Mihail Radu Solcan

Ghid LaTeX

versiunea pentru Windows

Bucureşti
2005

Avertisment

Textul acestei cărţi reflectă experienţa autorului. El reflectă, de asemenea, o serie de preferinţe ale autorului.

Este important să înţelegeţi că toate explicaţiile referitoare la folosirea computerului sunt rodul practicii personale a autorului şi oglindesc în mod inevitabil şi limitele acestei practici. Toate soluţiile şi programele descrise în carte au fost testate de către autor, inclusiv pe parcursul elaborării cărţii ca atare. Autorul şi editura nu-şi asumă însă nici un fel de răspundere directă sau indirectă pentru eventuale neajunsuri sau daune care ar putea rezulta din folosirea cărţii sau a programelor de calculator prezentate în carte.

Utilizarea programelor şi a soluţiilor descrise în carte trebuie făcută de cititoare sau cititori pe propriul lor risc.

Sprijin în utilizarea programelor sau aplicarea soluţiilor prezentate în carte nu pot primi, în limita timpului disponibil, decât studentele sau studenţii care elaborează lucrări sub îndrumarea autorului. Orice alte persoane trebuie să ia textul cărţii ca atare, fără posibilitatea de a cere sprijin sau îndrumări în aplicarea soluţiilor sau utilizarea programelor.

În sfârşit, autorul nu se angajează sub nici o formă să asigure sprijin în dezvoltarea de noi programe sau soluţii de aplicare a unor programe pentru nici o categorie de persoane.

Multe denumiri sunt folosite de către firme sau creatorii de programe individuali pentru a distinge produsele lor. Unele dintre ele sunt menţionate în carte, respectând forma proprie denumirii respective. Cititoarele şi cititorii trebuie să ştie că aceste denumiri nu pot fi utilizate de către alte firme sau persoane individuale decât pentru desemnarea produselor respective.

Pentru alte precizări şi informaţii suplimentare consultaţi pe Internet situl <`http://www.ub-filosofie.ro/~solcan/wt`>.

Cuprins

Avertisment iii

1 Creionul electronic **1**
1.1 Sistemul de operare şi managementul fişierelor 1
1.2 Editorul Vim 24
1.3 Expresiile regulate 46
1.4 În căutarea surselor 52
1.5 Corectura computerizată 60

2 Tehnoredactarea computerizată **63**
2.1 LaTeX 64
2.2 BibTeX 114
2.3 Turnul Babel 121
2.4 Tabele şi formule 144
2.5 Indexarea electronică 156

3 Pensula electronică **159**
3.1 Imaginile 159
3.2 Inserarea imaginilor în LaTeX 164
3.3 Inserarea literelor ca inserare de imagini 167

4 Translatorii **171**
4.1 Drumul către HTML 172
4.2 Drumul către RTF 178
4.3 Înapoi către LaTeX 179

Bibliografie **181**

Indice **183**

Capitolul 1

Creionul electronic

Cuprins

1.1 Sistemul de operare şi managementul fişierelor **1**
1.1.1 Maşina virtuală ideală 2
1.1.2 De la fişele tradiţionale la fişierele computerelor 7
1.1.3 Vizualizarea fişierelor 13
1.1.4 Cine se teme de utilizarea computerelor? 15
1.1.5 Principiul surselor deschise 22
1.2 Editorul Vim **24**
1.2.1 Învăţarea Vim într-o zi 25
1.2.2 Vim la modul serios 32
1.3 Expresiile regulate **46**
1.3.1 Definirea şabloanelor 47
1.3.2 Utilizarea şabloanelor 50
1.3.3 Utilizarea şabloanelor cu `grep` 51
1.4 În căutarea surselor **52**
1.4.1 Construirea unei concordanţe 53
1.4.2 Arheologie pe Internet 54
1.5 Corectura computerizată **60**

1.1 Sistemul de operare şi managementul fişierelor

Nu este suficient ca un computer să fie în perfectă stare de funcţionare ca simplu dispozitiv fizic. Fără un sistem de operare este un obiect decorativ. S-ar putea să fie util pentru a presa ceva sau pentru a împiedica accesul pisicii într-un ungher unde nu vreţi ca ea să intre.

Ce este sistemul de operare? Ca şi filosofia, sistemul de operare

nu este uşor de definit. Tanenbaum şi Woodhull spun că sistemul de operare este programul fundamental al computerului.[1] Ce face însă acest program? Am putea să-l vedem ca pe un program care creează o **maşină virtuală**. Aceeaşi maşină fizică devine fie o maşină Windows98 sau WindowsXP, fie o maşină Linux sau altceva. Am putea vedea sistemul de operare şi ca pe un manager al resurselor sistemului.[2]

1.1.1 Maşina virtuală ideală

Care maşină virtuală este mai bună? În ce să-mi transform computerul? Totul depinde de ceea ce vrem să facem cu computerul.

Dacă vrem să aflăm care este structura unui sistem de operare, atunci Minix sau Linux sunt sistemele de operare la care trebuie să apelăm. Sistemul Minix este descris de Tanenbaum şi Woodhull.[3] Prin contrast, sursele Windows sunt secrete. Firma care le realizează îşi protejează astfel drepturile sale de proprietate.

Dacă vrem doar să utilizăm un sistem de operare, atunci alte criterii intră în joc. Achiziţia unei licenţe pentru un sistem Windows98 sau pentru o versiune de casă a sistemului WindowsXP nu este chiar aşa de costisitoare şi costurile totale ale deţinerii unui astfel de sistem sunt probabil sub cele ale unui sistem cu sursele deschise. Multe periferice, precum scannerele sau aparatele foto, se integrează foarte uşor sub Windows. Nu se poate spune acelaşi lucru despre Linux.

Cum cea mai mare parte a publicului care consultă acest ghid lucrează cu un sistem Windows, soluţiile descrise aici sunt cele valabile pe acest sistem.[4] Presupunerea noastră a fost una minimală. Am pornit de la ideea că sistemul utilizat este Windows98. De altfel, dacă nu sunteţi conectate direct la Internet şi prelucraţi doar texte şi imagini nu foarte complicate, sistemul Windows98 este preferabil. Sistemul WindowsXP, cu distincţiile lui între utilizatori obişnuiţi şi administrator, mai mult v-ar încurca.

[1] Tanenbaum şi Woodhull, *Operating Systems* (Upper Saddle River, NJ: Prentice-Hall, 1997), p.1.

[2] Aceste posibilităţi de caracterizare a sistemului de operare sunt evidenţiate de către Tanenbaum şi Woodhull, *op.cit.*, pp.3–5.

[3] În Tanenbaum şi Woodhull, *op.cit.* găsiţi în anexă 27646 de rânduri esenţiale din sursa sistemului de operare Minix. Ca şi-n anexele de faţă, fiecare rând de cod este numerotat. Puteţi face astfel o comparaţie între micile fragmente de cod reproduse aici şi dimensiunile programului care serveşte drept sistem de operare.

[4] Se pare că 95% dintre *PC*-urile din lume funcţionează cu ajutorul Windows.

1.1.1.1 Ce se găseşte în ghidul introductiv Windows

Înainte de a ajunge la fişiere trebuie să ne familiarizăm cu sistemul Windows. Presupun că acesta a fost instalat de către firma de la care aţi cumpărat PC-ul. De asemenea, firma v-a dat un CD autentic cu sistemul şi o carte care vă arată pe scurt cum să vă descurcaţi cu sistemul. Trebuie să ştiţi că fără sistemul de operare computerul este un simplu obiect decorativ sau o achiziţie menită să vă facă să creşteţi în ochii prietenilor.

Dacă n-aţi lucrat niciodată cu un computer, urmaţi sfatul din ghidul introductiv Windows98 şi studiaţi anexa despre utilizarea mouse-ului şi capitolul al 3-lea, cel despre suprafaţa de lucru[5].

În capitolul al 3-lea, fiţi siguri c-aţi înţeles exact rolul butonului Start şi al pictogramelor My Computer şi Network Neighborhood. Utilizarea lui Start este descrisă pe larg.

Esenţial este apoi să înţelegeţi felul în care se creează, se caută şi se deschid dosarele cu fişiere şi fişierele.[6]

Ghidul introductiv Windows98 vă explică, de asemenea, pe scurt ce anume este Internet-ul. Vom presupune, de asemenea, că utilizaţi în mod curent Internet Explorer 6.[7]

1.1.1.1.1 Taste funcţionale şi combinaţii de taste în Windows Unul dintre avantajele interfeţelor grafice este faptul că rolul meniurilor şi butoanelor este explicat chiar de numele lor sau de pictogramele asociate. De multe ori este însă mai uşor de lucrat cu taste şi combinaţii de taste.

Tastele cu utilizare generală le găsiţi listate în finalul ghidului Windows98. Dată fiind importanţa lor, vom descrie şi noi câteva taste.

meniuri

F10 serveşte la activarea meniurilor. Observaţi, de asemenea, că – în meniuri – literele subliniate indică tasta pe care puteţi apăsa pentru a lansa o comandă.

copiere

CTRL+C serveşte la copierea unei porţiuni selectate dintr-un fişier. Fiţi sigure c-aţi înţeles ce înseamnă selecţie înainte de a utiliza combinaţiile care urmează.

decupare

CTRL+X serveşte la decuparea unei porţiuni selectate. Nu trebuie să vă inducă în eroare dispariţia de pe ecran a porţiunii selec-

[5] În englezeşte, *desktop*.

[6] Termenul englezesc pentru „dosar“ este *folder*. Cel pentru „fişier“ este *file*.

[7] Dacă versiunea preinstalată este 5, puteţi aduce lesne Internet Explorer la o versiune superioară.

tate. Ea este disponibilă pentru inserare în alt punct, în momentul următor.

inserare Ctrl+v serveşte la inserarea unei porţiuni copiate sau decupate, în punctul în care se află cursorul.

anulare Ctrl+z serveşte la anularea acţiunii pe care aţi întreprins-o în momentul anterior. Este extrem de util să ştiţi de existenţa ei, din motive lesne de înţeles.

Acestea sunt combinaţii de taste valabile în general în programele Windows (în ferestrele pe care le deschide Windows).

Ca să vă mişcaţi între ferestre vă puteţi sluji de Alt+Tab. Evident, clicurile cu mouse-ul vă permit să faceţi acelaşi lucru.

Mouse-ul nu este întotdeauna o unealtă cu care poţi lucra precis. Comparaţi selecţia cu ajutorul mouse-ului şi selecţia pe care o puteţi face ţinând Shift apăsat şi lucrând cu săgeţile sau Pg Up, Pg Dn.

1.1.1.1.2 Taste speciale Windows Dacă aveţi o tastatură specială Windows, atunci găsiţi pe rândul de jos al tastaturii două taste speciale Windows.

Tasta din stânga, cea cu steagul Windows pe ea, dacă este apăsată, derulează meniul de start.

Tasta din partea dreaptă, cea cu o foaie şi săgeată pe ea, este poate mai rar observată, dar poate fi extrem de utilă. Selectaţi pictograma My Computer de pe suprafaţa de lucru. Apăsaţi tasta respectivă. Apare un meniu.

meniul contextual

Acelaşi meniu apare şi la un clic pe butonul dreapta al mouse-ului. Acest meniu se numeşte „meniu contextual“.

Meniurile contextuale sunt deosebit de importante şi fără a fi familiarizată sau familiarizat cu ele practic nu poţi lucra eficient în Windows. De ce? Un fişier poate fi deschis de către mai multe programe. Ai nevoie de meniul contextual pentru a avea acces rapid la programele care prelucrează fişierul respectiv. De asemenea, meniul contextual vă oferă acces la proprietăţile obiectelor cu care sistemul de operare a populat computerul dumneavoastră.

1.1.1.1.3 Amenajarea suprafeţei de lucru Ghidul primit o dată cu achiziţionarea Windows98 vă oferă unele sugestii referitoare la amenajarea suprafeţei de lucru.

imaginea de pe suprafaţa de lucru

Cu puţin noroc puteţi înfrumuseţa lesne suprafaţa de lucru. Căutaţi pe Internet un situl <`http://www.webshots.com`>. Descărcaţi de acolo una sau mai multe dintre colecţiile de imagini pentru su-

prafaţa de lucru. Instrucţiunile de instalare le găsiţi pe sit. Puteţi adăuga şi propriile imagini.

1.1.1.2 Partiţionarea discului dur şi segmentarea sistemului

S-ar putea ca multora această secţiune să li se pară vrăjitorească. În realitate, cred că este bine ca să vă gândiţi la cele scrise aici încă din prima fază a amenajării sistemului dumneavoastră de calcul.

Aş putea adăuga că vă puteţi gândi la aceste lucruri chiar atunci când achiziţionaţi componentele sistemului. De ce? Pentru că puteţi primi nu doar CD-ul cu sistemul Windows, ci şi multe alte programe utile care sunt, cum se spune în engleză, *bundled* (ataşate) unor componente. De pildă, o placă de baza poate avea ataşat un program de partiţionare a discului dur. Programele acestea ataşate le puteţi folosi în condiţii perfect legale doar împreună cu echipamentul procurat.

Vă întrebaţi probabil la ce v-ar folosi partiţionarea discului dur? Ei bine, dacă aveţi un singur disc dur, programul de partiţionare va crea mai multe discuri virtuale. Este ca şi cum aţi avea mai multe discuri în computer.

Pot surveni accidente care s-ar putea să vă determine să ştergeţi sistemul Windows şi să-l reinstalaţi de la zero. În asemenea condiţii este mult mai sigur să aveţi sistemul de operare pe un disc separat. Citiţi însă şi recitiţi instrucţiunile programului de partiţionare. Puteţi nu doar să distrugeţi date preţioase, dar chiar să faceţi ceva care ar duce la nerecunoaşterea discului dur de către sistem.

partiţiile

Avariile teribile sunt mai mult o chestiune de principiu. Practica de zi cu zi ne furnizează alte motive pentru a partiţiona discul dur. Putem avea, de exemplu, trei discuri virtuale. Discul `C` este dedicat sistemului Windows şi programelor care sunt ataşate diverselor componente. Pe discul `D` sunt puse programele cu sursă deschisă.[8] De asemenea, aici sunt adăpostite datele importante. Discul `E` este folosit pentru stocarea temporară de date şi testarea de programe.

calea

Discurile `D` şi `E` trebuie să aibă însă şi un sistem de dosare judicios organizat. Dosarele create de mine au nume scurte. De regulă, două-trei litere sugestive sunt uşor de ţinut minte şi este uşor de tastat ceea se numeşte tehnic „calea“ către un fişier (în engleză termenul tehnic este `path`). Un exemplu de genul `D:\use\bin\` ne arată imediat cum desemnează Windows căile: mai întâi este indicat discul,

[8] Pentru ideea de sursă deschisă v. §1.1.5.

apoi se pune bara oblică inversă şi numele unui dosar şi aşa mai departe.

căi în Unix

Folosirea \ în numele de căi este exact opusă celei din sistemele Unix, unde se utilizează în acelaşi scop /. Încurcăturile create sunt numeroase, dar trebuie să învăţăm să ne descurcăm. De asemenea, sistemele Unix nu folosesc conceptul de disc. Totul este organizat pe un singur arbore, a cărui rădăcină este notată cu /. Dosarele sunt organizate foarte disciplinat. Dosarele `bin`, de pildă, sunt strict pentru programe executabile.

nu daţi dosarelor nume cu spaţii

Pentru o bună compatibilitate cu programele din lumea Unix nu daţi, de asemenea, nume de dosare cu spaţii. Folosiţi spaţiul subliniat _ acolo unde vreţi să introduceţi un spaţiu. Un nume de dosar care are spaţii, dacă apare în calea invocată de către un program, va crea confuzii în sistem.[9]

În rezumat, reorganizarea şi segmentarea sistemului au şi scopul de pregăti condiţiile pentru un bun transfer al programelor din lumea Unix. Aveţi astfel la dispoziţie şi sistemul Windows, cu posibilităţile lui de a manevra sofisticat interfeţele grafice, şi elemente ale sistemului Unix, mai orientat către programare. Calculatorul se transformă într-o veritabilă staţie de lucru.

adăpostiţi periodic fişierele preţioase

1.1.1.2.1 Copiile de siguranţă Dacă aveţi un fişier sau un folder important, atunci este bine să-l salvaţi periodic într-un loc sigur. Din acest punct de vedere, partiţionarea discului dur joacă un rol important. Chiar dacă ar trebui să reinstalaţi de la zero sistemul de operare, aveţi un spaţiu pe care să adăpostiţi fişierele preţioase.

