítői csak válogatnak a már elkészült programok közül, lefordítják, konfigurálják, könnyen letölthetővé és telepíthetővé teszik őket; a fejlesztésre azonban nincs közvetlen befolyásuk, és, szerencsére, nem vezetnek koncepcionális változtatásokat az eredeti szoftverekben. E decentralizált, spontán szerveződő fejlesztési modell legfontosabb hátránya a lassúság, és a kései alkalmazkodás a felhasználók és a környezet elvárásaihoz. Amikor a PostScript kimeneti dokumentumformátum elterjedt, Tomas Rokicki megírta a dvips-et; a PDF formátum előretörésével Hàn Thế Thành megíta a pdfTEX-et (és más megoldások is születtek, pl. dvipdfm), a Unicode szabvánnyá válása után John Plaice és Yannis Haralambous beépítették támogatását az Omega-ba, a magas magyar tipográfiai igényeket kielégítendő Nagy Bence a GrailTEX fejlesztésébe kezdett, a web mint médium berobbanásával elkészültek a LATEX → HTML konverterek stb. Ezek a fejlesztések mind egyének vagy is létszámú fejlesztői csoportok munkái, és gyakran (mint ahogy maga a TEX is) egy konkrét igényre készített egyedi megoldások letisztult általánosításai. Nem létezik egységes, megvalósítási lépésekre lebontott koncepció a TEX és a LATEX változtatási irányairól. Ez versenyhátrány, ami a különböző szoftverek laza integrációjából fakadó kompatibilitási problémákkal együtt a felhasználók táborának csökkenését vonhatja maga után, de e sorok írójának véleménye szerint a TEX és kiegészítőinek spontán módon létrejött decentralizált, nyílt és eken alapuló „fejlesztési modellje” végső soron a (tudatos) felhasználók érdekeit szolgálja: biztosítja a szabad szoftverhasználatot és a választást, és választási lehetőség híján a megfelelő új szoftver kifejlesztésének lehetőségét. A TEX használatához és bővítéséhez szükséges minden dokumentum, és az alapszoftverek forráskódja is szabadon hozzáférhető a CTAN-en. A MagyarLATEX sem készülhetett volna el ezek tanulmányozása nélkül. (Természetesen szabad szoftverrel is lehet pénzt keresni: erre jó példa a magyar TypoTEX, és a holland Pragma ADE könyvkiadó és szoftverfejlesztő cég.)

Ebbe a keretbe illeszkedik a LATEX magyarítása is. E sorok írója nem állítja (és nem lenne módja előidézni sem), hogy a MagyarLATEX legyen az egyetlen, legjobb, és kizárólagosan hozzáférhető megoldás. A MagyarLATEX pusztán egy mozaikdarab, amelyet bárki szabadon felhasználhat munkájához, a dokumentációt megtekintheti, és a programot szabadon bővítheti és módosíthatja, ha szükségletei így kívánják.

A LATEX-es programok és kiegészítők fejlesztése során leszűrhető általános tapasztalat, hogy bár sok minden programozható, és tág korlátok között automatizálható, az ügyetlen programozási nyelv gyakran lassítja, hátráltatja a programozót, és néha elveszi a kedvét a fejlesztéstől, vagy egyszerűen lekódolható, de speciális esetekben helytelenül működő szükségmegoldások születését segíti elő. A fejlesztési hatékonyság igencsak elmarad a modern RAD (*rapid application development*) eszközökétől, de még az ügyes szkriptnyelvek (pl. Perl, Ruby) és az alacsony szintű programozási nyelvek is (pl. C) maguk mögé utasítják. Nevezetes kivétel az ex-TEX, amely a TEX motor Java-s újraimplementációja, melyben az egyes alprogramok jól definiált interfészeken keresztül kommunikálnak, és igény szerint lecserélhetők: megoldható például a fájlmegnyitó rész lecserélésével, hogy közvetlenül a webről is be lehessen tölteni egy stílusfájlt a dokumentum fordításakor. Cserébe viszont kb. 10-szer lassabban fordul, és így e dolgozat lefordítása kb. 2 percig tart a tesztgépen (paramétereit lásd a 78. oldalon), ami azon kívül, hogy kivárhatatlanul

hosszú idő a mai WYSIWYG igények és kb. $^1/_3$ másodperces emberi reakcióidő mellett komoly ergonómiai hátrányt jelent, megakasztja a szerző lendületét, és a fordítással töltött két perceknél több idő vész el. Ennek csökkentésére vannak kezdeményezések, például texlite és preview-latex.

A diplomatervezés részét képező kutatás és szoftverfejlesztés során csak erősödött e sorok írójának azon meggyőződése, hogy a (LA)TEX és segédprogramjai képezik a legjobb dokumentum-előkészítési technológiát azok közül, amelynek végső tipográfiai és esztétikai minősége kiváló (beleértve a szöveget a képleteket, a táblázatokat és az ábrákat is), továbbá a dokumentum formai elemeinek széles skáláját (pl. utalások, hivatkozások, tartalomjegyzék, tárgymutató, irodalomjegyzék, lábjegyzetek, élőfej, névelők, számnévragozás) képesek automatikusan, különböző stílusokban generálni. Ez utóbbi szempont fontos szerepet játszott a magyar.ldf elkészítésekor. Dokumentumszerkesztőkkel és kiadványszerkesztőkkel összehasonlítva a LATEX-nek számos súlyos hátránya is van, de a mérleg műszaki és tudományos művek készítésekor az előnyök oldalára billen.