Inscripţionarea de CD-uri este o altă soluţie. Un CD poate adăposti o mare cantitate de date.

Dacă nu dispuneţi de un inscriptor de CD, atunci un program care poate să împartă un fişier mare în bucăţi care încap pe o dischetă este foarte util. Veţi găsi cu siguranţă că programul `Splitfile` al lui Magnus Nilsson[10] joacă bine acest rol.

splitfile

`Splitfile` ocupă un loc foarte mic pe o dischetă. Nu depindeţi de nici un fel de program instalat pe un computer sau altul.[11] Singura precauţie de care trebuie să ţineţi cont este aceea de a nu încerca

[9] Puteţi păţi acest lucru şi când treceţi un fişier din Windows98 în Windows2000, nu doar când apelaţi programe Unix sub Windows.

[10] Programul Splitfile nu pare să aibă o pagină de web proprie. Ultima dată (12/01/2004) l-am găsit la adresa <`http://biphome.spray.se/mason40/index.html`>. Trebuie căutat şi cu ajutorul cuvintelor-cheie.

[11] Versiunea 1.4.1.12 a 2xExplorer dispune şi ea de un `Split file` intern. Nu uitaţi însă că vă trebuie 2xExplorer şi pe computerul pe care vreţi să refaceţi

să umpleţi la maximum o dischetă. Este mai prudent să indicaţi o valoare mai mică de `1.44`MB pentru bucăţile de fişier.

O alternativă, pe care am testat-o mai puţin, dar care pare o soluţie excelentă, este programul SplitFile creat de către Jitendra Garodia.[12] Programul are mai multe posibilităţi decât cel al lui Nilsson. De asemenea, formatul fişierelor create este compatibil cu cel din 2xExplorer.[13] SplitFile creează o arhivă desfăcută în mai multe fişiere; cu această arhivă poate lucra şi 7zip.[14]

În rezumat, morala principală este foarte simplă: nu lucraţi fără a face, în prealabil, o copie a fişierului prelucrat, fie pe alt disc dur, fie pe alt calculator, fie pe un CD, fie pe dischete. Puneţi periodic la adăpost fişierele cu date preţioase. În orice caz, salvaţi din când în când fişierele pe discul dur[15]

1.1.2 De la fişele tradiţionale la fişierele computerelor

Munca intelectuală tradiţională stă sub semnul munţilor de fişe. Era nevoie de zeci de sertare, de tot felul de cutii pentru a stoca fişele. Aveai nevoie de fişe pentru a-ţi consemna ideile proprii, pentru a sistematiza ideile altora, pentru a crea liste bibliografice, note şi pentru a indexa cărţi.Doar manuscrisul ca atare era aşternut pe coli de scris de format A4.

fişierele electronice ca o colecţie de fişe tradiţionale

Deşi este doar o analogie, asemuirea unui fişier cu o colecţie de fişe este cât se poate de utilă. Ea ne ajută să facem legătura între munca intelectuală de factură tradiţională şi cea efectuată cu ajutorul calculatorului.

Fişierele cele mai importante pentru noi vor fi cele zise „fişiere plate“. În esenţă, acestea sunt fişiere de tip text, în care sunt introduse eventual anumite marcaje structurale, dar care pot fi citite aidoma unor texte.

Dacă vă gândiţi la fişele pe care vă notaţi ideile, atunci puteţi face

fişierul decupat în bucăţi! De asemenea, formatul folosit de 2xExplorer nu este compatibil cu cel al lui Nilsson.

[12] Pagina sa de web este la <`http://jituonline.freeservers.com`>.

[13] Atenţie doar la faptul că arhivele generate de 2xExplorer trebuie redenumite. Extensia `.1` trebuie transformată în `.001` şi aşa mai departe, conform extensiilor produse de SplitFile sau 7zip.

[14] Arhiva este compatibilă cu cea generată de către `split.exe` din setul de unelte Unix, în versiunea GnuWin32. Acesta este un program cu sursă deschisă, foarte cunoscut şi bine testat. Este mai greu de folosit însă de către începători.

[15] Căderile de curent pot fi inamicul cel mai periculos. Dacă mâine trebuie să predaţi eseul şi sunteţi aproape gata, o cădere de curent poate să compromită totul, în cazul când n-aţi fost prudentă sau prudent.

o analogie între ele şi rândurile dintr-un fişier de tip text. Analogia aceasta nu este lipsită de ambiguităţi,[16] dar este utilă.

Ca lucrurile să fie mai încurcate decât par, sfârşitul de rând în sens logic nu este marcat la fel pe cele trei platforme importante(Windows, Unix şi Mac). Într-un fişier de calculator totul are un cod. Dacă am crea un fişier `woo.txt` în Windows sau în predecesorul acestuia, MS-DOS, am vedea că sfârşitul de rând este marcat prin două coduri. Numele convenţional al acestora este CR LF. Un fişier analog, `uoo.txt`, creat sub Unix, are marcat sfârşitul de rând logic cu LF[17]. Iar fişierul analog, `moo.txt`, creat sub Mac, are marcat sfârşitul de rând logic cu CR[18].

CR LF

LF

CR

1.1.2.1 Managementul fişierelor

Mai importantă decât înfrumuseţarea suprafeţei de lucru este extinderea posibilităţilor de a explora discul dur al computerului. Sub Windows98 şi WindowsXP există un program special destinat explorării fişierelor şi pe care-l găsim în meniul contextual ataşat lui My Computer dacă apăsăm pe Open sau Explore.

Programul de explorare a fişierelor este setat automat la instalare după o formulă cam paternalistă. Cea mai utilă schimbare mi se pare a fi afişarea extensiei tuturor numelor de fişiere. Windows foloseşte extensia (literele puse după un punct în numele fişierului pentru a determina tipul acestuia şi acţiunile asociate). Ca să vedem toate extensiile trebuie să parcurgem ruta Start→ Settings→ Folder Options.... În Folder Options apăsăm pe View şi deselectăm Hide file extensions for known file types.

extensiile fişierelor

WindowsXP include Folder Options... în `Control Panel`.

Ucenicii vrăjitori nu vor rezista desigur tentaţiei de a vedea fişierele ascunse ale sistemului şi vor selecta opţiunea Show all files. Atenţie însă la modificările pe care le faceţi în fişierele care vă sunt dezvăluite prin această manevră.

1.1.2.1.1 2xExplorer Programul de explorare inclus în Windows este oarecum limitat. Ar trebuie să aveţi un veritabil manager de fişiere.

Programul care mie mi se pare cel mai potrivit pentru acest scop

[16] V. distincţia dintre rânduri logice şi rânduri vizuale în §1.2.1.2.

[17] De la expresia englezească *line feed*.

[18] De la expresia englezească *carriage return*, care avea un sens evident pe vremea maşinilor de scris.

este 2xExplorer. Este scris de Nikos Bozinis.[19] Instalarea programului lui Bozinis este extrem de simplă,[20] dar în faza în care presupun că vă aflaţi acum s-ar putea să vă vină mai greu să faceţi acest lucru sub Windows98.[21] WindowsXP dispune de un program de dezarhivare aşa că ajunge să indicaţi un dosar în care să fie despachetate fişierele programului. Ca să-l porniţi comod puteţi crea o scurtătură pe suprafaţa de lucru.

De îndată ce aţi făcut dublu clic pe scurtătura nou creată apare o fereastră divizată în trei panouri. Eu am închis însă panoul din stânga apăsând semnul × din colţul din dreapta sus. Am tras, de asemenea, bara cu unelte în partea de sus. Oricum, aceste lucruri sunt mai puţin importante în sine. Fiecare poate aranja cum vrea aspectul ferestrei în care apare 2xExplorer.

La ce este bun 2xExplorer sau un program similar? Uşurează enorm managementul dosarelor şi fişierelor. Am văzut mulţi utilizatori care pun totul pe suprafaţa de lucru, în dosarul `My Documents`, creat automat la instalarea Windows, sau care plasează fişierele direct în rădăcina discului `C`. Aceste obiceiuri denotă faptul că nu au făcut un efort pentru a înţelege rolul managementului fişierelor. În orice caz, nici să nu vă gândiţi să folosiţi uneltele transferate din Unix fără managementul adecvat al fişierelor. Riscaţi să produceţi sute de fişiere în acelaşi dosar, printre care vă va fi absolut imposibil să vă descurcaţi.

1.1.2.1.1.1 Crearea de dosare şi fişiere cu 2xExplorer

Pentru început deschideţi 2xExplorer şi verificaţi dacă înţelegeţi cum puteţi crea dosare şi fişiere. Navigaţi către un loc care nu este periculos.[22] Apăsaţi F8 şi veţi vedea cum apare un dosar pe care scrie `New Folder`. Schimbaţi-i numele în „Test“.[23] În 2xExplorer exersaţi acum clicul simplu (apăsarea) pe butonul din stânga al mouse-ului, cu săgeata îndreptată către un dosar. Veţi vedea cum se schimbă culoarea pictogramei dosarului. Dacă apăsaţi Enter pătrundeţi în dosarul respectiv. Cu Back Space reveniţi.

F8

Intraţi acum în dosarul nou creat. Creaţi alte câteva dosare.

[19] Îl puteţi găsi la adresa <`http://www.netez.com/2xExplorer`>.

[20] Versiunea 1.4.1.12 dispune şi de un instalator. Eu am folosit însă, în continuare, dezarhivarea, copierea fişierelor şi generarea de scurtături.

[21] Citiţi secţiunea 1.1.2.2 dacă nu ştiţi să lucraţi cu arhive (cu fişiere comprimate).

[22] Altundeva decât în dosarul Windows.

[23] Evident, puteţi folosi oricare nume care vă convine. Recomandarea ar fi totuşi să nu utilizaţi nume cu spaţii. De pildă, în loc de „New Folder“ folosiţi „New_Folder“. V. şi pagina 6, rândul 11.

F7

Intraţi într-unul dintre aceste dosare. Apăsaţi F7.[24] Apare o pictogramă lângă care scrie `New File.txt`. Schimbaţi numele ca şi-n cazul dosarului.

Ce aţi creat? Un fişier zis „de tip text“. Dacă daţi un dublu clic pe un fişier de tip text şi aveţi o instalaţie Windows nemodificată, atunci se deschide fereastra unei aplicaţii numită Notepad. În traducere, numele programului însemnă *carnetul de notiţe*. Este chiar ceea ce sugerează pictograma acestui program.

Notepad

Este bine să învăţaţi folosirea Notepad explorând pe meniurile sale. Presupun c-aţi deprins folosirea în special a comenzilor *Copy, Cut, Paste*. Acest lucru este desigur imposibil fără să ştiţi să selectaţi porţiuni dintr-un text. Exersaţi, de asemenea, copierea de text între diverse instanţe ale aplicaţiei Notepad.

1.1.2.1.1.2 Operaţii cu fişiere Fişierele pot fi copiate (sub alt nume, chiar şi-n acelaşi dosar!), pot fi mutate sau şterse. Aceste operaţii ar trebui să fie evidente pentru oricine s-a familiarizat cu Windows. 2xExplorer oferă o bară cu pictograme speciale pe care puteţi da un clic atunci când vreţi să executaţi astfel de operaţii. Funcţionează şi meniurile contextuale Windows.

1.1.2.1.1.3 Filtre Creaţi un dosar. Creaţi în el mai multe fişiere, unele cu extensia `txt`, altele cu `tex` (nu uitaţi punctul dinaintea extensiei!).

Mergeţi acum, în 2xExplorer pe ruta Files→ Filters.... Apăsaţi pe tasta cu săgeata către dreapta. Puneţi un punct şi adăugaţi extensia `tex`. Apăsaţi ENTER. Sunt afişate acum numai fişierele de tip `tex`.

1.1.2.1.1.4 Proprietăţile fişierelor şi dosarelor În 2xExplorer selectaţi un fişier sau un dosar.[25] Apăsaţi tasta F12! Apare o fereastră cu proprietăţile fişierului sau dosarului.

F12

Uitaţi-vă mai ales la Attributes. De multe ori vrem să bifăm sau să debifăm Read-only. După cum sugerează şi numele englezesc al

[24] Versiunea mai nouă a 2xExplorer include şi un meniu special `Actions`. Una dintre **acţiuni** este chiar crearea de fişiere.

[25] Studiaţi felul în care sunt setate clicurile pe sistemul dumneavoastră. Un clic pe pictograma din stânga numelui este mai prudent. Clicul pe nume sau un dublu clic rapid pe nume s-ar putea să deschidă o mică fereastră pentru operarea de schimbări în nume.

acestei proprietăţi, fişierul read-only poate fi doar citit. Dacă vrem să-l schimbăm, trebuie să eliminăm această proprietate.

1.1.2.1.1.5 Selectarea unui grup de fişiere O operaţiune asemănătoare cu filtrarea este selectarea unui grup de fişiere. Mergeţi în meniul principal la Mark. Alegeţi Select Group....

Acum trebuie să învăţaţi doar să construiţi tipare foarte simple. Semnul * ţine locul unui grup de litere. Semnul ? ţine locul unei singure litere. Dacă scrieţi *.tex, atunci vor fi selectate toate fişierele de tip tex.

tipare simple

1.1.2.1.1.6 Informaţii despre un dosar cu fişiere În meniul principal daţi clic pe Tools şi apoi pe Folder Data.... Veţi obţine date despre dosarul respectiv.

La ce sunt bune aceste date? Cea mai simplă utilizare este legată de folosirea spaţiului de pe discul dur. De pildă, pe computerul pe care scriu, dosarul cu complexul de programe TEX ocupă 653.5MB. Pot însă vedea şi arborele subdosarelor din dosarul respectiv, cu statistica numărului de dosare şi de fişiere din ele.

Datele astfel obţinute pot fi păstrate într-un fişier de tip text. Pentru aceasta folosiţi meniul contextual al ferestrei cu date despre dosare. Daţi un clic pe Print. În ciuda numelui, aveţi posibilitatea să creaţi un fişier. Puteţi face acest lucru şi pentru un CD-ROM.

Păstraţi documentele descărcate de pe Internet în dosare care au ca nume data descărcării. În aceste dosare grupaţi fişierele în dosare cu nume semnificative. Adăugaţi, dacă este necesar, într-un fişier special adresa de Internet şi date cu privire la locul de unde aţi descărcat documentele. Creaţi cataloage cu aceste dosare după metoda descrisă mai sus. În acest fel nu veţi avea probleme cu citarea documentelor de pe Internet.

cataloage cu documente descărcate de pe Internet

1.1.2.1.1.7 Căutarea de fişiere O operaţie pe care o întreprindem adesea este căutarea de fişiere. Mergeţi la Tools în 2xExplorer. Căutaţi un fişier al cărui nume nu-l ştiţi. Puneţi doar *.*, semn că nu ştiţi nici numele, nici extensia. Ştiţi însă un fragment de text din fişier. Introduceţi fragmentul de text respectiv în caseta de dialog, după ce aţi bifat use text constraints.

Căutarea pe care o oferă 2xExplorer nu este mult mai sofisticată decât cea pe care aţi găsi-o la Start→ Find în Windows, dar o puteţi efectua mai uşor, direct în dosarul care vă interesează.

1.1.2.1.1.8 Aranjarea meniului de start Activaţi panoul din stânga al 2xExplorer-ului. Urmaţi ruta Bookmarks→ Go to folder→Start Menu şi daţi clic pe Start Menu. Creaţi acum dosare în care grupaţi scurtături către programele pe care le-aţi instalat.

Puteţi crea noi scurtături într-un mod cât se poate de simplu. În panoul din dreapta mergeţi la un dosar unde aveţi un executabil Windows. Selectaţi-l! Mergeţi la Edit şi copiaţi (CTRL+C). Treceţi apoi în panoul din stânga. Mergeţi la Edit şi apăsaţi `Paste Shortcut`. Acum puteţi redenumi scurtătura.

Nu uitaţi sfatul de a folosi un dosar şi programe destinate doar testării până sunteţi sigure şi siguri pe ceea ce faceţi.

1.1.2.2 Managementul Fişierelor Comprimate

Chiar dacă n-aveţi o idee prea exactă despre fişiere vă puteţi da seama că ele ocupă un loc în mediile de stocare. Arhivarea este este un proces prin care fişierele sunt comprimate. Mai mult decât atât, o colecţie de fişiere este strânsă la un loc şi circulă sub forma unui pachet unic.

WinXP dispune de un sistem de dezarhivare propriu. Lucrul acesta nu este însă adevărat pentru mai vechiul Win98. Pe Internet sau pe CD-urile ataşate revistelor dedicate lumii calculatoarelor veţi găsi lesne destul de multe variante gratuite de programe de arhivare.

7-zip

O soluţie care are numeroase avantaje o reprezintă programul `7-zip`, creat de Igor Pavlov. Pagina de web a acestui program este <`http://www.7-zip.org`>. Consultaţi, de asemenea, <`http://sf.net`>.

`7-zip` are propriul manager de fişiere. Cu ajutorul acestui manager puteţi căuta în interiorul unui pachet de fişiere.

De asemenea, vă puteţi sluji de comenzile pe care programul le instalează în meniul ataşat oricărui fişier (meniul care apare când faceţi clic pe dreapta pe pictograma fişierului).

O problemă este lipsa unui dezarhivator pentru arhivele specifice Mac. O soluţie este Aladdin Expander al firmei AladdinSystems. În acest caz, citiţi cu atenţie licenţa produsului. Acest lucru este valabil, evident, pentru toate programele pe care le vom menţiona aici. În special fiţi atente şi atenţi la faptul că n-aveţi dreptul să obţineţi un profit comercial în urma distribuirii produsului.[26]

[26] Cu alte cuvinte, dacă aţi descărcat de pe Internet acest dezarhivator sau l-aţi obţinut în alt mod nu-l puteţi vinde altcuiva. Evident, puteţi să-l daţi mai

1.1.3 Vizualizarea fişierelor

Probabil v-aţi familiarizat deja cu sistemul de extensii al fişierelor în Windows. Extensia este grupul de litere care urmează după punctul pus la sfârşitul numelui fişierului.

În 2xExplorer puteţi vizualiza un fişier dacă-l selectaţi şi apăsaţi apoi tasta F3. Evident, acesta este un mod foarte limitat de a vizualiza fişiere şi vă va fi de folos doar pentru cele care conţin doar texte simple. Pentru alte fişiere aveţi nevoie de programe de vizualizare dedicate unui anumit tip de fişier.

1.1.3.1 Vizualizarea fişierelor PostScript

Pe parcursul acestui ghid avem mare nevoie de fişiere cu extensia `ps`; denumirea completă a acestor fişiere este „fişiere PostScript“.

Pentru a vizualiza comod un fişier PostScript sub Windows aveţi nevoie de două tipuri de programe. Primul este un interpretor al conţinutului fişierului.[27] Al doilea este programul de vizualizare ca atare.

Descărcaţi de pe Internet unul dintre fişierele `gs700w32.exe`, `gs704w32.exe`, `gs800w32.exe` sau ceva mai nou din această serie. Citiţi eventuale instrucţiuni de instalare. Oricum, procesul de instalare este extrem de simplu. Un fişier `exe` este executabil direct în Windows. După ce aţi plasat convenabil fişierele rezultate mergeţi la dosarul doc şi citiţi fişierul `PUBLIC`. El conţine **Aladdin Free Public License**, pe scurt AFPL. Comparaţi-o pe loc sau ulterior cu importanta licenţă GNU a Free Software Foundation.

Ghostscript

Pentru început n-are sens să încercaţi să faceţi mare lucru cu programul pe care l-aţi instalat. Este doar bine să ştiţi un se află, care este versiunea sa şi că se numeşte Ghostscript.

Pentru a instala vizualizatorul descărcaţi un fişier cu un nume de genul `gsv44w32.exe`. Îl găsiţi în multe locuri pe Internet. Primul număr indică versiunea. Aveţi grijă să fie compatibilă cu versiunea programului Ghostscript. Din nou, instalarea este relativ simplă. Citiţi însă instrucţiunile. Va trebui să indicaţi însă ce versiune a Ghostscript vreţi să folosiţi. Programul pe care tocmai l-aţi instalat se numeşte Ghostview. Este foarte folosit în lumea academică,

Ghostview

departe, dar nu puteţi cere o compensaţie financiară decât pentru mediul de stocare sau pentru alte cheltuieli implicate de transferul produsului.

[27] PostScript este un limbaj creat de firma Adobe pentru tipărirea de documente pe printere cu laser. Limbajul poate fi folosit şi pentru crearea de imagini pe ecran. Pentru o scurtă prezentare a limbajului PostScript v. §3.1.2.

dar are o licenţă, ca să zic aşa, „la limita“ programelor care vor fi recomandate în această carte. Pentru a face să dispară o fereastră care tot apare când deschideţi programul[28] trebuie să înregistraţi programul contra unei sume de bani. Versiunea neînregistrată este însă absolut funcţională şi nu cuprinde nici un fel de limitări.

pstoedit Versiunile mai noi ale Ghostview nu integrează însă o unealtă extrem de necesară denumită „pstoedit“. Despre pstoedit puteţi citi la <`http://www.pstoedit.net/pstoedit/`>. Concret, pentru instalare, vă trebuie fişierul `pstoeditsetup.exe` şi, desigur, priceperea minimă necesară pentru a instala programe sub Win98.[29]

Ce face pstoedit? Vă oferă posibilitatea de a converti un fişier PostScript într-unul de tip `pdf`.

1.1.3.2 Vizualizarea fişierelor pdf

`Pdf` este o prescurtare a unui termen englezesc care ne spune că este vorba despre fişiere care stochează documente într-un format portabil. Portabilitatea înseamnă, între altele, că aceste fişiere pot fi văzute atât sub Windows, cât şi sub Unix şi Macintosh, cele trei platforme principale în lumea calculatoarelor.

Acrobat Reader Tot atât de importantă ca şi portabilitatea este calitatea fişierelor `pdf` de a fi veritabile cărţi electronice. Pentru a vedea aceste cărţi trebuie să instalaţi un sistem de vizualizare. Cel mai faimos este Adobe Acrobat Reader. Citiţi despre acest program la adresa <`http://www.adobe.com/products/acrobat`>. Ca mai toate programele de vizualizare, Acrobat Reader este distribuit gratuit.

Pentru a avea pe deplin senzaţia cărţii electronice puteţi instala Adobe Acrobat eBook Reader.

1.1.3.3 Vizualizarea altor tipuri de fişiere

Programele de vizualizare a fişierelor, spre deosebire de cele de editare, sunt de regulă oferite gratuit de către firme.

Word Viewer În mod sigur veţi avea nevoie periodic de un vizualizator de fişiere `doc`. Microsoft Word Viewer poate fi folosit în mod gratuit. Puteţi vedea cu el şi fişierele de tip `rtf`.

Open-Office OpenOffice este o suită completă de aplicaţii pentru birou; poate

[28] Acestui tip de fereastră i se spune în engleză *nagscreen*.

[29] Între altele, instalarea sub WinXP este mai complicată. Trebuie să studiaţi în mod special cum se pot instala programe sub asemenea sistem de operare.

fi folosită atât pentru vizualizarea, cât şi pentru editarea de fişiere `doc` şi `rtf`.[30]

Pe Internet veţi găsi, de asemenea, fişiere de tip `lit`. Pentru a le vedea aveţi nevoie de Microsoft Reader.[31]

Microsoft Reader

Un alt tip de fişiere pe care s-ar putea să le întâlniţi pe Internet sunt cele pentru **palm computere**. Există cărţi întregi în format `pdb`. Un vizualizator gratuit pe care l-am folosit cu succes este DocReader-ul creat de Mike Pickering.[32]

DocReader

Evident, pentru omniprezentele, pe Internet, fişiere `html`, este nevoie de un vizualizator pentru asemenea fişiere. De regulă, Microsoft Internet Explorer sau Netscape Navigator sunt folosite pentru a vizualiza asemenea fişiere.

Internet Explorer şi/sau Netscape Navigator

1.1.4 Cine se teme de utilizarea computerelor?

Pentru a adopta perspectiva programatorului trebuie să ştim ceva mai mult despre utilizarea Windows.

Mergeţi la Start→ Run. Tastaţi cuvântul `command` şi apoi apăsaţi ENTER. Apare o fereastră neagră ca aceea din vechiul MS-DOS. În această fereastră nu pot fi date decât comenzile numite „comenzi în linie“. Dacă nu ştiaţi, tocmai aţi dat o asemenea comandă pentru a chema fereastra MS-DOS.

`command`

Reveniţi la Start→ Run. Apăsaţi butonul Browse. Navigaţi un pic până găsiţi pe discul C dosarul Windows. Apoi căutaţi `NOTEPAD.EXE` şi când îl găsiţi daţi un clic pentru selecţie şi apăsaţi butonul Open. Aranjaţi fereastra neagră a MS-DOS prompt-ului în aşa fel încât să vedeţi mica fereastră Run deschisă în colţul din stânga-jos al ecranului. Mergeţi în fereastra neagră a MS-DOS. Tastaţi cu grijă comanda din fereastra Run. Verificaţi dacă înţelegeţi conceptul de cursor[33] Puteţi recurge şi la procedura de copiere şi inserare de text dintr-o fereastră într-alta. După ce aţi scris sau copiat textul relativ misterios, apăsaţi tasta ENTER. Se va deschide o fereastră Notepad.[34]

exemplu de comandă în linie

Ce mai putem scrie în fereastra neagră? Tastaţi `echo Un text`.

[30] Puteţi să vă procuraţi OpenOffice de la adresa <`http://www.openoffice.org`>.

[31] Puteţi descărca programul de instalare de pe situl firmei la <`http://www.microsoft.com`>.

[32] Îl puteţi descărca de la <`http://www.crosswinds.net/~mpicker0`>.

[33] O liniuţă clipeşte în fereastra neagră atunci când aceasta este fereastra în care lucraţi. Această liniuţă de subliniere indică punctul în care se află cursorul. Este punctul în care, practic, puteţi insera un text.

[34] Am făcut acelaşi test şi-n WindowsXP. Nu au fost probleme. Copierea în

Programul va afişa cuminte `Un text`. Cuvântul `echo` este o comandă, care spune programului să afişeze textul care urmează imediat după comanda respectivă.

istoria comenzilor

Putem influenţa ceea ce se întâmplă la prompt-ul MS-DOS? Da. Merită să dispunem de o istorie a comenzilor. Mergeţi pe discul C chiar în rădăcină şi plasaţi săgeata cursorului Windows pe AUTOEXEC.BAT. Faceţi clic pe dreapta. Faceţi apoi clic pe `Edit`. Fişierul acesta este un fişier de tip text, dar al cărui conţinut este, pentru sistemul de operare, un şir de comenzi. Adăugaţi la sfârşitul fişierului comanda `c:\windows\command\doskey.com`. Verificaţi dacă efectiv aveţi pe calea indicată un fişier cu numele `doskey.com`. Reporniţi calculatorul.[35]

Când sistemul a repornit deschideţi din nou o fereastră MS-DOS. Daţi ce comenzi vreţi în ea. Apăsaţi apoi tasta cu o săgeată în sus şi pe cea cu săgeata în jos. Observaţi că acum sistemul ţine minte comenzile.[36] Cu tastele cu săgeţi către stânga şi către dreapta vă puteţi mişca prin textul comenzii. Folosiţi tasta `Ins`[37] pentru a modifica o comandă fără a o scrie de la capăt.

Aţi dat deja peste un fel de resursă secretă şi extrem de utilă a sistemului MS-DOS.

1.1.4.1 Comenzile în linie din 2xExplorer

În multe situaţii s-ar putea să aveţi nevoie să daţi rapid o comandă în linie. Putem folosi pentru a atinge acest scop şi 2xExplorer-ul.

Run Command

Cea mai simplă comandă în linie este formată din numele unui program executabil. Mergeţi la Tools şi daţi clic pe Run Command. Tastaţi `sfc` şi apoi ENTER. Apare unul dintre cele mai utile programe din Windows98. Acest program verifică integritatea sistemului. n-are nici un rost să reinstalaţi sistemul de câte ori aveţi o pană. Îl puteţi restaura cu `sfc`.

Dac-aţi pus `splitfile` undeva în PATH, atunci puteţi repeta comanda de mai sus cu `splitfile`.

Comenzile în linie de mai sus sunt utile pentru a chema programe

fereastra neagră diferă desigur. Practic trebuie recurs la meniul obţinut prin clic pe butonul din dreapta al mouse-ului.

[35] Windows vă va sâcâi de multe ori cu aceste reporniri după instalări sau modificări în sistem.

[36] Le ţine minte cât timp fereastra este deschisă.

[37] Tasta care alternează două moduri de scriere: unul prin inserarea de text; altul în care textul este suprascris.

Windows. Tastaţi însă `dir` urmat de apăsarea pe tasta `Enter`. Apare doar preţ de o clipă o fereastră neagră. N-aţi văzut nimic.

`Dir` este o comandă în linie care trebuie dată într-o fereastră MS-DOS. Apăsaţi tasta F10 urmată de `Enter`. Apare o fereastră neagră (cel puţin aceasta este culoarea setată automat). Acum `dir` are alt efect. **F10**

Avantajul oferit de `2xExplorer` poate din nou să pară minor. Putem chema fereastra MS-DOS direct din sistemul Windows. Diferenţa este însă dată de faptul că ea are promptul chiar în dosarul în care am chemat-o. În plus, 2xExplorer are propria sa istorie a comenzilor. Avantajul nu este chiar mic.

1.1.4.2 Fişiere cu comenzi în 2xExplorer

O comandă în linie este un text ca oricare altul. De ce n-am putea crea un fişier cu asemenea comenzi?

Mergeţi în dosarul unde aţi creat (cu titlu de probă!) fişiere cu extensia `tex`. Selectaţi-le! Mergeţi în meniul principal la Mark şi daţi un clic pe Generate batch.... Tastaţi comanda `ren $N $B.txt`. Daţi clic pe `Create`. Salvaţi fişierul cu extensia `bat`. Daţi comanda de executare a acestui fişier. **Generate batch...** **redenumirea fişierelor**

Selectaţi fişierul `bat` creat şi apăsaţi F3. Nu executaţi dublu clic pe un astfel de fişier pentru a-l deschide! Este un fişier cu comenzi! Se vor executa comenzile.

Nu este greu să înţelegeţi structura comenzilor create. V-ar trebui un manual de MS-DOS pentru a crea fişiere `bat` mai complexe.[38] Oricum, redenumirea este una din operaţiile de care te loveşti foarte des. Este extrem de greu să o execuţi manual.

O altă operaţie pe care uneori trebuie să o executăm asupra unui grup de fişiere este ştergerea. Comanda este `del`. Mare atenţie însă la folosirea ei. Învăţaţi pe fişiere şi-n dosare cu fişiere create doar pentru teste. Testaţi în mod repetat ceea ce aţi creat. Şi nu blestemaţi autorul pentru efectele nedorite. Cartea nu este însoţită de nici un fel de garanţii. Faceţi totul pe propria răspundere. **ştergerea de fişiere**

[38] În WindowsXP există comenzile în linie specifice. Se pot obţine explicaţii recurgând la ajutorul oferit de sistem.

1.1.4.3 Programele ascunse din Windows

Există oare şi resurse Windows mai mult sau mai puţin ascunse? Cum să nu! De unele este relativ periculos să vă apropiaţi. Altele sunt însă puţin cunoscute, dar extrem de utile.

Uneori există scurtături în `Accessories` pentru aplicaţii utile, dar ele nu sunt folosite prea des. Un exemplu este un program pe nume `kodakimg.exe`. Îl găsiţi în dosarul Windows.