4.2. Továbbfejlesztési lehetőségek

4.2.1. magyar.ldf

- A longtable csomag `\caption` parancsa által generált táblázatcím magyar tipográfiájú szedése (kettőspont helyett pont).
- Csupa nagybetűs *AZ* névelő generálása.
- A `magyar-plain` tételszámozási stílus csak akkor aktiválódjon a preambulum végén, ha a magyar az alapértelmezett nyelv.
- Az `activespace=` által szabályozott karakterek használata `\shu` után.
- Az alul megjelenő lábjegyzet lehessen az őt követő szöveggel azonos méretű, ha a `\footnotestyle{bottom=textsize}` opcióval.
- A lábjegyzet utolsó csillaga a behúzástól jobbra álljon.
- A fancyhdr csomag alapértelmezett fejléceinek magyarítása.
- Betöltési opció (`listseps=zero`), hogy a `\@listi`-be, `\@listii`-be stb. a függőleges térközt csökkentő `\topsep=0` and `\parsep=0` kerüljön.
- A listakörnyezetek és egyéb kiemelések, például tételek és képletek alatti és fölötti köz csökkentése.
- A felsorolásjel behúzásának továbbá a jel és a szöveg távolságának beállítása a magyar hagyományok szerint. Működnie kell a paralist csomaggal, az *AMS* osztályokkal és azok nélkül is.
- A hyperref csomag `\autoref` parancsának és a magyar.ldf `\refstruc` parancsának összehangolása.
- A hyperref csomag által készített *bookmarks*-ban a magyar aktív karakter jól működjön.
- A levelek írását segítő letter.cls stílus lokalizálása a magyar hivatalos és magánlevél tipográfiája szerint.
- A `\hunnewlabel` parancs a nameref csomag mintájára megjegyezhetné, hogy struktúrális egységről, vagy az alábbiak valamelyikéről van szó, hogy a `\refstruc` parancs a helyes feliratot helyezhesse el: `table`, `figure` or `equation`.
- Új virtuális fontok és virtuális font-generálási módszerek kidolgozása, hogy dupla többjegyű mássalhangzók gépeléséhez ne legyen szükséges az aktív karakter (`‘tty`).
- Az Omega alaprendszer bővítése, hogy dupla többjegyű mássalhangzók gépeléséhez ne legyen szükséges az aktív karakter (`‘tty`).
- A soregyen pontos betartása (ez szinte lehetetlen LATEX)-ben. [3] is említi.
- Egyéb díszpontok a `\paragraph` parancs címének és a `description` környezet ele-

meinek lezárására ([3]).
– A strukturális egységek címei ne legyenek nagyobbak a főszöveg betűméreténél ([3]).
– Félkövér betűk használata félkövér nyújtott helyen minden lehetséges esetben. Ha nem lehetséges, tovább futás félkövér nyújtott betűkkel figyelmeztető üzenet nélkül ([3]).
– Ha üres az oldal, ne legyen se élőfej, se oldalszám ([3]). Egyelőre nem világos, hogy a LaTeX `\output` rutinjába hol érdemes belenyúlni (amely részt más csomagok nem írnak át) az üres oldal detektálására.
– `\parindent` számolása `\textwidth` alapján ([3]): `\textwidth` $\leq 24\,$cc (or $20\,$cc) $\Longrightarrow$ `parindent` $:= 1\,$quad különben $2\,$quad ([3])
– Az OT1 fontkódolásból hiányoznak a `\H` (˝) és `\.` (˙) ékezetek írógép-betűtípusok esetén. Az ékezetek automatikus pótlása egy hasonló családba tartozó betűtípussal. Például `\texttt{Erdős}` ne Erdos-t jelenítsen meg.
– Az OT1 kódolásban ismert `\umlautlow` (és `\umlauthigh`) parancs megvalósítása (vagyis az ékezetek lejjebb mozgatása) T1 kódolású virtuális fontok automatikus generálásával. A T1 kódolású EC fontokban nem látszik különbség, de az OT1 kódolású CM-ekben annál több.
– Meggondolni, hogy a leíró list (`description` környezet) címszavai félkövér helyett dőlt betűvel szedődjenek-e.
– `\told\az+\cite{foo}+ban{}` működésre bírása, hogy ne kelljen `\told\acite{foo}+ban{}`-t írni.
– Néhány betöltési opció értékét futás közben is változtatni lehessen.
– a [3]-ban levő `*hyph*.tex`-re vonatkozó megjegyzések figyelembe vétele illetve továbbítása a `*hyph*.tex` szerzőihez.
– `\parskip=0pt` beállítása a preambulum végén, kivéve `\parindent=0pt`-t használó rövid dokumentumok esetén ([3]).
– A bekezdés utolsó sorának hosszára vonatkozó korlátok betartatása (lásd a 25. oldalon és [3]-ban). Donald Aresenaurre vonatkozó ötleteinek tesztelese egy hosszabb művön, és annak megvizsgálása, hogy mely LaTeX csomagokkal akad össze.
– A `\section` és társai körül hagyott függőleges köz csökkentése ([3]).
– Az „értelmezés\per beállítás” szedésére a `\per` parancs definiálása (nem kötődik szorosan a magyar nyelvhez): `\def\per{/\penalty\exhyphenpenalty\hskip\z@skip}`
– A hierarchikus listakörnyezetek vízszintes behúzásának előállítása `\parindent` többszöröseire, és [3]-ban említett egyéb szabályok figyelembe vétele, például tartalmaz-e a felsorolás többsoros elemet (ezt LaTeX-ben lehetetlen előre detektálni, ezért a szerzőnek kelljen külön jeleznie, ha nem).
– A layout és egyéb csomagok magyar feliratainak elkészítése (eddig egyedül a varioref csomagé készült el). Valószínűleg az alábbi elég lenne:

```
%\DeclareOption{magyar}{%
 \def\Headertext{Fejl\'ec}
 \def\Bodytext{A dokumentum teste\qquad}
 \def\Footertext{L\'abl\'ec}
 \def\MarginNotestext{Sz\'el-\\jegyzet}
 \def\oneinchtext{1 inch}
 \def\notshown{nem l\'atszik}}
```