Puteţi crea şi propriile dosare cu scurtături în meniul Start. Mergeţi în 2xExplorer pe următorea rută cu meniuri: Bookmarks→ Go to Folder→ Start Menu. Creaţi un nou dosar şi daţi-i un nume sugestiv. Intraţi în acest dosar şi faceţi o scurtătură către programul `kodakimg.exe`.

Când studiaţi meniul Start vedeţi un meniu derulant în care găsiţi scurtătura nou creată. Puteţi porni acum un program de prelucrare

tiff de imagini. Acest program creează fişiere de tip `tiff`. Dacă aveţi un scanner sau acces la un scanner şi copiaţi electronic, să zicem, un manuscris de zece pagini, fişierul de tip `tiff` ţine toate aceste pagini la un loc, într-un singur fişier. Puteţi răsfoi acest fişier ca pe o carte. În plus, acesta este formatul folosit de către marea majoritate a programelor OCR, programele care recunosc literele. Aceste programe pot transforma un fişier de tip `tiff` într-unul de tip text.

sfc Un alt program deosebit de util este `sfc.exe`.[39] Îl găsiţi tot în dosarul Windows.

Alte programe ne duc limpede pe terenul unde se avântă doar ucenicii vrăjitori. Mergeţi, de exemplu, din dosarul Windows în do-

MSCONFIG sarul System şi căutaţi acolo programul `MSCONFIG.EXE`. Acest program vă permite să configuraţi sistemul Win98. Evident, o faceţi pe propriul risc.

SYSEDIT Cu `SYSEDIT.EXE` puteţi edita fişierele de configurare a sistemului. Dacă nu sunteţi ucenici vrăjitori, recomandarea ar fi să editaţi doar fişierul `AUTOEXEC.BAT` în modul indicat deja mai sus.

REGEDIT Un program pe care ucenicii vrăjitori îl vor încerca desigur este `REGEDIT.EXE`. Lansaţi acest program. Acum aveţi acces la fişierele în care Win98 îşi ţine datele cu privire la programele instalate, tipuri de fişiere, moduri de deschidere a lor şi aşa mai departe. Nu umblaţi aici fără socoteală, deoarece aveţi toate şansele să instalaţi haosul în sistemul dumneavoastră.

După ce aţi lansat în execuţie Regedit, daţi clic pe cruciuliţa de la HKEY_CLASSES_ROOT. Folosiţi glisorul pentru a ajunge la

[39] V. pagina 16, rândul 29

`Folder` şi faceţi iar clic pe cruciuliţă. Puneţi săgeata pe Folder şi daţi clic pe dreapta. Alegeţi din meniu pe New ▷ Key. Numiţi această cheie `2xExplorer`. Acum construiţi o nouă cheie, dar cu rădăcina în `2xExplorer`, cheia pe care tocmai aţi creat-o. Daţi-i acestei chei numele `command`. Nu-i daţi alt nume! Aşa cum sugerează numele, aceasta este cheia pentru comanda în linie. Deschideţi-o aşa cum aţi deschide orice dosar. Chemaţi un meniu prin clic pe dreapta valorii Default. Introduceţi o comandă de tipul `D:\use\2xExplorer.exe`. Evident, puneţi calea adecvată pentru sistemul dumneavoastră. Verificaţi eventual calea cu Start→ Run→ Browse....

Închideţi Regedit. Obţineţi un meniu prin clic pe dreapta pe icoana cu numele calculatorului dumneavoastră. În meniu apare acum şi 2xExplorer. Puteţi renunţa la pictograma de pe suprafaţa de lucru.

1.1.4.4 Strat Unix peste Windows

Înainte de a trece mai departe trebuie să ne reamintim un principiu important. Cartea aceasta vă îndeamnă să învăţaţi cât mai mult experimentând cu diverse programe. Nu faceţi acest lucru însă cu mai multe programe deodată. Instalaţi pe rând diverse programe şi vedeţi ce se întâmplă. Programele interacţionează între ele şi aceste interacţiuni nu pot fi anticipate. Aţi instalat poate în acelaşi dosar două fişiere care, dacă ar sta separat, rezultatul ar fi altul. Sau poate aţi estimat eronat felul în care un program face apel la alt program. Nu vă grăbiţi să puneţi aceste efecte nedorite pe seama unor viruşi.

programele interacţionează

De asemenea, dacă un program este nestabil, Win98 tinde să se destabilizeze în totalitatea sa.[40] Nu este neapărat un motiv să renunţaţi la Win98 în favoarea unui sistem de operare care separă mai bine felul în care funcţionează diversele programe. Win98 este rodul unui efort superb de a păstra compatibilitatea cu mai vechile programe MS-DOS. Poate, de asemenea, emula Unix în bune condiţii. Instalaţi foarte prudent noi programe.

Când ne apucăm să modificăm sistemul Windows98 este bine să ştim să modificăm `AUTOEXEC.BAT`. Folosesc, din nou, exemplul computerului pe care este scrisă cartea:

Autoexec

```
SET HOME=D:\dat\home
SET VIM=D:\USE\VIM
SET GHOSTSCRIPT_FONT_DIR=D:\ed\gs\fonts
SET PATH=C:\WINDOWS;C:\WINDOWS\COMMAND;D:\USE\BIN
```

[40] Avantajul WindowsXP constă, între altele, în eliminarea acestui tip de instabilitate.

Aceasta este o parte din fişierul AUTOEXEC.BAT. Citiţi cu atenţie aceste rânduri, dar nu faceţi modificări mecanice pe calculatorul dumneavoastră. Ţineţi cont de denumirile propriilor dosare.

HOME este numele unei variabile al cărei conţinut este folosit de către diverse programe. Programele transferate din Unix folosesc dosarul indicat drept HOME pentru a plasa acolo fişiere de configurare.

Trebuie să fiţi atente şi atenţi la modul în care se instalează Vim (şi alte programe) în funcţie de existenţa sau nu a unei valori pentru HOME. Editorul care este marea alternativă la Vim, Emacs, va cere în mod explicit să daţi o valoare lui HOME.

Următorul punct important în setările de mai sus îl constituie SET PATH şi căile care urmează după aceea (vedeţi linia a patra din fragmentul din AUTOEXEC.BAT). Aceste setări vă scutesc de tastarea căii complete atunci când apelaţi un program printr-o comandă în linie. De exemplu, D:\USE\BIN va permite să apelez în linie orice executabil pus în dosarul BIN din dosarul USE aflat pe discul D. În dosarul respectiv pun, între altele, unelte aduse din lumea Unix.

variabile de mediu în WindowsXP

Cum modificăm variabilele de mediu sub WindowsXP? În sistemul XP AUTOEXEC.BAT nu mai joacă rolul din MS-DOS sau Windows98. Mergeţi pe ruta Start→ Control Panel→ System→ Advanced→ Environment Variables şi ajungeţi la o fereastră în care vedeţi lista variabilelor de mediu. XP distinge între variabilele create de utilizatori şi cele ale sistemului. De exemplu, eu am creat o variabilă HOME. Căutaţi pe PATH printre variabilele de sistem şi selectaţi cu mouse-ul respectiva variabilă de mediu. Daţi apoi un clic pe butonul Edit şi adăugaţi calea pe care doriţi să o ştie sistemul.

1.1.4.4.1 Sistem Unix minimal sub Windows

Resursele native Windows pentru comenzile în linie în fereastra MS-DOS sunt foarte limitate. O soluţie o reprezintă instalarea de unelte Unix. Pentru început recomandabilă este instalarea unui strat minimal de unelte Unix.[41]

Puteţi instala numai uneltele, fără compilatorul de C++. Dacă vreţi să accesaţi direct uneltele Unix instalate, nu uitaţi să setaţi în mod corespunzător variabila PATH a sistemului.

număraţi cuvintele din fişiere

Vom da o ilustraţie foarte simplă a modului în care sunt folosite uneltele Unix. Vreţi să ştiţi, să zicem, câte cuvinte sunt într-un fişier. Apelaţi la programul `wc.exe`, al cărui nume vine de la expresia

[41] Sistemul zis `msys` dispune de un program de instalare în stil Windows. Este excelent pentru început. Pentru detalii a se vedea `http://www.mingw.org/msys.shtml`.

englezească *word count*. Mergeţi, cu ajutorul 2xExplorer, în zona rezervată încercării programelor şi creaţi un fişier `test.txt` în care scrieţi câteva rânduri. Apăsaţi, în 2xExplorer, tasta `F10` şi scrieţi comanda `wc test.txt` şi apăsaţi tasta `Enter`. Puteţi scrie, evident, comanda şi direct în fereastra MS-DOS. Rezultatul constă dintr-o serie de cifre. Prima cifră indică numărul de rânduri; a doua numărul de cuvinte; a treia numărul de semne din fişier.

Vreţi ceva ajutor? Scrieţi comanda `wc -help` şi studiaţi utilizarea opţiunilor. Parantezele drepte indică elementele comenzii care nu sunt obligatorii. Singurul mod de a învăţa cu adevărat utilizarea unor astfel de programe este experimentatul. Numai aşa vă puteţi obişnui cu sistemul de opţiuni al comenzilor în linie.

cereţi ajutor

1.1.4.4.2 GnuWin32 Sistemul `msys` nu foloseşte direct Windows. Există unelte Unix care folosesc în mod direct sistemul de operare Windows. Proiectul GNU are o versiune pentru Windows. Găsiţi în ea un set vast de unelte Unix, precum şi multe alte programe.

Versiunea pentru Windows a proiectului GNU se numeşte „GnuWin32“. Consultaţi pagina de web cu pachetele de programe realizate în cadrul acestui proiect.[42] Alegeţi programele de care aveţi nevoie. Recomandarea noastră este ca, la început, să instalaţi doar programe care dispun de *setup*. Orice începătoare sau începător vor avea mari dificultăţi la instalarea fără *setup*.

1.1.4.4.3 Programarea în C/C++ Ucenicii vrăjitori interesaţi de programarea în C++ vor fi desigur atraşi de versiunea pentru Windows a compilatorului `gcc`. Acest compilator este foarte important în sistemul de operare zis „Linux“. Este atât de important încât mulţi cred că sistemului respectiv ar trebui să i se zică GNU/Linux. Numele celui care iniţiat elaborarea `gcc` şi mişcarea GNU, Richard Stallman, ar trebui să fie cunoscut de toată lumea.[43]

`gcc`

La ce este bun compilatorul? Nu vă convine modul cum se comportă `wc.exe`? Puteţi rescrie programul.

Pe situl <`http://sourceforge.net`> găsiţi un excelent mediu integrat de dezvoltare de programe în `C/C++`. Căutaţi un program

[42] Adresa paginii de web cu lista de pachete GnuWin32 este <`http://gnuwin32.sourceforge.net/packages.html`>. Pentru surse şi documentaţie, v.<`http://www.gnu.org`>, <`http://gnu.eunet.fi`> sau orice alt sit Internet legat de proiectul GNU.

[43] Pentru ideile mişcării GNU v. §1.1.5

care se numeşte „Dev-C++“. Veţi avea o interfaţă grafică tipică pentru Windows şi compilatorul `gcc` integrat. Cu acest mediu puteţi rescrie, evident după ce învăţaţi C/C++, programele de genul celui de numărat cuvinte.

1.1.5 Principiul surselor deschise

Sistemul de operare propriu-zis oferă puţine posibilităţi de lucru.[44] Pentru a produce un eseu frumos tipărit sunt necesare o serie de alte programe.

Ce facem însă? Noi programe ar putea însemna noi cheltuieli. Dacă este vorba despre o firmă, atunci aceste cheltuieli ar putea să-şi găsească justificarea: ele pot fi recuperate vânzând produsele firmei. Dacă este vorba despre o studentă sau un student, se profilează o dilemă foarte neplăcută. S-ar părea că alternativa este între a plăti sume considerabile şi a pirata programe.

evitaţi pirateria

Pirateria este însă hoţie şi ar trebui exclusă din start. Din păcate nu se întâmplă de multe ori aşa. În ciuda accesului masiv la Internet,[45] sunt puţin folosite programele gratuite.[46]

Gratuitatea nu este însă un scop în sine. Este foarte important să consultaţi licenţele. Uneori produsul este gratuit doar pentru a-i face (un timp) reclamă. Versiunile ulterioare sau versiunile complete nu sunt gratuite. Recomandarea noastră, mai ales în cazul eseurilor studenţeşti, este orientarea către o licenţă GNU sau o licenţă asemănătoare.

licenţa GNU

GNU este, în esenţă, o licenţă menită să asigure accesul la sursele programelor, să prevină plagiatul. Licenţa interzice, de asemenea, ascunderea rezultatelor făcute astfel publice într-un program care n-are surse deschise. Persoanele care au scris sursele sau părţi ale acestora trebuie menţionate în mod adecvat. De asemenea, cine operează o modificare în surse trebuie să arate acest lucru în mod explicit şi să nu împiedice accesul altora la surse.

GNU nu este o licenţă menită să asigure gratuitatea programelor.

[44] Linux este, de obicei, distribuit împreună cu o mulţime de programe de editare de texte, imagini etc. Cred că trebuie distins însă între aceste programe şi sistemul de operare ca atare. Windows include, cu unele excepţii, programe suplimentare foarte modeste şi firma a fost chiar atacată în justiţie din pricina integrării în distribuţia de bază a Internet Explorer.

[45] În Uniunea Europeană, în 2003, accesul la Internet, în diverse ţări, variază între aproximativ 20% şi 75%. La noi procentul este desigur mic, dacă numărăm casele şi apartamentele conectate la Internet. Situaţia este diferită în cazul universităţilor.

[46] Programele gratuite sunt distribuite, de asemenea, şi de către reviste.

Aşa cum atunci când reproduci în editura proprie lucrarea altcuiva sau plăteşti pentru a citi o carte, nimic nu împiedică plata unei sume de bani pentru a avea acces la sursa unui program. În practică însă, multe dintre persoanele caru au scris programe cu sursă deschisă au fost generoase şi au plasat sursele şi programele în domeniul public.

O serie de astfel de programe sunt rodul cercetării din mediul academic. Este absolut firesc ca, atunci când le cerem studentelor şi studenţilor să tehnoredacteze un eseu, filosofic sau nu, să le arătăm şi cum pot folosi aceste programe.[47]

Pentru a înţelege mai bine sensul acestor principii ar trebui să clarificăm ce se înţelege prin *sursa* unui program.

forţa minţii este mai importantă

Toţi cei care sunt ahtiaţi după **forţa** calculatorului lor discută despre **viteza** procesorului calculatorului. Mai importante sunt însă programele rulate pe calculatoare. Adevărata forţă o reprezintă minţile care programează inteligent. Pentru programare sunt folosite limbaje de nivel mai înalt, care au anumite asemănări cu limbajele umane.

Sursele programelor sunt scrise în aceste limbaje de nivel mai înalt. Ele sunt stocate în fişiere de tip text.[48]

sursele deschise şi accesul la cunoaştere

În mediul academic şi-n cercetare, în lipsa surselor, nu poate fi studiată decât utilizarea programelor. Lucrul acesta este absurd. Gândiţi-vă la un program care, dacă îi daţi două numere naturale, vă spune care este cel mai mare divizor comun al lor. N-aveţi cum şti însă ce algoritm foloseşte. Este ca şi cum am învăţa matematica fără a studia algoritmul lui Euclid. A studia doar utilizarea programelor ar echivala cu transformarea universităţilor în şcoli de meserii.

ameliorarea programelor

Există însă şi alte avantaje ale surselor deschise în raport cu programele care sunt doar gratuite. Autoarea sau autorul unui program gratuit ar putea să înceteze să mai dezvolte produsul său. Aceasta îi obligă pe utilizatori să folosească alt program. În cazul surselor deschise, dacă programul este cu adevărat valoros, se va găsi cu siguranţă cineva care să-l dezvolte în continuare.

În practică, principiul surselor deschise este foarte avantajos când trebuie să operezi modificări, de multe ori banale, într-un program. De pildă, am vrut, cum este normal, ca bibliografia acestei cărţi să nu fie presărată cu cuvinte englezeşti inutile, precum `and` (şi). Am

[47] Nu cred că este etic să-i forţăm să folosească programe comerciale scumpe. S-ar putea ca astfel să-i împingem pe calea pirateriei. De aici şi până la producerea de eseuri plagiate nu mai este decât un pas. Dacă un gen de hoţie este permis, alte tipuri de hoţie devin şi ele acceptabile.