– a ‘ aktív karakter visszavonása matematikai módban, hogy a nath csomag ezt H_{symm} így `$H_{‘symm}$` és `H_‘‘symm␣` is kiszedhesse (a nathb csomag nélkül is). Az alábbi kódrészletből érdemes kiindulni:

```
%\let\@@magyar@saved@mathoptions@on\mathoptions@on
%\def\mathoptions@on{% overrides \iiik{nath.sty}
 \catcode‘‘12 % deactivate
 \begingroup\lccode‘~‘‘\lowercase{\endgroup\def~}{\abbreviation}%
 % ^^^ \def‘{\abbreviation}, but with active ‘
 \@@magyar@saved@mathoptions@on}
```

– Hivatalos opció, amivel elő lehet írni, hogy a dokumentum csak a magyar.ldf egy elég új változatával forduljon le.
– „A. függelék", „A függelék" és „1. függelék" közül melyik a helyes? [5] az utóbbit ajánlja.
– A `\told\ref{foo}+ban{}` akkor is működjön, ha az `\r@foo` címke száma üres, mert még nem volt `\section` vagy hasonló parancs a forrásban.
– Végződjön-e az `.aux` fájlba írt `\newlabel` parancs argumentumában található fejezet, ábra stb. száma pontra? Jelenleg nincs a végén pont, és a hivatkozásba kell pontot írni, például: `\aref{ekezetek}.\ táblázat`. Az ezen való változtatás megtörné a kompatibilitást.
– Opció a `\index{|run husort.pl}` automatikus kiadására, esetleg a `defaults=` értékének megfelelő husort.pl opciókkal.
– Az aktív karakter(ek) (pl. ‘ és ?) futásidejű ki-be kapcsolgatása. ngermanb.ldf `\def\mdqon{\shorthandon{"}}`, ukraineb.ldf `\def\mdqon{\bbl@activate{"}}` definíciót tartalmaz.
– Az `\initiate@active@char` megjavítása (hogy ne aktiválja túl korán a karaktert), ezáltal más nyelvek aktív karakterei is megjavulnak.
– A spanish.ldf-ből jó ötletek átvétele (`\addto\shorthandsspanish`, `\es@activate`, `\if@safe@actives`).
– Miért szerepel az az esperanto.ldf-ben: `\declare@shorthand{system}{^}{\csname normal@char\string^\endcsname}` ?
– Az `accents=low` a `\.`, `\,`, és `\H` ékezetekre is hatással legyen, ne csak a `\"` (¨)-ra. A `\.` az „i" betűhöz szükséges.
– Ellenőrizni, hogy a magyar.ldf teljesen kompatíbilis az amsart.cls-szal és más *AMS* osztályokkal.
– Annak megértése, hogy a huhyph3.tex miért választja el helytelenül az alábbi értelmetlen szavakat: *az-t-he, zs-hehe, sza-badstrand.*
– A hyperref csomag által generált tárgymutató bővítése formázó tag-ekkel, pl. `\index{foo|hyperpagearg{\textbf}}` (nem kapcsolódik szorosan a magyar nyelvhez).
– A `titleskip=`, `amstocnumskip=` opciók és a `\magyar@post@...` egybeolvasztása.
– A `\textqq` parancs környezetként (`\begin{textqq}`) is működjön.
– A tartalomjegyzékben a szám és a cím közötti kihagyás csökkentése (minden szinten), de csak akkor, ha az egész fában minden bejegyzés belefér.
– `\restoreparindent` definiálása listakörnyezeten belül (nem kötődik szorosan a magyar nyelvhez), amely a második bekezdésüktől kezdve behúzásossá teszi a listaelemeket. Az alábbi definíciót kéne a `\@listi`-ben és a `\@listI`-ben elhelyezni: `\edef\restoreparindent{\parindent\the\parindent\relax}`.
– Az `\emitdate{f}` által generált hónapnevek toldalékolása, például `\emitdate[a+nak]{f}{1848-03-15}+a+nak{} közepe` → 1848 márciusának közepe.
– Az `\emitdate{h}` birtokos személyragot automatikusan a hónapnév mögé, engedjen meg más ragokat is: `\emitdate[rol]{h}{1848-03-15}` → 1848 márciusáról. `{f}` és `{h}` összevonása.
– A `\told*` definiálása, ami csak a kötőjelet és a toldalékot rakja ki.

– A \told toldalék végét megkereső programkód rövidítése, áttekinthetővé tevése.
– Alternatív \told beviteli szintaxisok kipróbálása implementációs szempontból, például \told45{-ed-hez}, \told{45}-ed-hez, \told45\od\hoz, \told45+\od\hoz.
– *-val/vel* rag helyes hozzáillesztése második toldalékként, például \told45+adik+val{} → 45-ödikkel.
– \told\tort13+ad+val{} → $^1/_3$-dal.
– \MathReal bővítése a frenchb.ldf \nombre parancsának tudásának beépítésével.
– A \MathReal-nek beállítható legyen, hogy a 9999-nél nagyobb számok egész részét hármasával csoportosítsa, például 12 345.
– A \MathReal bővítése fizikai mennyiségek szedésére ([21, 250–251. oldal]), például \MathReal{-3331234.5678pF^{-1}} → $-3\,331\,234{,}5678\,\mathrm{pF}^{-1}$; csak akkor jelenítsen meg tizedesvesszőt, ha a magyar nyelv aktív, \az-on belül is működjön.
– A frenchb.ldf \CyrillicGuillemets és \LasyGuillemets parancsaihoz hasonló megoldás a lúdlábak megjelenésének futásidejű szabályozására.
– Az angol nyelvű fejlesztői dokumentáció (magyar.dtx) befejezése.

4.2.2. husort.pl

– Figyelmeztető- vagy hibaüzenet esetén az .idx fájl-beli sorszámon kívül a TEX dokumentum-beli oldalszám megjelenítése.
– .ilg fájl készítése.
– A konfiguráció beolvasása .ist vagy ahhoz hasonló formátumú fájlokból.
– A t1_pres2alpha függvény által a rendezés során figyelmen kívül hagyott parancsok listája (pl. \rm, \bfseries, \allowbreak, \tiny, \-) konfigurálható legyen.
– Rendezés az angol szabályok szerint (ez megegyezik az általános latin betűs ábécével), és a német szabályok szerint, melyek pl. az *ä* betűt ae-ként veszik figyelembe (DIN 5007), vagy a-ként veszik figyelembe (Duden).
– A consonant_equiv opció bevezetése, amely több külön betűnek tekintené a többjegyű mássalhangzókat (S5b. szabály a 63. oldalon).
– A magyarp.xdy Xindy stílusfájl kifejlesztése, amely – a Xindy keretein belül – majdnem mindenben úgy viselkedik, mint a husort.pl.
– Az S10a. és S10b. szabályok megkülönböztetése, a tételt nyitó határozott névelő figyelmen kívül hagyása a tárgymutatóban.
– Külön névmutató-stílus születési, halálozási és uralkodási évszámokkal.

4.2.3. huplain.bst

– A rendezési szabályok ([12, §16]) pontosabb követése.
– Opció, hogy a bibliográfiai stílus a hivatkozott mű nyelvét kövesse.
– Opció a szerzők nevének kiskapitálissal való szedésére ([22, 73. oldal] ezt megengedi).
– A *huname* mező a szerzők nevét külön-külön szabályozhassa.
– A *crossref* mező helyes kezelése.
– Betartani a [21, 312. oldal] szerinti elvet, miszerint a háromnál több szerzővel rendelkező művek címét előrébb kell helyezni a szerzőnél.
– {\NG} automatikus kisbetűsítése.
– Egyéb bibliográfiai stílusok, például unsrt.bst, abbrv.bst, abbrvnat.bst, amsplain.bst magyarítása.
– A nemzetközi bibliográfiai adatbázis-kezelő programok, például az MLBIBTEX magyar moduljának kifejlesztése.
– A *teTeX-url* mező helyett az általánosabb *TDS-url* mező bevezetése.

Köszönet és hála

E sorok írója szeretne köszönetet mondani mindazoknak, akik egyetemi tanulmányai során segítették. A szerző nem vezetett naplót mindarról a bátorításról, kedvességről, türelemről, idő rááldozásról és anyagi támogatásról, melyben az utóbbi öt évben része volt; ezért lehetségesnek, sőt valószínűnek tartja, hogy az alábbi tételes listában elfelejt megemlíteni sokakat, akiknek köszönettel és hálával tartozik. Kéri ezért a kimaradókat, hogy pusztán e hiányos emlékezés miatt ne tápláljanak haragot iránta. Elnézést kér továbbá azoktól, akiknek nehezményezik, hogy nevük és tudományos fokozatuk nem a lehető leghosszabban szerepel a felsorolásban.

Szülei biztosítottak otthont, nyugalmat és ellátást a szerzőnek, hogy tanulmányaira koncentrálhasson – eme áldozat nagyságát a szerző majd csak akkor érezheti át, amikor saját gyerekeinek próbálja meg minden kitartását összeszedve ugyanezt nyújtani. *Kinga* nevű húga erőn felül vette ki részét a házimunkából, amikor a szerző nem tartózkodott otthon, és akkor is, amikor igen. *Wettl Ferenc* több ízben is igen jó főnöke volt, aki szem előtt tartotta beosztottja érdekeit, és mindent megmozgatott, hogy segítsen neki. A szerző visszaemlékszik olyan esetekre, amikor durva hibákat követett el a rábízott munkában, és Feri minden esetben jelesre vizsgázott építő kritikából. Külön köszönet illeti Feri mellett *Kollár Istvánt,* aki konzulensi minőségében szintén segítette e dolgozat elkészültét. *Hegyi Veronika* nagy lelkesedéssel vetette magát a dolgozat átnézésébe, stílus- és fogalmazásbeli javaslataival sokat segített a szerzőnek. *Pröhle Péter* meghallgatta a szerzőt, és bármiről felelősséggel adott tanácsot neki: a témát többször is körüljárta, tiszta érvei gyakran váratlanok, de utólag annál nyilvánvalóbbak voltak – minden megnyilvánulásában szívén viselte a szerző sorsát. *Fritz Józsefné* mindig őszinte volt a szerzőhöz: építő módon szembesítette a hibáival, és nem titkolt el semmit előle, az összefüggésekre bátran, egyenesen hívta fel a figyelmét, és sok mindenre rávilágított. *Halász Claudia* – azon kívül, hogy a tanszéken mindent elintézett számára – nem kevés időt szakított arra, hogy felvidítsa, támogassa, etesse és itassa a szerzőt, minden mondatában a szerző javát keresve. *Mihók Zsuzsa* volt az a hivatalban, aki már azelőtt kérés nélkül megoldotta az adminisztratív problémákat, hogy azok a szerzőben megfogalmazódtak volna: az index, a matricák és az űrlapok mind maguktól, vagyis Zsuzsa kezétől a helyükre kerültek; a szerző csak egy dolog elintézése végett nézett be a dékániba, de ezen felül Zsuzsa vidáman megoldott még kettőt; mindezt számottevő sorban állás nélkül, még akkor is, amikor a többi ajtó előtt kígyóztak a sorok. Rejtély, hogyan csinálta. *Bujdosó Gyöngyi* segítette a szerzőt a TEX-konferenciákon, annak ellenére, hogy a szerző figyelmetlenségében és önelégültségében ebből semmit sem vett észre. Gyöngyi példát mutatott még teherbírásból, kitartásból és felelősségből. *Nagy Dániel* sok eredeti gondolatát megosztotta a szerzővel, de nem ódzkodott megfeddni sem, amikor baklövést követett el. Az E épület földszintjén dolgozó büféslányoktól külön öröm volt egy pár virslit vásárolni. A közelben üldögélő portások gyakran nyitották ki a kaput a késő este hazainduló szerzőnek. *Mann Zoltán* igen sokat segített a készülő TDK dolgozatban, és igen aprólékosan nézett át minden betűt és írásjelet. *Hajnács Zoltán* egy éjjel, amikor nem adódott más közlekedési lehetőség, hazafuvarozta a szerzőt. Egy néni a központi könyvtárban megengedte a nem túl barátságos arcú szerzőnek, hogy a tilalom ellenére internetezzen. *Flóra* és *Nóra*, a Matematika Intézet könyvtárosai minden könyvet megtaláltak, és néha még lépcsőztek is, hogy személyesen a szerző kezébe adhassák. *Vecsei Balázs* beugrott órát tartani, amikor a szerző nem ért rá. *Törley Gábor* egy ritka, az önzetlenség, a szeretet és a másokon segíteni vágyás értékeit valló társaságba hívta meg a szerzőt, és példát mutatott ezekből az értékekből. *Maczkó Katalin*, aki a diplomaterv beadásának napján már boldog házasságban