[48] De aceea un bun editor de texte este atât de important pentru programatori.

folosit accesul la sursele programului care determină stilul bibliografiei pentru a obţine rezultatul dorit.[49]

1.2 Editorul Vim

Vim continuă tradiţia principalului editor din lumea Unix, editorul vi. Vi a fost creat de William Joy.[50] În ciuda faptului că, în forma sa actuală, Vim dispune de meniuri, mulţi s-ar putea să-l găsească neatrăgător.

Cred că unii se gândesc deja să-şi scrie eseul cu unul dintre *office*-uri.[51] Cartea de faţă vă propune însă altă perspectivă decât cea pe care o adoptă un program pentru munca de birou. Problema nu este de a avea un program pentru munca de birou care să fie gratuit.[52] Ideea este de a schimba perspectiva utilizării cu aceea a programării.

perspectiva programării

Din perspectiva programării, un document tipărit este realizat cu ajutorul unui program. Programul este cel care stabileşte aşezarea în pagină a textului, imaginilor, tabelelor. Tot prin program sunt specificate tipurile de litere folosite. Tot programarea este cea care ne permite să inserăm note, trimiteri, bibliografii.

[49] Un program care nu permite accesul la surse oferă posibilitatea de a regla o serie de parametri. Dacă parametrul care ne interesează nu este reglabil trebuie să ne adresăm direct firmei producătoare. Licenţa nu ne permite să modificăm programul.

[50] William Joy este unul dintre fondatorii Sun Microsistems. Vi ţine însă de perioada premergătoare celei de la Sun. Pentru istoria Vi v. interviul luat lui William Joy de către Eugene Eric Kim, *Linux Magzine*, <`http://www.linux-mag.com`>, noiembrie 1999 (data accesări: 11/01/2004), „What inspired you to write vi?“.

[51] Are, evident, sens să instalaţi un *office*. Puteţi pune foarte uşor, de pildă, OpenOffice, care este versiunea cu sursă deschisă a unui produs al firmei Sun, firma lui William Joy. Programul acesta poate face şi operaţii care depăşesc sfera operaţiilor normale într-un birou. Este un program care, dacă mergeţi pe ruta Tools→ Data Sources veţi vedea, cu titlu de demonstraţie, cum puteţi lucra şi cu o bibliografie ale cărei elemente le puteţi integra în documentul dumneavoastră. Apăsaţi OK. Apoi apăsaţi F4. Expandaţi arborele Bibliography şi ramura Tables. Veţi vedea chiar datele pe care le cuprinde bibliografia demonstrativă instalată o dată cu programul. OpenOffice are şi o serie de alte facilităţi pentru munca intelectuală. De asemenea, este capabil să deschidă şi să creeze documente într-o serie de formate întâlnite frecvent.

[52] Dacă folosiţi Vim, vedeţi desigur că acesta nu este un program pur şi simplu gratuit. Bram Moolenaar îi încurajează pe utilizatori să facă donaţii unei fundaţii care desfăşoară activităţi caritabile în Uganda.

1.2.1 Învăţarea Vim într-o zi

Vim este mai prietenos decât vechiul vi. Vi te punea în faţa unui ecran negru. Apăsai tastele şi nu se întâmpla nimic, dacă nu tastai din greşeală `i` sau `a`. Atunci apăreau nişte litere pe ecran, dar mare lucru nu se întâmpla. Te întrebai cine o fi folosind Vi. Răspunsul este relativ simplu: cei care scriu programe.

Mergeţi la adresa de Internet <`http://www.vim.org`> pentru a afla cum puteţi descărca Vim de pe Internet. Versiunea cu care sunt scrise şi sursele acestei cărţi este 6.2.

Instalarea versiunii 6.2 sub Windows este cât se poate de simplă. Vă recomand să optaţi pentru o apropiere cât mai mare de stilul Windows. Astfel, ceea ce aţi învăţat exersând cu Notepad puteţi aplica în continuare.

Optaţi, de asemenea, pentru posibilitatea de a recurge la Vim direct din meniurile contextuale ataşate oricărui fişier.

1.2.1.1 Un editor modal

Vim este un editor dedicat în special editării de fişiere de tip text. Este bine să vă familiarizaţi mai întâi cu modul în care Vim vă permiteţi să prelucraţi un fişier de tip text.

Creaţi un fişier de tip text în 2xExplorer. Deschideţi-l cu Vim, nu cu Notepad. Dac-aţi reuşit, vedeţi cum clipeşte un dreptunghi negru în colţul din stânga sus al ferestrei Vim. Inutil apăsaţi diverse litere. De regulă, nu veţi face decât să apară litera respectivă pe ultima linie a ecranului. Tastaţi `i`. Forma cursorului s-a schimbat. Pe ultimul rând al ferestrei scrie acum `-- INSERT --`, iar dumneavoastră aţi început să descoperiţi, fără să ştiţi, ideile de bază ale programării.

La început eraţi în modul numit „modul normal“. Puteţi reveni oricând în acest mod apăsând tasta `Esc` care este plasată în colţul din stânga sus al tastaturii unui PC. În acest mod Vim interpretează apăsarea oricărei taste ca pe o comandă sau ca pe începutul unei comenzi.

modul normal

1.2.1.1.1 Atenţie la moduri Când au fost introduse calculatoarele la bordul avioanelor, un teribil accident a avut loc în Franţa din pricina nefamiliarizării piloţilor cu ideea de **mod de lucru**. Piloţii au vrut să ocolească un munte. Au dat o comandă pentru a câştiga în altitudine. Din păcate, nu au fost atenţi la **modul** în care se afla programul calculatorului de bord. Avionul s-a dus cu încăpăţânare direct spre munte.

Când deschideţi Vim vă aflaţi în modul normal. Dacă tastăm `d` de două ori, ştergem un rând! Din fericire, putem tasta `u` şi rândul se reface.

Sistemul complet al modurilor Vim este descris în documentaţia care însoţeşte editorul.

1.2.1.1.1.1 Modul normal Modul normal al Vim este cel în care daţi comenzi dintr-un punct de pe suprafaţa de lucru a editorului. Cea mai frecventă comandă în acest mod va fi probabil `i`. Tastând `i` puteţi începe să introduceţi text.

`i`

Încercaţi să vedeţi care este efectul apăsării diverselor taste în modul normal. Care este, de pildă, diferenţa tastarea lui `a` şi a lui `i`? Care este efectul lui `A`?

`a`

1.2.1.1.1.2 Inserarea textului Felul în care ajungem să introducem text a fost deja explicat (vezi secţiunea 1.2.1.1, rândul 26). Ca să ieşim din acest mod apăsăm tasta Esc şi revenim în modul normal.

1.2.1.1.1.3 Înlocuirea unei porţiuni de text Din modul normal puteţi trece în modul în care suprascrieţi textul deja existent în două feluri: dacă tastaţi `r` sau dacă apăsaţi combinaţia de taste Shift+r. În primul caz, după înlocuirea unei litere reveniţi la modul normal. În al doilea caz, reveniţi în modul normal doar apăsând Esc.

`r`

Sub Windows, este probabil mai comod să suprascriem chiar pe parcursul introducerii de text. Apăsaţi tasta Ins. Pe ultima linie a ferestrei Vim apare `-- REPLACE --`. Ca să ieşim din acest mod (familiar oricui a mai utilizat programe Windows!) apăsăm din nou tasta Ins.

1.2.1.1.1.4 Selectarea textului ca-n Windows Pentru a selecta text ţineţi tasta Shift apăsată şi folosiţi tastele cu săgeţi sau cele pentru derularea paginilor. Observaţi mesajul `- (insert) SELECT -` care apare pe ultimul rând al ferestrei Vim.

În acest mod puteţi lucra cu obişnuitele comenzi Windows de copiere, decupare şi inserare de text.[53]

[53] Vezi secţiunea 1.1.1.1.1.

1.2.1.1.1.5 Modul vizual Vim are şi modul său propriu de a selecta text. Dacă, în modul normal, tastaţi `v`, atunci treceţi în acest mod. Puteţi selecta apoi text în maniera Windows sau folosind comenzi de deplasare a cursorului specifice Vim. v

Modul vizual are o versiune în care selectaţi câte un rând în întregimea sa.[54] Pentru aceasta tastaţi `V` în modul normal. V

La prima vedere, modul vizual nu prezintă mare interes. El are însă unele avantaje subtile în raport cu selecţiile Windows când trebuie să copiem text dintr-un mod Vim într-altul.

Regula pe care v-aş recomanda să o urmaţi este următoarea: folosiţi cu selecţiile în stil Windows comenzile (de copiere, decupare, lipire) în stil Windows. Când vreţi să folosiţi comenzi specifice Vim, utilizaţi modul vizual pentru selecţii.

Există, de asemenea, un mare avantaj al selecţiei în modul vizual: selecţia unor coloane de text. Creaţi un tabel precum cel de la pagina 33, rândul 6. Mergeţi în colţul uneia dintre coloane, treceţi în modul normal şi apăsaţi Ctrl-Q. Acum puteţi selecta coloana respectivă. **Ctrl-Q**

În Unix selecţia blocurilor de text în mod vizual se face după apăsarea Ctrl-V, dar acest lucru nu este posibil sub Windows, unde combinaţia respectivă de taste serveşte la inserarea unui text copiat sau decupat.

1.2.1.1.1.6 Comenzile în linie Comenzile pe care le putem da tastând doar o literă sau folosind combinaţii de taste sunt fatalmente limitate. Vim are însă un set de comenzi (native) mult mai bogat. Pentru a avea acces la ele trebuie să treceţi din modul normal în modul comandă în linie. Cum? Apăsând în modul normal pe tasta : (tasta cu două puncte pe ea).[55] **trecerea la comanda în linie**

În aceste anexe găsiţi fragmente de cod. Fiecare rând este numerotat. Puteţi obţine şi-n Vim o numerotare a rândurilor textului. În modul normal tastaţi `:set nu!` (nu vă speriaţi din pricina faptului că după ce-aţi tastat două puncte treceţi pe ultima linie; este semnul c-aţi schimbat modul). Apăsaţi Enter şi apar numerele.

Cum scăpaţi de numere? Treceţi iar în modul comandă în linie!

[54] Este vorba despre rânduri în sens logic. Vezi despre rânduri şi secţiunea 1.2.1.2.

[55] Se poate trece în modul comandă în linie şi apăsând tasta /. Acest lucru îl facem atunci când vrem să dăm o comandă de căutare prin restul textului (din punctul în care se află cursorul în modul normal). Tastăm însă ? (în modul normal) atunci când vrem să căutăm în porţiunile de text aflate înainte de punctul în care se află cursorul.

Apăsaţi tasta cu săgeata în sus. Vim ţine minte comenzile în linie.[56] Repetaţi comanda anterioară. Numerele dispar.

1.2.1.2 Rânduri logice şi rânduri vizuale

Scrieţi, de pildă, ceva care vă trece pe moment prin cap. Apăsaţi şi Enter de câteva ori. Încercaţi să vă mişcaţi prin text cu săgeţile de pe tastatură. Observaţi cum Vim consideră că un rând se termină logic când apăsăm ENTER. Puteţi să vă mişcaţi însă oriunde în text folosind mouse-ul şi cursorul din Windows. Atenţie însă! Pentru a schimba astfel poziţia cursorului Vim trebuie să faceţi clic în punctul în care vreţi să vă opriţi.

Distincţia dintre rândurile logice şi cele vizuale este foarte importantă. Rândurile vizuale depind de dimensiunea ferestrei Vim. Pentru a obţine rânduri vizuale daţi comanda `:set wrap!`; efectul ei este afişarea rândurilor logice în întregimea lor, chiar dacă depăşesc dimensiunile ferestrei.

Dezavantajul lui `wrap` este că rupe afişarea în mijlocul cuvintelor. Alternativa este comanda `:set lbr!`. Ea are drept efect crearea unor rânduri vizuale în care cuvintele rămân întregi.

1.2.1.3 Explorarea meniurilor

Citiţi ceea ce găsiţi când faceţi clic pe `Help` în meniuri, dar nu încercaţi să pricepeţi tot de la început. Pe de o parte, explicaţiile par scrise pentru cineva destul de versat în lucrul cu acest gen de editoare. Pe de altă parte, multe dintre noţiuni sunt mai degrabă relevante în contextul unui sistem de operare de tip Unix.

Mai productiv, la început, mi se pare alt mod, foarte simplu de a învăţa Vim. Parcurgeţi cu răbdare meniurile programului şi vedeţi ce este de folos pentru dumneavoastră acolo. De exemplu, meniul File vă spune cum puteţi deschide, închide şi salva fişiere.

Veţi vedea însă în meniuri şi expresii misterioase de genul `:w`. Dacă scrieţi `:w` şi apăsaţi `Enter` veţi salva fişierul şi veţi primi o confirmare de genul `"test.txt" 3L, 144C written`. Mesajul afişat de Vim cuprinde numele fişierului, numărul de rânduri, numărul de caractere şi cuvântul englezesc *written*, pentru a ne indica scrierea pe disc a fişierului respectiv.

La meniul Edit acordaţi importanţă în special secţiunii File Set-

[56] Vim ţine minte separat comenzile puse după două puncte de cele realizate cu comenzile de căutare prin text.

tings şi rubricii Select Font.... Mergeţi la File Format... şi observaţi că Vim cunoaşte cele trei tipuri de fişiere text.[57]

Literele (în engleză *fonts*) sunt o problemă cu mult mai dificilă. Alegeţi un tip de literă (în engleză, *font*) care dispune de un `Script` de tipul `Central European`. În acest fel aveţi acces la literele româneşti. Nu trebuie decât să comutaţi în Windows pe tastatura românească.[58] Vim este perfect capabil să lucreze cu ea.

litere româneşti

Dacă revenim acum la meniurile principale, urmează trei meniuri mai ciudate: Tools, Syntax, Buffers. Nu este nevoie să le studiaţi cu foarte mare atenţie de la bun început. Primul dintre cele trei meniuri va fascina desigur pe ucenicii vrăjitori. Cu ajutorul lui puteţi învăţa, de pildă, cum să editaţi codurile numerice pe care le foloseşte Windows în fişierul dumneavoastră. Al doilea ne introduce o trăsătură crucială a Vim: capacitatea sa de a vizualiza în culori diferite comenzile şi datele din fişiere.[59] Al treilea trebuie studiat cu atenţie. Conceptul folosit acolo nu trebuie confundat cu cel de fereastră. Buferele sunt însă efectiv utile când prelucrăm deodată un mare număr de fişiere.

1.2.1.3.1 Configurarea Vim Cum putem regla modul în care se comportă Vim? Pentru a înţelege acest lucru trebuie să ne amintim că Vim vine din lumea Unix. În acest sistem de operare obiceiul este să se folosească fişiere de configurare. Dacă mergeţi în dosarul în care este instalat Vim veţi găsi un fişier care se numeşte `_vimrc`.[60] Imediat după instalare veţi găsi rândul următor în acest fişier:

```
1 source $VIMRUNTIME/vimrc_example.vim
```

Căutaţi scriptul `vim` la care se face referire în acest rând în dosarul în care se află principalul fişier executabil din Vim `gvim.exe` şi faceţi o copie a acestui fişier. Redenumiţi copia. Cel mai normal ar fi să-i ziceţi simplu `vimrc.vim`; puteţi acum să faceţi modificările dorite în această copie. Apoi, în `_vimrc` modificaţi în mod adecvat rândul care trimitea la exemplul de configurare creat la instalare.

[57] Fişierele Dos sunt totuna cu fişierele text din Windows. Vezi mai sus (pagina 8, rândul 4) deosebirile dintre trei tipuri de fişiere text.

[58] V. §2.3.1

[59] Pentru detalii v. §1.2.1.3.3.1.

[60] Linia din faţa numelui fişierului imită stilul Unix. În Unix fişierele de configurare au nume care încep cu un punct. Linia n-are un mare rol aici. S-ar putea să fie o măsură de precauţie faţă de programele antivirus care văd un duşman în tot ce nu este ca-n Windows.