él, nem kevés szilvára való éjszakai séta és pesti plázázás, sőt, egy ízben még egy vacsorameghívás során is aktív figyelmét a szerző gubancos gondolataira fordította, vidám társasága páratlan színfoltot jelentett a szerzőnek, és még ki tudja, mi mindent. *Gergi Miklós* sok izgalmas programozási talány felvetésével késztette gondolkozásra a szerzőt, és rendszergazdaként, szakmai tanácsadóként és házimozi-partnerként mindig rendelkezésre állt. *Vető Kata* csatlakozott a szerzőhöz Párizsban egy séta és egy ebéd idejét kitöltő kötetlen beszélgetésre, és ezzel többet segített, mint gondolná. *Benedek Balázs* és *Marx Dániel* nagy lelkesedéssel és magas szakmai színvonalon szerveztek többféle programozási versenyt a hallgatók számára. *Friedl Zita* elfogadta a szerző ajándékát, és épségben vissza is juttatta a szerzőhöz, amikor már nem volt rá szüksége. *Nagyné Szilvási Márta* hajlandó volt heteket és hónapokat is várni, és gondoskodott arról is, hogy a szerzőnek ne korogjon a gyomra munka közben.

A felsorolásból már csak azok vannak hátra, akik előadóként, gyakorlatvezetőként vagy laborvezetőként a szerzőt tanították. *Tóth Csaba* szóban és tettben is azon fáradozott, hogy jó mérnök váljon az őt hallgatókból; értelmes embereknek értelmesen magyarázott, minden mondat, minden közbevetés és minden ábra a helyén volt az előadásán; a jól szervezett oktatás őszinte érdeklődéssel és szakmai segíteni akarással párosult. *Selényi Endre* óráin ügyesen keverte a gyakorlatias, készségfejlesztő oktatást a jó humorral, nem hagyta lankadni a hallgatóság figyelmét. *Csima Judit* bravúros módon zsúfolta be a tananyagot a gyakorlati foglalkozás rövid idejébe, és mindent érthetően, információvesztés nélkül sikerült elmondania – még a kis-ZH kérdések is megválaszolhatók voltak. Gondolkodtató szorgalmi feladatokat adott, és mindig rendelkezésre állt a tananyaggal és bármilyen egyéb matekos témával kapcsolatban. *Szeszlér Dávid* tartotta a leginteraktívabb, az agytekervényeket legjobban megmozgató gyakorlatot – a tudás néhány érdekes feladat megoldásának melléktermékeként állt elő. Dávid is nyitott volt minden szakmai kérdésre. Később előadóként is remekelt: a táblakép és az elmondottak azonnal világossá tették a bonyolult matematikai levezetést. E tárgy keretében nagy türelemmel viselte a szerző egyéni problémáit. A tárgy egyéb oktatóinak is érdeme, hogy kiváló jegyzet készült. *Botzheim János* ifjú titánként tartott jól sikerült órákat, szemet hunyt a szerző háromnegyed órás késései fölött, és a vizsgajegy kialakításakor érdemként vette figyelembe az órákon való aktív részvételt. Szívügyének tartotta a tárgy oktatását. *Rónyai Lajos* megtalálta a módot arra, hogy érdekes algoritmusokat és azok precíz matematikai elemzését közérthetően tálalja az informatikus hallgatóságnak. *Serény György* lelkesedéstől átfűtött előadói stílusa fokozhatatlan élménnyé tette a logikaórákat, és igazi kihívást jelentett a szerzőnek a többszörösen indirekt, a biológia felvételik relációanalízis-feladatait maga mögé utasító záróteszt hibátlan kitöltése. *Bach Iván* választékos stílusával káprázttatta el a hallgatóságot; a folyamatos beszédben nem veszett el az anyag struktúrája sem. *Koblinger Egmont* gyakorlati foglalkozásait könnyű volt követni, gondolkodását lelassította arra a másfél órára. A ZH-kat jóindulatúan javította, de maximális pontszámért néha reklamálni kellett. A szerző bátran fordulhatott hozzá kérdéseivel, minden problémára a lehető legtisztább, legegyszerűbb megoldást találta meg, és olyan szempontokra is rámutatott, amely másokban fel sem merültek. *Bodon Ferenc* közvetlen, barátságos hangulatban igen nagy anyagot tudott áttekinteni egyetlen félév alatt, és bár a jegyzet riasztóan töményre és hosszúra sikerült, csak a megtanulható részeket kérte számon. *Apagyi Barnabás* rendelkezésre bocsátotta a saját előadói jegyzeteit, hogy a szerző felkészülhessen a vizsgára. Ugyanebből a tárgyból *Horváth Miklós* extra konzultációs lehetőséget biztosított, amikor a szerző végképp reménytelennek látta a helyzetét. *Héthelyi László* nem törődött a formaságokkal, amikor a szerző igen zavartan és tanácstalanul állt az algebrai tételek között egy szóbeli vizsga során, és előadóként is kedélyes, épp hogy észrevehetően szórakozott

matematikus képét mutatta. *Halácsy Péter* igazán érdekfeszítő, beszélgetős órákat tartott a számítógépes nyelvészetről. Előadótársa, *Hamp Gábor* sosem emelte fel a hangját, a másik véleményét tiszteletben tartó békítő szándék áradt minden mondatából. *Vajda István* a tiszta ész megvilágosító erejével teremtett rendet a ZH-ra készülés során információval túltelített hallgatók gondolatai között – az előadásain végigvett egyszerű gondolatmenet megnyugtatta a szerzőt, hogy ehhez hasonlót ő is tud majd produkálni a számonkérés során. *Vetier András* bevezette a lélektani ZH fogalmát, és alaposan helyre tette a szerző elbizakodottságát saját tanulási stílusában. A félévi jegy végül mindig jól alakult. *Charaf Hassan* minden szempontból a hallgatóság érdekeit nézte, és céges kapcsolatait is latba vetette mind az oktatás, mind a hallgatók karrierjének előmozdítása terén. A szerző főleg az ő számára ajánlja ezt a dolgozatot. *Gajdos Sándor* elit pizzázásra hívott néhány, a vizsgán jól szereplő hallgatót. *Galántai Zoltán* tartotta a legvidámabb órákat, és eközben fontos társadalmi kérdésekről nyílt mód eszmecserére. *Verhás Péter* tartotta a leggyakorlatiasabb előadást pusztán diavetítéssel: a programozás több izgalmas területét összefogó, nem triviális kódrészletekkel gazdagon illusztrált kurzus jártasságot és készséget is adott. *Recski Andrásról* nemcsak közérthető matematikai témájú előadások, hanem végtelen, az emberi hibákat elnéző türelme és a hozzá forduló hallgatók munkáját segítő jótanácsok és magas szintű szakmai támogatás marad meg a szerző emlékezetében. *Szeredi Péter* igen jó, a lényeget kiemelő fóliákat készített minden általa oktatott tárgyhoz, és igen sok időt szánt az egyéb oktatási segédanyagok, például egy webes gyakorlórendszer kifejlesztésére; sikerült elérnie, hogy a tanulási folyamat ne egy leküzdendő feladat, hanem érdekességeket rejtő és megvilágosodást tartogató izgalmas kaland legyen. A szerző külön köszönettel (és még mással is) tartozik neki, hogy a jócskán és tetőzötten megmutatkozó emberi gyengeségeire sosem reagált bántóan, és mikor a szerző ezek miatt szorult helyzetbe került, mindig nyitott neki egy új kaput és egy új lehetőséget.

Hála és köszönet érte.

Hivatkozások

[1] Johannes L. Braams: *Babel, a multilingual package for use with LaTeX's standard document classes*. 2001. CTAN `macros/latex/required/babel/babel.dtx`.

[2] Bujdosó Gyöngyi – Fazekas Attila: *TeX kezdőlépések*. Budapest, 1997, Tertia Kiadó.

[3] Bujdosó Gyöngyi – Wettl Ferenc: On the localization of TeX in Hungary. *TUGBoat*, 23. évf. (2002) 1. sz. ISSN 0896-3207.

[4] Csikós Zsolt: *Számítógéppel számországban*. Budapest, 1990, Szerzői kiadás.

[5] Gyurgyák János: *Szerkesztők és szerzők kézikönyve*. Osiris kézikönyvek sorozat. Budapest, 2000, Osiris Kiadó. ISBN 964 379 227 4. ISSN 1416-6321.

[6] Horváth Jenő – Makk Attila – Tartsay Vilmosné: *Hogyan írjunk, szerkesszünk műszaki könyvet?* Budapest, 1982, Műszaki Könyvkiadó. ISBN 963 10 4449 8.

[7] Juhász József és mások (szerk.): *Magyar értelmező Kéziszótár*. 9. kiad. Budapest, 1992, Akadémiai Kiadó.

[8] David Kastrup: De ore leonis. Macro expansion for virtuosi. In *EuroBachoTeX* (konferenciaanyag). 2002. május.

[9] Donald E. Knuth: *The Art of Computer Programming*. 2. kiad. Reading, Massachusetts, 1981. január 10., Addison-Wesley.

[10] Donald E. Knuth: *The TeXbook*. 1984, Addison–Wesley.

[11] Donald E. Knuth: *A számítógép-programozás művészete*. 2. kiad. Budapest, 1994, Műszaki Könyvkiadó. ISBN 963 16 0075 0.

[12] Magyar Tudományos Akadémia (szerk.): *A magyar helyesírás szabályai*. Budapest, 1984, 1994[11], Akadémiai Kiadó.

[13] Mayer Gyula: *A TeX és a LaTeX magyar elválasztási modulja*. 2002. július 10. URL `http://www.math.bme.hu/latex/huhydok.pdf`. Publikálatlan.

[14] Nagy Bence: *Huhyphn – magyar elválasztás TeX-hez, Scribus-hoz és OpenOffice.org-hoz*. 2003. október 11. URL `http://www.freeweb.hu/gimb/files/huhyphn.pdf`.

[15] Németh László: *Magyar ISpell dokumentáció*. 2003. július 22. URL `http://magyarispell.sf.net/magyarispell.pdf`. A HunSpell leírása is.

[16] Németh László: *A Szószablya fejlesztés*. 2003. október. URL `http://magyarispell.sf.net/szoszablya.pdf`.

[17] Tobias Oetiker – Hubert Partl – Irene Hyna – Elisabeth Schlegl: *The Not So Short Introduction to LaTeX 2ε*. 1999. január 8. CTAN `info/lshort/english/lshort.dvi`. teTeX `doc/latex/general/lshort.dvi`.

[18] Szabó Péter: *The Hungarian language*. CTAN `macros/latex/required/babel/magyar.dtx`. A magyar.ldf v1.5 felhasználói és a teljes forráskódot tartalmazó és azt magyarázó fejlesztői dokumentációja. Megjelenés alatt.

[19] Szabó Péter: Implementation tricks in the Hungarian Babel module. *TUGBoat*, 25. évf. (2004) 1. sz. ISSN 0896-3207.

[20] Szántó Tibor: *A betű*. Budapest, 1982, Akadémiai Kiadó.