Pe calculatorul cu care este scrisă cartea adăugirile mai importante sunt următoarele:

```
set guifont=lucida_console:h12:cEASTEUROPE
set listchars=tab:\|\_,trail:-,eol:<
set lbr!
set shiftwidth=3
set tabstop=3
set noexpandtab
```

literele Prima linie de mai sus spune Vim ce font să folosească atunci când deschide un fişier, cât de mare să fie corpul literei şi să recurgă la un litere printre care se găsesc şi cele româneşti. Rolul celorlalte linii îl puteţi descoperi lesne dacă folosiţi bara cu instrumente (bara **ajutor Vim** cu pictograme) şi faceţi clic pe semnul de întrebare cu lupă. Puneţi cuvântul care vă interesează în caseta de dialog şi cereţi ajutor. Atenţie! Semnul de exclamare, egalul etc. nu fac parte din cuvintele-cheie ca atare.

1.2.1.3.2 Scripturile Vim Vim are desigur şi multe puteri ascunse. Este imposibil să converteşti în meniuri tot ce poate face Vim. **limbajul Vim** Importantă este extensibilitatea sa. Vim foloseşte pentru aceasta propriul său limbaj.[61] Acest limbaj nu este însă foarte greu de însuşit şi putem crea cu ajutorul lui o mulţime de lucruri utile, inclusiv noi meniuri.

În limbajul Vim se pot scrie programe. Un program Vim este scris **script** într-un fişier de tip text, dar care are extensia `vim`. Aceste fişiere sunt colecţii de comenzi pe care le interpretează Vim. Există şi un nume generic pentru astfel de programe. Ele se numesc „script“-uri.

În primul rând, merită să extindeţi Vim cu ajutorul altora. În **vimfiles** funcţie de modul în care aţi instalat Vim, căutaţi un dosar care se numeşte `vimfiles`. Căutaţi în el dosarul `plugin`. Recomandarea ar fi să nu folosiţi dosarul `plugin` din `vimfiles`. Puteţi astfel distinge între extinderile operate direct din instalare şi cele ulterioare. O altă variantă ar fi să păstraţi unul dintre dosare pentru extinderile create de alţii şi una pentru cele create de dumneavoastră.

Eu am adăugat, de pildă, `UnMtchBracket.vim`. Este o extensie creată de Chandra Naveen. Îţi permite să sesizezi dacă o paranteză a fost sau nu închisă. După cum veţi observa ulterior, acest lucru este absolut esenţial din perspectiva programării. Extensia aceasta vă scuteşte de multe bătăi de cap.

[61] Spre deosebire de Emacs, care foloseşte Lisp.

Uneori vreţi să vedeţi însă care este perechea unei paranteze date. Pentru aceasta procedaţi în felul următor. Treceţi în modul comandă. Puneţi cursorul pe paranteza căreia vreţi să-i găsiţi perechea. Apăsaţi `%`. Cursorul se va muta pe paranteza pereche. Puteţi face acest lucru cu toate cele trei tipuri de paranteze. %

O altă extindere utilă a Vim este `DirDo.vim`. Autorul acestui program este William Lee. Valoarea sa o veţi descoperi imediat ce încercaţi să modificaţi foarte multe fişiere aflate în dosare diferite.

1.2.1.3.3 Vim bun la toate Ucenicii vrăjitori vor vrea desigur să deschidă cu Vim printr-un dublu clic o mulţime de fişiere. Pot face acest lucru în felul următor. Se porneşte programul Regedit. În HKEY_CLASSES_ROOT este localizată cheia txtfile. Se merge pe ramura care porneşte din această cheie până se ajunge la comanda pentru deschiderea fişierelor de tip text şi acolo se schimbă comanda care spune Windows să folosească Notepad cu una care deschide fişierul cu Vim (ceva de genul `D:\use\vim\vim62\gvim.exe %1`). Evident, calea către `gvim.exe` trebuie să fie cea de pe calculatorul respectiv. Trebuie pus şi `%1`, care spune sistemului să deschidă fişierul pe care se face dublu clic.

Nu trebuie însă abuzat de procedurile de genul celei de mai sus. Putem deregla relativ lesne Windows. De asemenea, de multe ori este mai interesant să deschidem simultan mai multe fişiere cu Vim sau să deschidem cu Vim un anume tip de fişier numai atunci când vrem să editǎm fişierul, nu şi atunci când vrem să-l vizualizăm.

1.2.1.3.3.1 Verificarea corectitudinii sintactice Fiecare limbaj de programare are sintaxa sa. Greşelile de sintaxă trebuie evident eliminate. Pentru a ne ajuta în procesul de identificare a erorilor de sintaxă Vim colorează textul programelor.

evidenţierea sintaxei prin colorare

Comanda pentru colorarea textului este dată în fişierul de configurare. Vim foloseşte extensia fişierelor pentru a detecta limbajul în care este scris conţinutul lor. De asemenea, Vim foloseşte informaţiile conţinute în primul rând al anumitor fişiere.

Dacă Vim nu detectează sintaxa sau vreţi să o porniţi manual, mergeţi la meniul Syntax; apăsaţi Show filetypes in menu pentru a putea alege limbajul dorit. Restul comenzilor au nume care explică limpede rolul lor.

1.2.1.3.3.2 Integrarea Vim în 2xExplorer Din 2xExplorer poate se poate apela direct la un editor extern. Folosind View→

Options... bifaţi butonul radio pentru editorul extern şi indicaţi calea către `gvim.exe`.

F4 Puteţi deschide acum cu Vim orice fişier prin simpla apăsare a tastei F4. Metoda aceasta este foarte comodă şi este o bună alternativă la meniul contextual.

1.2.1.3.3.3 Câte cuvinte sunt în fişier? Studentele şi studenţii sunt uneori foarte nedumeriţi când li se cere ca eseul lor să nu depăşească un anumit număr de cuvinte. Vor să măsoare totul în pagini de text.

În epoca fişierelor electronice este însă relativ lesne să numeri cuvintele.

Vreţi să aflaţi câte cuvinte aţi scris în fişier? Treceţi în modul comandă. Tastaţi G. Apoi tastaţi CTRL+G (ţineţi tasta CTRL apăsată şi, în acelaşi timp, apăsaţi G). Pe ultimul rând al ferestrei Vim va apărea un mesaj care vă spune la al câtelea cuvânt sunteţi şi câte cuvinte sunt în fişier.[62]

1.2.1.3.3.4 Semne de carte În fişierele foarte mari este greu să revenim cu uşurinţă la un punct din fişier. Vim are posibilitatea de a se deplasa la rândul dorit, dar pentru a folosi această metodă trebuie să notăm numere de rând. Mai simplu este să punem semne de carte.

Treceţi în modul comandă (modul normal al Vim). Tastaţi `m`. Apoi tastaţi o literă oarecare. Este recomandabil ca litera respectivă să vă fie cumva asociată cu tema textului în punctul unde am pus semnul de carte.

Mergeţi în altă parte a textului. Cum reveniţi în locul în care am pus semnul de carte? Treceţi în modul comandă. Tastaţi `'`. Tastaţi apoi litera care slujeşte drept semn de carte şi veţi reveni în punctul în care aţi pus semnul de carte.

1.2.2 Vim la modul serios

Ce facem, de fapt, cu Vim? Am tot vorbit despre fişierele text. Eu am crea unul pe care l-am folosit când am testat exemplele care urmează. Este un fişier foarte simplu, cu următorul conţinut:

```
1 filosofie uman
```

[62] Pentru mai multe detalii a se vedea manualul scris de Bram Moolenaar pentru utilizatorii Vim (secţiunea 12.5).

```
\section{woo} poveste mit
filosofic bibliografie
```

Am botezat acest fişier `woo.txt`. Litera `w` sugerează faptul că este vorba de un fişier text de tip Windows. Puteţi crea lesne unul la fel sau unul asemănător.

Mergeţi în meniul principal la Tools şi daţi clic pe Convert to HEX. Fereastra Vim se modifică. În cazul fişierului meu `woo.txt` conţinutul ferestrei arată astfel:

```
0000000: 6669 6c6f 736f 6669 6520 756d 616e 0d0a  filosofie uman..
0000010: 5c63 6861 7074 6572 7b77 6f6f 7d20 706f  \section{woo} po
0000020: 7665 7374 6520 6d69 740d 0a66 696c 6f73  veste mit..filos
0000030: 6f66 6963 2062 6962 6c69 6f67 7261 6669  ofic bibliografi
0000040: 650d 0a                                  e..
```

Ce s-a întâmplat? Vim a folosit un program extern[63] pentru a ne dezvălui din ce anume este făcut fişierul `woo.txt`. Fişierul este pur şi simplu o colecţie de coduri. **xxd**

Rezultatul afişat de către Vim constă din cinci rânduri. În stânga vedeţi numerele pe care le-am pus aici pentru a vă uşura numărătoarea.În centrul primelor patru rânduri se văd opt grupuri de patru cifre. Trebuie să grupaţi aceste cifre câte două. Fiecare pereche de cifre reprezintă un număr scris în baza 16.[64]

S-ar putea să vi se pară curios, dar scrierea numerelor în baza 16 este mai comodă. De ce? Procesorul computerului lucrează, de fapt, cu numere în baza 2. El vede miezul unui tabel de genul următor:

```
00: 01100110 01101001 01101100 01101111 01110011 01101111  filoso
06: 01100110 01101001 01100101 00100000 01110101 01101101  fie um
0c: 01100001 01101110 00001101 00001010 01011100 01100011  an..\c
12: 01101000 01100001 01110000 01110100 01100101 01110010  hapter
18: 01111011 01110111 01101111 01101111 01111101 00100000  {woo}
1e: 01110000 01101111 01110110 01100101 01110011 01110100  povest
24: 01100101 00100000 01101101 01101001 01110100 00001101  e mit.
2a: 00001010 01100110 01101001 01101100 01101111 01110011  .filos
30: 01101111 01100110 01101001 01100011 00100000 01100010  ofic b
36: 01101001 01100010 01101100 01101001 01101111 01100111  ibliog
3c: 01110010 01100001 01100110 01101001 01100101 00001101  rafie.
42: 00001010                                               .
```

Observaţi că nu mai sunt acum decât şase coloane în centrul tabelului. Sunt aceleaşi numere ca şi mai sus, dar scrise în baza 2. Se văd

[63] Programul `xxd.exe`, pe care-l puteţi folosi independent de Vim. Îl găsiţi în dosarul în care se află şi `gvim.exe`.

[64] De aici denumirea de „hex“ (de la *hexadecimal*).

limpede grupurile de opt cifre de `0` sau `1`. În limbajul informaticii un grup de asemenea cifre binare se numeşte „octet“[65]

Creaţi cu ajutorul Vim un fişier `woof.txt` cu următorul conţinut:

```
1 filosofie uman
2 \section{woof} poveste mit
3 filosofic bibliografie
```

În 2xExplorer selectaţi pe rând cele două fişiere şi apăsaţi tasta F12. Comparaţi dimensiunile! Fişierul `woo.txt` are 67 de octeţi(*bytes*); fişierul `woof.txt` are 68 de octeţi. Fişierele diferă între ele printr-o singură literă! Litera `f` de pe rândul 2 diferenţiază cele două fişiere.

octet

Concluzia este foarte simplă. Un octet serveşte la codificarea unei litere. Dar un octet este format din două grupuri de patru `0` sau `1`. În reprezentarea în baza 16 a octetului fiecărui grup de patru cifre binare îi corespunde o singură cifră. Este mai comod să folosim scrierea codului numeric în baza 16: scriem doar două cifre, nu opt. Iar trecerea este directă: fiecare cifră din *hex* este tradusă separat în patru cifre binare corespunzătoare.

Uitaţi-vă şi la numărătoarea de pe margine. Noi am folosit obişnuita reprezentare în baza 10 a numerelor pentru rânduri. Programul `xxd` a pus însă (pe coloana separată cu două puncte de rest) cifre în baza 16. În baza 16, `10` din baza 10 este `a`, `11` este `b`, `12` este `c`, `13` este `d`, `14` este `e` şi 15 este `f`. Când avem de a face cu o reprezentare în baza 16 formată din două cifre, prima cifră trebuie înmulţită cu 16 pentru a-i afla valoarea (în baza 10, mai accesibilă nouă); rezultatul, adunat cu a doua cifră ne dă, reprezentarea în baza 10. Uitaţi-vă pe rândul opt mai sus: cifra `2a` reprezintă numărul $2 * 16 + 10 = 42$ (în baza 10). Este numărul de octeţi de pe primele şapte rânduri.

Complicat? Şi da şi nu. Ar fi imposibil să manevrăm direct codurile numerice. Vim ne oferă însă o vizualizare a acestor coduri. Programul ne ajută, de asemenea, să facem tot felul de modificări în coduri: să adăugăm, să ştergem, să schimbăm între ele coduri şi aşa mai departe.

Să ne concentrăm atenţia asupra codurilor în baza 16. Puteţi să identificaţi codul pentru litera `f`? Este `66`.[66]

[65] În engleză, *octet* sau *byte*.

[66] Citiţi „şase-şase“! Nu este chiar corect, dar „şaizeci şi şase“ ar fi absolut greşit, deoarece este vorba despre o reprezentare în baza 16 a numărului pe care în baza 10 îl reprezentăm prin `102`. „Şase ori şaisprezece plus şase“ ar fi corect, dar – dacă n-aveţi cumva capacităţi de a face rapid calcule în minte – mai mult ne încurcă.

spaţiul alb are un cod

Un spaţiu alb are şi el un cod! Este vorba de `20`. Dacă fişierele dumneavoastră nu coincid cu dimensiunile indicate aici, probabil aveţi vreun spaţiu alb pe undeva.

Sfârşitul de rând este indicat prin două coduri (în Windows). Acestea sunt `0d`, zis CR, şi `0a`, zis LF.[67]

CR LF

Există un cod pentru sfârşitul de fişier? În principiu da, dar Vim a construit fişierele fără să introducă acest cod. Nu este o greşeală. Codul respectiv ar fi `1a`, zis şi EOF[68]. Vim l-ar vizualiza (în modul în care este setat la instalare) cu un `^Z`.

Există fireşte un cod şi pentru bara oblică inversă. Acest cod este `5c`.

În rezumat, serviciul fundamental pe care ni-l face un editor ca Vim este acela de a ne facilita manipularea colecţiilor de coduri de genul celor descrise mai sus. Vim nu este destinat afişării frumoase a unui text într-o pagină. Nu este un sistem de tipărire, fie pe ecran, fie pe hârtie.

1.2.2.1 Atracţia interfeţelor grafice

Vi este editorul favorit al persoanelor care folosesc **metoda oarbă** de tastare.[69] Pentru a folosi **oarba** trebuie însă să ţii permanent degetele pe taste.[70] Mouse-ul mai mult încurcă lucrurile pentru că trebuie să ridici mâna de pe taste.

Vim continuă tradiţia vi şi poate fi folosit fără a ridica mâna de pe taste. În versiunea sa cu interfaţă grafică Vim are însă meniuri şi permite recursul masiv la serviciile mouse-ului.

În continuare, vom folosi meniuri pentru a prezenta posibilităţile Vim. În general, vom urma ordinea temelor din manualul lui Bram Moolenaar[7]. Moolenaar explică însă ideile pentru cineva care foloseşte cu prioritate tastatura.[71]

[67] Pentru detalii despre sfârşitul de rând în diverse tipuri de fişiere vezi §1.1.2, pagina 8, rândul 4.

[68] De la expresia englezească *end of file*.

[69] Numele sugerează faptul că persoana respectivă se uită la ecran sau în altă parte, dar nu la taste.

[70] Tastele F şi J au pe ele mici profile distincte pentru a putea reveni lesne la poziţia standard a mâinilor pe tastatură.

[71] Pentru explicaţii cuprinzătoare, bogat ilustrate, vezi cartea despre Vim scrisă de Steve Oualline[9].

1.2.2.1.1 Construirea meniurilor Vim are un sistem foarte flexibil de adăugare a unor meniuri suplimentare.[72] Pentru a învăţa scrierea de meniuri, creaţi un dosar special pentru exerciţii. În acest dosar (folder) creaţi un fişier aidoma lui `woo.txt`[73].

un prim meniu

Comenzile pentru toate meniurile pe care le veţi crea trebuie puse în fişiere cu extensia `vim`. Creaţi, în dosarul pentru exerciţii, un fişier `simplu.vim`; aici veţi pune cel mai simplu meniu posibil. Scrieţi următoarea linie de cod (fără numărul de linie din faţă!):

```
1 :imenu Simplu.text Am realizat un prim meniu.
```

Salvaţi conţinutul fişierului.

Deschideţi fişierul `woo.txt` cu ajutorul Vim. Sunteţi în modul normal! Mergeţi cu ajutorul mouse-ului sau al tastelor cu săgeţi pe ultimul rând. Apăsaţi tasta O (fără să apăsaţi SHIFT!).[74] Cursorul se mută pe rândul următor şi editorul trece în modul `insert`. Pe bara cu instrumente (bara cu pictograme) daţi clic pe omuleţul care aleargă (Run a Vim Script). Se deschide o casetă de dialog. Deschideţi `simplu.vim` ca pe orice fişier. În bara cu meniuri apare un nou meniu.

Acum puteţi da un clic pe meniul Simplu şi apoi pe text. Pe ultimul rând al fişierului apare textul `Am realizat un prim meniu.` Puteţi da un clic pe pictograma Undo (săgeata răsucită către stânga) şi anula efectul meniului dumneavoastră. Cu un alt clic pe Undo aduceţi fişierul la forma iniţială.

Ce trebuie să studiaţi în codul meniului? În primul rând cele trei blocuri ale meniului.

Primul bloc începe cu două puncte. Am putea deci folosi acest cod pentru o comandă în linie! Apoi urmează `imenu`, care spune programului că este vorba de un meniu care funcţionează în modul `insert`.

Al doilea bloc conţine textul care apare în meniuri. Atenţie la punct. Nu puneţi spaţii albe aici! Ar fi interpretate ca un semn că urmează al treilea bloc. Dacă vreţi spaţii albe, trebuie să le precedaţi cu o bară oblică inversă, ca-n exemplul următor:

```
1 :imenu Simplu.pre\ text Am realizat un prim meniu.
```

[72] A se vedea manualul lui Moolenaar[7, §42].

[73] Vezi pagina 32, rândul 33

[74] Combinaţia SHIFT+O creează un rând deasupra celui pe care am dat comanda.

În al treilea bloc pot să apară spaţii albe. Al treilea bloc conţine comenzile pe care le execută Vim când faceţi clic pe `item`-ul respectiv din meniu. În exemplele de mai sus suntem în modul `insert` şi ceea ce i se spune, de fapt, editorului Vim este să insereze textul respectiv în fişier.

1.2.2.1.1.1 Submeniuri Dacă mergeţi la meniul standard `Edit` şi apoi la `File Settings`, vedeţi cum apare un submeniu. Puteţi exersa crearea de submeniuri în Vim meşterind liniile de cod care urmează:

```
:imenu Simplu.nod.text1 Am realizat submeniul 1.
:imenu Simplu.nod.text2 Am realizat submeniul 2.
```

Fiţi atente şi atenţi la faptul că trebuie să daţi din nou clic pe pictograma cu omuleţul care aleargă! Vim trebuie ţinut la curent cu schimbările pe care le-aţi făcut.

Dacă vreţi să luaţi totul de la început, închideţi fişierul text. Redeschideţi apoi fişierul şi repetaţi operaţiile de creare a meniului.

1.2.2.1.1.2 Acceleratori Mai mult ca sigur, ştiţi deja la ce slujesc literele subliniate din titlurile meniurilor. Combinaţia de taste Alt+E, de pildă, deschide meniul Edit. Apoi este suficient să apăsăm tastele care corespund literelor subliniate pentru a avansa prin meniu sau pentru a executa comanda asociată respectivului articol din meniu.

Cum pot fi realizaţi acceleratorii? Foarte simplu. În linia de cod de mai jos `m` este transformat într-un accelerator.

```
:imenu Si&mplu.&accelerat Am realizat un meniu accelerat.
```

Puneţi semnul & în faţa literei care desemnează acceleratorul. &

Trebuie să fiţi atente şi atenţi la posibilele conflicte cu alte meniuri sau inconsistenţe. Dacă nu folosiţi des acceleratorii, atunci n-are probabil rost să-i creaţi.

1.2.2.1.1.3 Bara cu instrumente B. Moolenaar explică în manual cum poate fi modificată bara cu instrumente.[75]

[75] Moolenaar[7, §42.4].

Efectul obţinut prin modificarea barei cu instrumente vă va convinge probabil că Vim este efectiv flexibil. Dacă aveţi pasiunea interfeţelor grafice, veţi pune desigur cel puţin o pictogramă proprie pe bara cu instrumente.

Să vedem un exemplu practic. Creaţi un fişier `vim` care conţine următorul cod:

```
1 :tmenu ToolBar.LtxCmd Comandă LaTeX
2 :amenu ToolBar.LtxCmd <Esc>\{}<Left><Left>
```

Prima linie de cod spune editorului ce mesaj să afişeze în mica fereastră care apare când ducem mouse-ul pe pictograma de pe bara cu instrumente. A doua linie include comanda. Am pus `amenu` pentru ca pictograma să fie afişată în toate modurile Vim[76]. Comanda ca atare este în cel de-al treilea bloc. Ea îi spune lui Vim să insereze scheletul unei comenzi LaTeX, format din semnele `\{}` şi apoi să mute cursorul după bara oblică inversă. Pe `<Esc>` îl puteţi ignora deocamdată. Nu-l eliminaţi însă.[77]

La urmă, dar nu în cele din urmă, examinaţi blocul din mijloc al celor două linii de cod de mai sus. Scrieţi `ToolBar` şi nu altceva. Nu schimbaţi nici majusculele în minuscule sau invers. Altfel Vim nu înţelege că vreţi să puneţi ceva pe bara sa cu instrumente. Vă trebuie însă şi o imagine pentru pictograma de pe bară.

Eu am creat un dosar (un folder) `bitmaps` în dosarul standard `vimfiles`. Vim citeşte o variabilă de mediu care-i spune unde sunt dosarele sale.[78] Dosarul `bitmaps` trebuie să fie undeva unde caută Vim. Eu prefer `vimfiles` pentru că aici stau doar fişierele create sau instalate local (de către mine) şi nu cele din instalaţia standard Vim. De asemenea, nu schimbaţi numele lui `bitmaps`. Puneţi în `bitmaps` imagini de tip `bmp` cu dimensiunea de 18 x 18 pixeli.

Imaginile din dosarul `bitmaps` sunt folosite pentru pictograme. Numele lor trebuie să fie exact acelaşi cu cel care urmează după `ToolBar` şi punct. În exemplul nostru, imaginea trebuie să aibă desigur numele `LtxCmd.bmp`.

Dacă nu ştiţi nimic despre imaginile folosite de către programele de computer, puteţi afla câte ceva aici în §3.1.

[76] Acel `a` din `amenu` vine de la englezescul *all modes* (toate modurile).

[77] Citiţi despre caracterul aparte al lui `amenu` în manualul lui Moolenaar[7, p.207]. Practic, acel `<Esc>` este echivalent cu apăsarea tastei Esc, necesară pentru a anula efectul lui Ctrl-O introdus automat în modul `insert` de către `amenu`. Noi vrem doar să introducem un text, nu să executăm o comandă şi abia apoi să revenim la modul `insert`.

[78] La instalare sunteţi întrebate unde vreţi să puneţi `vimfiles`.

1.2.2.1.1.4 Meniurile standard Morala de bază în cazul meniurilor standard este să nu le modificaţi. Daţi meniurilor dumneavoastră alte nume. Folosiţi alte taste pentru accelerare.

Morala aceasta fundamentală nu vă interzice însă să examinaţi felul în care sunt construite meniurile standard. Mergeţi în dosarul `Vim`*xx*.[79] Căutaţi fişierul `menu.vim` şi studiaţi-l.

Nu folosiţi unele dintre părţile meniurilor standard? Nu folosiţi, de exemplu, din `Tools` secţiunea a treia, cea care se referă la compilare? Nu este nevoie să distrugeţi ceva în `menu.vim`. Eu am pus în fişierul de configurare al Vim următoarele rânduri:

```
:aunmenu Tools.&Make
:aunmenu Tools.&List\ Errors
:aunmenu Tools.L&ist\ Messages
:aunmenu Tools.&Next\ Error
:aunmenu Tools.&Previous\ Error
:aunmenu Tools.&Older\ List
:aunmenu Tools.N&ewer\ List
:aunmenu Tools.Error\ Window
:aunmenu Tools.Set\ Compiler
```

Aceste comenzi fac să dispară din meniu porţiunile pe care nu le folosesc. n-am distrus însă nimic pentru a putea restaura lesne meniurile iniţiale.

Atenţie la comenzile de mai sus. Ele au între litera care indică modul Vim şi `menu` secvenţa `un`. Restul se înţelege de la sine.

Mai există o metodă blândă de schimbare a meniurilor standard. Putem transforma toată linia de cod (de program deci) într-un *comentariu*. Vim ignoră comentariile. Comentariile sunt rânduri (în sens logic, nu vizual) care încep cu ghilimele duble. Exemplul următor ne arată cum scap de linia de separare rămasă în plus în meniu după comenzile de anulare de meniuri de mai sus:

comentarii în Vim

```
"an 40.520 &Tools.-SEP3-<Nop>
```

Singura modificare pe care am făcut-o a fost inserarea ghilimelelor duble la început de rând. Evident, este uşor să le elimin la nevoie.

Tot cu ghilimele duble am scăpat şi de unele pictograme:

```
"  an 1.250 ToolBar.Make :make<CR>
"  an 1.270 ToolBar.RunCtags :!ctags -R .<CR>
```

[79] ele două *x*-uri stau pentru numerele versiunii Vim.

1.2.2.2 Mişcările cursorului Vim

Cu un simplu clic pe butonul din stânga mouse-ului putem schimba poziţia cursorului în fereastra Vim. Foarte comod! Câtuşi de puţin dacă vrem să scriem un meniu Vim care să facă şi altceva decât să insereze text.

Vim este un editor, după cum am văzut, de colecţii de coduri pentru tot felul de semne. În aceste condiţii ne interesează prea puţin punctele de pe ecran şi afişarea pe ecran a semnelor. Ne-ar interesa să ne deplasăm cumva de la un cod la altul. Este ceea ce explicăm în continuare.

1.2.2.2.1 Mişcări ale cursorului în modul normal

În modul normal, care este modul în care putem da comenzi apăsând pe taste, există o serie de taste care controlează mişcările cursorului. Este destul să indicăm aceste taste într-un script Vim pentru ca să obţinem acelaşi efect ca atunci când apăsăm fizic pe taste.

1.2.2.2.1.1 Micile mişcări

Dacă aţi studiat felul în care se creează meniurile, nu este greu să folosiţi următoarele rânduri într-un script Vim:

```
menu Cursor.<- h
menu Cursor.^ k
menu Cursor.v j
menu Cursor.-> l
```

După cum se observă mai sus, cele două puncte din faţa cuvântului `menu` nu sunt absolut necesare. Evident dacă aţi folosi tastatura, ar trebui să apăsaţi pe tasta cu două puncte pentru a trece în modul comandă în linie.

Meniul pe care-l creaţi nu este, în practică, de mare folos. După cum sugerează şi semnele noastre, săgeţile pot face lesne acelaşi lucru. Importantă este însă descoperirea efectelor comenzilor.

1.2.2.2.1.2 Numărarea paşilor

Ne putem însă mişca nu doar cu paşi de melc. Putem face un fel de salturi de cangur.

```
menu Cursor.<- 3h
menu Cursor.-> 3l
```

Cifra din faţa literei indică de câte ori se aplică comanda. Fişierul meu `woo.txt` este prea strâmt pentru salturi peste rânduri, dar ideea se aplică şi-n acel caz.

Meniurile nu se compară aici ca eficienţă cu tastarea directă în modul normal, dar jucăria ar trebui să poată sugera posibilităţile de acţiune ale Vim. Eu unul mă joc pornind de la litera `w` de pe rândul 2 din `woo.txt`.

1.2.2.2.1.3 De la un cuvânt la altul Deplasarea de la un cuvânt la altul este o chestiune mult mai subtilă. Pentru început puteţi studia posibilităţile următoarelor meniuri:

```
1 "comanda w deplasează cursorul la următorul început de cuvânt
2 menu Cursor.w w
3 "comanda e deplasează cursorul la următorul sfârşit de cuvânt
4 menu Cursor.e e
5 "comanda b deplasează cursorul la precedentul început de cuvânt
6 menu Cursor.b b
7 "comanda ge deplasează cursorul la precedentul sfârşit de cuvânt
8 menu Cursor.ge ge
```

Comentariile de pe rândurile 1, 3, 5 şi 7 oferă explicaţiile necesare. Restul este o chestiune de exerciţiu.

Ca şi-n cazul paşilor mici, se pot şi aici număra paşii care urmează să fie făcuţi. Putem da înapoi, de pildă, cu trei cuvinte.

1.2.2.2.1.4 Deplasările pe distanţe lungi Putem să mergem însă de la un capăt la altul al rândului. Putem merge de la un capăt la altul al fişierului.

```
1 "comanda ^ deplasează cursorul la început de rând (logic)
2 menu Cursor.capRand ^
3 "comanda b deplasează cursorul la sfârşit de rând
4 menu Cursor.sfarsitRand $
5 "comanda gg deplasează cursorul la început de fişier
6 menu Cursor.capFisier gg
7 "comanda G deplasează cursorul la sfârşit de fişier
8 menu Cursor.sfarsitFisier G
```

Asemenea deplasări sunt foarte utile într-un script Vim. Putem spune deja că scripturile Vim sunt programe veritabile. Uneori vrem, de pildă, să căutăm ceva prin tot fişierul. Atunci are sens să mergem la începutul fişierului şi să începem căutarea.

1.2.2.2.2 Mişcările cursorului în modul insert În scripturile Vim putem comanda mişcările cursorului şi fără să fim în modul normal (cel în care dăm comenzi). Metoda este limpede dacă veţi construi un meniu folosind rândurile de cod Vim de mai jos:

```
imenu Cursor.Stanga <Left>
imenu Cursor.CuvantStanga <C-Left>
imenu Cursor.Sus <Up>
imenu Cursor.Jos <Down>
imenu Cursor.Dreapta <Right>
imenu Cursor.CuvantDreapta <C-Right>
imenu Cursor.CapRand <Home>
imenu Cursor.SfarsitDeRand <End>
```

Meniul este activ în modul insert.

Se observă lesne că denumirile sunt mai sugestive (pentru cei care ştiu limba engleză). Nu omiteţi parantezele unghiulare! Ele fac parte din limbajul în care sunt scrise programele Vim.

O explicaţie aparte necesită doar `<C-Left>`. Ideea este că `C` ne arată că ţinem tasta Ctrl apăsată. Similar, `S-Left` conţine o indicaţie de apăsare a tastei Shift. Evident, în programe, aceste apăsări de taste sunt apăsări virtuale, nu reale.

Comenzile `<Left>`, `<Up>` şi celelalte pot fi folosite şi-n modul normal. Ele pot fi precedate de un contor al paşilor doar în modul normal. În modul insert comanda `3<Left>` ar avea drept rezultat inserarea cifrei `3` în text.

`<Esc>`

1.2.2.2.2.1 Trecerea de la un mod la altul Pentru a trece de la un mod la altul putem folosi în comenzi pe `<Esc>`. De pildă, dacă suntem în modul normal, `<Esc>i` sau `<Esc>a` fac trecerea la modul insert. Invers, pentru a trece din modul insert în modul normal, putem folosi pe `<Esc>`.

1.2.2.3 Modificarea fişierelor

Putem modifica un fişier prin adăugarea de text. Putem modifica un fişier prin inserarea de text. Asta este tot ce ştim dacă stăpânim doar mişcările prin fişier.