[21] Timkó György (szerk.): *Helyesírási és tipográfiai tanácsadó*. 2. kiad. Budapest, 1972, Nyomdaipari egyesülés.

[22] Virágvölgyi Péter: *A tipográfia mestersége – számítógéppel*. Osiris kézikönyvek sorozat. Budapest, 2001, Osiris Kiadó. ISBN 963 379 529 X. ISSN 1416-6321.

[23] Wettl Ferenc – Mayer Gyula – Sudár Csaba: *LaTeX kezdőknek és haladóknak*. Budapest, 1998, Panem. ISBN 963 545 141 5.

[24] Wettl Ferenc – Mayer Gyula – Szabó Péter: *LaTeX kézikönyv*. Budapest, 2004, Panem. Megjelenés alatt.

[25] Hermann Zapf: *About micro-typography and the* hz*-program*. 1993. szeptember.

Tárgymutató

Számok

88591lat.csf fájl 69

A, Á

a *66*
a/az 19
abbrv.bst fájl 89
abbrvnat.bst fájl 89
ábécérend 18
AbiWord program 13, 16
Adobe Reader program 60
aer10.vf fájl 50
amsart.cls fájl 88
amsmath csomag 43, 44
amsplain.bst fájl 68, 89
$\mathcal{A}\mathcal{M}\mathcal{S}$-TEX 16
amsthm csomag 36, 46, 47
article.cls fájl 35
ASCII 64, 69
.aux fájl 52, 54, 55
.bib fájl 67

B

babel csomag 10, 11, 32, 34, 35, 39, 41, 42, 47–49, 56, 57, 81, 82
babel.def fájl 38
babel.sty fájl 56
baseline grid lásd soregyen
behúzás 24
bekezdéses cím *lásd* megnevezett bekezdés
better_afm2tfm.pl program 10, 11
bibliográfiai tétel 67
BibTEX 4, 11, 32, 67–74, 83, 84
BibTEX8 10, 68, 69
book.cls fájl 35
book.cls csomag 40
bookmarks 49, 86
break line 25

C

C 19, 35, 50, 53, 81, 85
ccs_extract.pl program 10, 11, 60
cégek
  Adobe 19, 23
  AMS 36, 39, 40, 45, 47, 86, 88
  BlueSky 39
  GNU 9, 10
  Morphologic 18, 29, 75
célpont *lásd* címke
cím 24
címke 51
címoldal 28
címrendszer 26
CM betűcsalád 24, 60, 87
CM-Super program 10, 39, 60
Computer Modern 21
ConTEXt 16, 17, 33, 82
copyrightoldal 28
CSS 14
CTAN 10, 32–35, 44, 60, 69, 74, 84, 85
CVS program 59

Cs

ccs_extract.pl fájl 61
csővezetés mód 74

D

dátum 20
Debian 10, 60, 62, 68, 75, 78
desktop publishing software lásd kiadványszerkesztő
DIN 5007 89
diviz 20
document preparation lásd dokumentum-előkészítés
dokumentum 11
dokumentum-előkészítés 11
dokumentumszerkesztő 13
Donald Aresenau 25, 87
Donald E. Knuth 16, 81
döntési mód 76
döntött 24
Duden 89
DVI 74
dvipdfm program 85
dvips program 85

E, É

EC betűcsalád 60, 87
egalizálás 22
  ~i táblázat 22
élőfej 28
elválasztás *lásd* szóelválasztás
elválasztási minták 57
elválasztójel 20
Emacs program 13, 76–79
em-dash lásd nagykötőjel
en-dash lásd gondolatjel
en-space 24
entry lásd bibliográfiai tétel

Epsilon program 78, 79
Erdős Pál 17
esperanto.ldf fájl 88
ε-TEX 81
Excel program 34
expandable lásd kifejthető
extended lásd nyújtott
ex-TEX 85

F

fancyhdr csomag 86
félkvirtmínusz 20
field lásd mező
first line indentation lásd behúzás
Flash 16
f-ligatúra 19
flyspell.el fájl 77
fmtutil program 57, 60
fnpara csomag 56
footmisc csomag 52, 56
footnote lásd lábjegyzet
format lásd formátum
formátum 16
FrameMaker program 16
Franz Schmid 17
free software lásd szabad szoftver
frenchb.ldf fájl 39, 49, 89
ftnright csomag 52
függőleges lista 52

G

gondolatjel 20
GPL 9, 10
GrailTEX program 16, 17, 31, 85
graphicx csomag 49
graphics csomag 36, 49

H

Halotti beszéd 18
Hàn Thế Thành 85
határozott névelő 19
Helyes-e? program 29, 75
Helyesebb program 29
Helyesel program 18
hint 39
hook 52
HTML 11, 14, 85
huhyphc.tex fájl 42, 58, 59
huhyphf.tex fájl 42, 58, 59
huhyphn.tex fájl 42, 48, 58, 59
huhyph.tex fájl 59, 60
huhyph3.tex fájl 48, 58–60, 88
HunSpell program 32, 75–77, 79
HunStem program 76
huplain.bst fájl 10, 11, 32, 67–74, 80, 83, 84, 89
husort.pl program 10, 11, 32, 33, 61–66, 80, 82, 83, 88, 89
huspell.vim fájl 79
hyhyph3.tex fájl 58
hyperref csomag 36, 49, 55, 86, 88
hyphen lásd diviz
hyphenation patterns lásd elválasztási minták

I, Í

inch 22
indentation lásd behúzás
InDesign program 14–16
index csomag 65, 82
index processor lásd indexrendező program
indexrendező program 61
iniTEX program 57
inputenc csomag 46, 65, 82
interaktív mód 74
írásjel 21
ISO-8859-1 *lásd* Latin-1
ISO-8859-2 *lásd* Latin-2
ISpell program 74–78
ispell.el fájl 77, 78
ispell.pl program 10, 11, 32, 77, 79

J

Java 63, 85
Johannes Braams 81
John Plaice 85

K

KDE 76, 77
képaláírás 27
kerning lásd egalizálás
kerning table lásd egalizálási táblázat
kétszeres betöltés lásd dual load
kétszeres betöltés 56
kézirat 11
kiadványszerkesztő 13
kifejthető 42, 52
kimenetoldal 26
kimenetsor *lásd* utolsó sor
kinézeti terv 12
kivonás jele 20
kötőjel 20
Kpathsea program 69
kvirtmínusz 20
KWord program 16

L

lábjegyzet 27
lafmtgen.pl program 10, 11, 34, 57, 60
language.dat fájl 57, 60
LaTeX 10, 14–16
latex.ltx fájl 35
Latin-1 4, 17, 18, 70
Latin-2 4, 16–18, 63–65, 70
layout lásd kinézeti terv
layout csomag 87
laza sor 51
Leslie Lamport 15
letter.cls fájl 35, 86
LibHnj program 58
ligatúra 19
Linux 32, 75, 79
LISP 62, 78
lokalizálás 8
longtable csomag 86
loose line lásd laza sor
LyX program 16, 76

M

magas törtek 22
magyar.dtx fájl 34, 81, 89
MagyarISpell program 32, 75–77, 79
MagyarLaTeX program 10, 11, 22, 29–32, 34, 59, 60, 62, 67, 68, 79, 85
magyar.ldf fájl 4, 10, 11, 29, 32–43, 45–57, 59–61, 68, 80–82, 86, 88
magyar-minta.bib fájl 67
MagyarOffice program 16
magyarp.xdy fájl 62, 89
magyar.xdy fájl 33
makeindex program 33, 61, 62, 65, 66, 82, 83
mami program 33
Márai Sándor 18
Mayer Gyula 58
megnevezett bekezdés 24
mértékegység 23
mező 67
Microsoft Word program 13, 14, 29, 31
Minion 21
MLBibTeX 67, 84, 89
monospaced lásd változó szélességű
morfológiai elemző 18, 76
mozgó argumentum 56
MSpell program 32, 75–77, 79
mű 11
MySpell program 75, 76

N

Nagy Bence 58, 85
nagykötőjel 20
nameref csomag 55, 86
nath csomag 43, 44, 87
nathb csomag 87
négyzet 24
Németh László 75, 76
ngermanb.ldf fájl 88
NotePad program 13, 16
ntheorem csomag 36, 46

O, Ó

Office XP program 75
Office 97 75
ogonek 18
oldstyle numerals lásd ugráló számjegyek
OM betűcsalád 39
Omega program 33, 81, 82, 85, 86
OpenOffice program 13, 16, 58, 76
OpenType 15, 19, 23
optikai sorkizárás 22
OT1 fontkódolás 20, 38, 87
outlines 49

P

paragraph lásd megnevezett bekezdés
paralist csomag 86
PatGen program 58, 59
Paul Renner 12
PDF 11, 16, 49, 60, 74, 85
pdfTeX 85
Perl 10, 11, 22, 33, 60, 62, 63, 77, 80, 82, 85
plain.bst fájl 67, 68, 70, 72, 73
plainTeX 16, 17, 34, 39
postfix aritmetika 69, 84
PostScript 11, 16, 69, 85
power user 35
Pragma ADE 85
preambulum 12
preview-latex program 86
primitive lásd beépített

Q

QuarkXPress program 14–16, 18–20, 24, 25, 29–31, 33

R

RAD 85
rag *lásd* toldalék
reguláris kifejezés 62, 82
report.cls fájl 35
Ruby 85

S

SciWord program 16
Scribus program 14, 17
Scribus.hu program 17
shipout 53
SI 23
slanted lásd döntött
sor
    utolsó 25
soregyen 23
sorozatcímoldal 28
spanish.ldf fájl 88
strukturális egység 19
subtraction sign 20

Sz

szabad szoftver 9, 10, 16, 31, 35, 75, 76, 85
szedő 12
szedőprogram 14
szennycímoldal 28
szerkesztő 12
szerző 11
szinonimaszótár 29
szóelválasztás 18
szoftver
    szabad ~ 16
szóköz 13
szoros sor 51
Szószablya 76
szövegszerkesztő 12

T

táblázat 27
tabulátor 13
tag fájl 88
tárgymutató 28
teTEX program 10, 32, 60, 74
TEX 16
texexec program 33
texlite program 86
TEX live program 32
textcomp csomag 20
text editor lásd szövegszerkesztő
TEXtrace 60
texutil program 33
tezaurusz 29
tfm_ligkernfix.pl program 22
theorem csomag 36, 46
tight line lásd laza sor
Times 21
tipográfia 12
tipográfus 12
tizedesvessző 23
toldalék 19
Tomas Rokicki 85
tompa szedés 25
tördelőszerkesztő 12
Troff program 14
Turing-teljes 81
tükörszélesség 17
typesetting software lásd szedőprogram
Type 1 20, 39, 60
Type 3 60
TypoTEX 16, 39, 85
t1enc csomag 46, 82
T1 fontkódolás 17, 20, 22, 38, 39, 60, 64, 68, 70, 87

U, Ú

ugráló számjegyek 23
ukraineb.ldf fájl 88
Unicode 4, 13, 15, 17, 81, 85
UNIX 10, 60, 62, 74–77
unsrt.bst fájl 89
url csomag 73
utalás 19
UTF-8 16, 60
UTF-16 16, 81
utolsó sor 25

V

varioref csomag 46, 47, 87
verzál 24
vftovp program 50
vi program 13
Vim program 76, 79, 80
vimspell.vim fájl 79, 80
virtuális font 39
Vladimir Volovich 60
vptovf program 50

W

Windows 62
Windows XP 16
Word program 15, 16, 18–20, 24–26, 29–31, 74
word processor lásd dokumentumszerkesztő
WYSIWYG 13–15, 86

X

XEmacs program 16, 76–78
Xindy program 33, 62, 89
XML 33, 83

Y

Yannis Haralambous 85