Pentru a copia porţiuni de text sau pentru a le şterge, pentru a înlocui porţiuni de text ar trebui să stăpânim alte tehnici.

1.2.2.3.1 Regiştrii de memorie Presupunerea noastră constantă este că Vim este configurat pentru lucrul în manierea MS Windows. Oricine a inspectat bara cu instrumente din Vim, a remarcat trei pictograme care servesc la modificarea textului: o foarfecă, două foi de hârtie şi un clipboard. Se pot folosi, de asemenea, combinaţiile Ctrl+X Ctrl+C Ctrl+V pentru a obţine aceleaşi efecte ca şi prin clicuri pe pictogramele amintite.

Clipboard-ul standard din Windows are însă o mare limită. Nu ţine minte decât o singură porţiune de text o dată. Are, ca să zicem aşa, o memorie de scurtă durată.

Creaţi însă un meniu Vim folosind următoarele rânduri de cod Vim:

```
"Copie în registrul a
menu Redactor.copieA "ay
"Scrie din registrul a
menu Redactor.scrieA "ap
"Copie în registrul b
menu Redactor.copieB "by
"Scrie din registrul b
menu Redactor.scrieB "bp
"Copie în registrul c
menu Redactor.copieC "cy
"Scrie din registrul c
menu Redactor.scrieC "cp
```

Eu m-am jucat cu fişierul `woo.txt` în felul următor: am creat un rând suplimentar gol; am selectat cuvântul „bibliografie“; meniul `Redactor` s-a activat şi am copiat textul selecta în registrul (sectorul) de memorie `a`; am pus apoi cuvântul „filosofic“ în registrul de memorie `b` şi cuvântul „woo“ în registrul `c`. Am folosit linia liberă pentru a plasa acolo conţinutul celor trei regiştri de memorie. Fiţi atenţi însă la faptul că meniul `Redactor` este activ doar în modul normal.

Să analizăm acum comenzile din meniu. Apoi trebuie să ne lămurim ce rol au ghilimelele duble. Ele trebuie folosite în modul normal (modul comenzilor) pentru a-i spune lui Vim să aştepte restul comenzii. De pildă, dacă am selectat text în modul vizual al Vim, putem da o comandă de genul `"dy`. Vim pune porţiunea selectată în registrul `d`. Cum? Acel `y`, care vine de la termenul *yank* este cheia comenzii respective.[80]

yank

Ce regiştri putem folosi? Vim are nouă tipuri de regiştri.[81] În practică, la început veţi folosi probabil regiştrii numerotaţi, de la *0* la *9*, şi regiştrii care au drept nume o literă, de la *a* la *z*, respectiv de la *A* la *Z*. În orice caz, cel mai simplu este să puneţi text în regiştrii care au drept nume o literă.

1.2.2.3.2 Copiere inteligentă Programarea se învaţă cel mai bine meşterind. S-ar putea ca ucenicul vrăjitor să se întrebe dacă

[80] Pentru copiere vezi manualul lui Moolenaar[7, §4.6].

[81] Vezi în documentaţia Vim fişierul `change.txt` pentru o descriere completă.

nu-r putea renunţa la ghilimelele duble. Nu în meniurile de mai sus. Cu selecţiile am putea lucra, dar, în modul normal, un `a`, de pildă, ne-ar duce în modul insert. Vim nu-r şti că este vorba despre registrul de memorie. Şi totuşi ucenicul vrăjitor are dreptate.

Există o soluţie. Meşteriţi meniuri cu ajutorul următoarelor rânduri de cod Vim:

```
"Copie un cuvânt
menu Redactor.copieCuvant bye
"Copie (inclusiv) până la acolada din dreapta
menu Redactor.copiePanaLaAcoladaDreapta yf}
```

Dacă *yank* n-are o selecţie cu care să lucreze, atunci va aştepta o comandă care să-i spună ce să copieze.

Mergeţi în `woo.txt` la prima acoladă şi copiaţi până la acolada din dreapta cu ajutorul meniului. Apăsaţi tasta P în modul normal pentru a vedea rezultatul. Observaţi că s-au copiat şi acoladele. Cum? Secvenţa `f}` este o comandă care-i cere editorului Vim să găsească[82] acolada din dreapta.

Dacă nu vreţi şi acoladele, puneţi cursorul pe cuvântul „woo“ şi copiaţi cu ajutorul meniului cuvântul. Puteţi apoi să vă gândiţi cum este realizată comanda.

Hm, mormăie nemulţumit ucenicul vrăjitor. Dacă nu este un singur cuvânt între acolade! Atunci foloseşte:

```
"Copie până la acolada din dreapta
menu Redactor.copiePanaLaAcoladaDreapta vf}hy
```

Selecţia textului este făcută chiar în interiorul comenzii.

1.2.2.3.3 Inserare de text în stil Vim

Dacă *yank* este un termen pe care nu-l auzim în lumea Windows, în schimb `p`-ul folosit pentru a pune text aminteşte de *paste*-ul din Windows.

atenţie la diferenţa dintre minuscule şi majuscule

Vim distinge însă între două feluri de a pune text, după cum se vede mai jos:

```
"Plasează (după cursor) ceea ce ai copiat
menu Redactor.plaseazaDupa p
"Plasează (înainte de cursor) ceea ce ai copiat
menu Redactor.plaseazaInainte P
```

Toată diferenţa este dată de folosirea unei litere minuscule sau a uneia majuscule.

[82] În limba engleză *find*.

1.2.2.3.4 Decupare/eliminare de text în stil Vim Vim poate şterge o singură literă. Bănuiesc însă că v-ar interesa mai mult eliminările masive de text. Meşteriţi meniurile următoare şi veţi vedea ce poate să facă Vim:

```
"Ştergem o literă
menu Sterge.x x
"Şterge până la sfârşitul cuvântului
menu Sterge.dw dw
"Şterge până la sfârşitul celui de al treilea cuvânt
menu Sterge.d3w d3w
"Şterge un rând
menu Sterge.dd dd
"Anulează efectul comenzii anterioare
menu Sterge.undo u
"Anulează anularea unei comenzi
menu Sterge.redo <C-R>
```

Am adăugat şi capacităţi de a elimina dezastrele produse de ştergeri sau anulări de ştergeri.

Cum decupăm? Foarte simplu. Mergeţi la începutul unui cuvânt şi ştergeţi (decupaţi) cu ajutorul comenzii `dw` din meniu. Mergeţi în alt punct din fişier şi, în modul normal, apăsaţi tasta P.

Ce facem dacă vrem să decupăm un cuvânt fără să mergem la începutul său? O soluţie este:

```
"aw este text-object (adică un cuvânt)
menu tObiect.unCuvantSters daw
```

Subiectul este însă cam avansat pentru ambiţiile acestei anexe.[83]

1.2.2.4 Meniuri care fac viaţa mai uşoară

Meniurile de până acum au avut mai mult un rol pedagogic. Ca atare, nu sunt foarte utile. La ce este însă cu adevărat bun un meniu? Este bun atunci când ai de tastat o secvenţă mai complicată de comenzi sau una simplă, dar pe care ai uitat-o.

Mai ţineţi minte cum se numără cuvintele din fişier? Probabil că nu. S-ar putea să găsiţi meşterirea următoarelor meniuri utilă sau chiar foarte utilă:

```
"Numără cuvintele din fişier
menu Util.numara g<C-G>
```

[83] A se vedea pentru detalii manualul lui Moolenaar[7, §4.8].

```
"Aflaţi în ce loc din fişier vă aflaţi
menu Util.loc :set noruler<CR><C-G>
"Ca să puneţi ruler-ul la loc
menu Util.ruler :set ruler<CR>
"Găsirea perechii unei paranteze
menu Util.perecheaParantezei %
"Schimbă minusculele în majuscule şi invers
menu Util.schimbaCasaDeLitere ~
"Schimbă un cuvânt întreg;
"mai bine cu aw decât cu bve~
menu Util.schimbaCasaDeLitereCuvant vaw~
"Schimbă între ele două litere
menu Util.xp xp
```

Meniurile de pe rândurile 4 şi 6 vă permit să vă aflaţi locul în care sunteţi în fişier. Meniul de pe rândul 6 repară eventuala eroare produsă de meniul de pe rândul 4, dacă vă place să aveţi permanent *ruler*-ul în faţa ochilor.

perechea parantezei

Găsirea perechii unei paranteze este ceva vital când folosim programe precum LaTeX. S-ar putea să socotiţi utilă instalarea permanentă a scriptului `matchit.vim`. Îl găsiţi în dosarul `macros` din distribuţia standard a Vim.[84]

casa de literă

Schimbarea casei de literă[85] este adesea o problemă.

Această anexă am prezentat Vim dintr-o perspectivă elementară. Cunoştinţele de bază de aici vă permit să editaţi însă fişiere de tip text simple. Problemele specifice fişierelor `tex` sau `bib` vor fi tratate separat. De asemenea, căutarea într-un fişier va fi discutată în alte anexe.

1.3 Expresiile regulate

Dincolo de orice teorie sofisticată, ideea care stă în spatele expresiilor regulate poate fi lesne explicată printr-un exemplu. Să zicem c-am ieşit în oraş şi studiez numere de înmatriculare ale maşinilor. Unele numere de înmatriculare încep cu una-două cifre şi continuă cu o literă. Dar sunt vechile numere! Altele au o literă urmată de mai multe cifre. Par a fi numere provizorii. Majoritatea maşinilor au numere de înmatriculare noi. Cum le disting? Aş putea oferi o descriere de acest gen: la început sunt una sau două litere, apoi două cifre, urmate de trei litere.

[84] Instalarea ca atare este banală. Este bine să puneţi `matchit.vim` în dosarul `plugin` din `vimfiles`. Instalarea ajutorului pentru acest script este descrisă în manualul lui Moolenaar[7, §5.5].

[85] Acesta este termenul tehnic din poligrafie. Vine de la franţuzescul „casse“.

Aş putea oare caracteriza cumva formal un număr de înmatriculare? Care este gramatica adecvată?

Mă pot gândi la numerele de înmatriculare ca la nişte expresii. Fiecare expresie conţine numere şi litere (majuscule). Există o mulţime a tuturor acestor expresii. Toată problema constă în separarea din mulţimea mare a acestor expresii a acelor expresii care constituie mulţimea numerelor de înmatriculare (posibile).

În ciuda posibilelor asociaţii greşite pe care le poate sugera numele, o „expresie regulată“ nu este o **expresie** în sensul de mai sus, ci un **tipar** (un şablon) care surprinde **regularităţile** care disting o expresie de alte expresii.

apartenenţa la o mulţime de expresii

Din explicaţia de mai sus decurge ideea că unei expresii regulate îi corespunde o mulţime de expresii. Expresia regulată este un şablon care ne permite să decidem dacă o expresie aparţine sau nu mulţimii respective. Aceasta este ideea care stă la baza gramaticilor cu expresii regulate.

La ce ar fi bune aceste şabloane? În primul rând, fără ele nici nu poate fi vorba de căutări mai sofisticate prin fişiere. De asemenea, orice program de prelucrare de texte, de pildă, nu poate trece de un stadiu foarte rudimentar dacă n-are posibilitatea să distingă între diverse expresii.

1.3.1 Definirea şabloanelor

Unii scriu „filosofie“, alţii „filozofie“. Eu unul nu cred că este o problemă de substanţă în spatele acestei diferenţe. Exemplul este însă bun pentru a explica modul de căutare într-un text cu ajutorul unor şabloane sau tipare.

Creaţi un fişier care cuprinde următorul text:

```
filosofie filozofie bibliofil
fiilosofie fiiilozofie b37fil
fiiiilosofie fiiiiilozofie
filoçofie flosofie f+losofie
```

Un nume sugestiv pentru acest fişier ar „test.txt“.

Pentru o primă căutare în fişierul de mai sus folosim un tiparul următor:

```
/filo[sz]ofie/
```

Cu tiparul se potrivesc atât `filosofie`, cât şi `filozofie`. Parantezele drepte indică faptul că oricare dintre literele din paranteză s-ar

bara oblică

putea potrivi. Barele oblice sunt necesare în Vim pentru căutări şi-nlocuiri de şiruri de semne. Le vom folosi în continuare şi pentru că marchează clar graniţele unui tipar.

Deschideţi fişierul `test.txt` în Vim. Puneţi cursorul la începutul fişierului.[86] Treceţi în modul comandă în linie şi tastaţi tiparul de mai sus. Când execută comanda, Vim mută cursorul la prima apariţie a unui şir de semne care corespund tiparului şi evidenţiază cu ajutorul fondului colorat tot ce corespunde cu tiparul dat.[87]

Vim foloseşte tipare pentru a filtra un text, separând ceea ce corespunde tiparului de ceea ce nu corespunde. Mulţimea expresiilor care corespund tiparului este evidenţiată prin colorare.

1.3.1.1 Logica şabloanelor

variabilă

Persoanele pasionate de logică vor descoperi cu plăcere modul de construire a tiparelor. În exemplul de mai sus este deja prezentă ideea de **lacună** într-un text. Astfel concep logicienii ideea de **variabilă**. Formal, o variabilă este o lacună dintr-un text; lacuna respectivă poate fi umplută cu un anumit material. Mai sus umplerea era limitată la literele `s` sau `z`. Puteam scrie `[a-z]` şi atunci domeniul de valori al variabilei ar fi cuprins literele de la `a` la `z`.

punctul

Dacă punem însă un simplu punct, tiparul devine `/filo.ofie/` şi Vim va evidenţia şi cuvântul „filoçofie“. Punctul pus într-un tipar corespunde cu ideea de lacună pentru un semn. Tiparul `/f.l/` nu va produce nici o evidenţiere în „flosofie“. Nici tiparul `/f[.]l/` nu va evidenţia nimic! Ar trebui să avem în text pe „f.losofie“ sau ceva de acest gen pentru ca să se producă o evidenţiere.

Din nou, persoanele interesate de logică pot să mediteze la **cuantificarea** implicată de parantezele drepte puse în tipar. În cuvinte aceasta ar însemna *pentru oricare dintre semnele din paranteză*.

\?

Punctul pus în tipar cere ca lacuna să fie completată efectiv. Pentru a admite şi zero apariţii ale unui semn trebuie, în setarea care este dată prin instalarea Vim,[88] trebuie scris un tipar de genul `/f\?losofie/`.

Bara oblică spre stânga din `\?` nu ţine de construcţia standard a tiparelor. Ea depinde de modul în care este setat Vim. Bara îi spune

[86] Pentru că modul căutării va fi **înainte**. Pentru căutări înapoi ar trebui să folosim un semn de întrebare în locul primei bare oblice.

[87] Atenţie! Verificaţi în meniul `Edit` la `Global Settings` dacă este dată comanda de activare-dezactivare a evidenţierii.

[88] Ucenicii vrăjitori care au modificat setările standard ale Vim trebuie să consulte documentaţia pentru a putea obţine efectele dorite cu tiparele folosite.

lui Vim că nu trebuie să ia în sens literal semnul care urmează. Observaţi diferenţa dintre `/f+losofie/` şi `/fi\+losofie/`. \+

Dacă vreţi un mod standard de formare a tiparelor, atunci trebuie să puneţi o opţiune `\v` chiar la începutul tiparului. Experimentaţi cu `/\vfi+losofie/`. Pentru alte opţiuni care afectează tiparele sau pentru eventuale nepotriviri cu explicaţiile date aici citiţi documentaţia Vim. **modul standard**

Am introdus deja mai sus o altă cuantificare în tipare. Ea ar putea fi citită în felul următor: una sau mai multe apariţii ale semnului anterior. Astfel, tiparul `/\vfi+losofie/` caută „filosofie“ cu una sau mai multe apariţii ale lui `i`.

Cuantificarea `*` funcţionează ca şi `+`, dar admite şi situaţia când în lacuna aflată în raza sa de acţiune nu a fost completată cu nici un semn. Setarea normală a lui Vim[89] ne permite să ne dispensăm de opţiune `\v` în acest caz. *

Putem cuantifica şi numeric. Tiparul `/\vfi{1,3}losofie/` va conduce la filtrarea acelor cazuri în care `i` apare cel puţin o dată sau cel mult de trei ori. Putem omite unul dintre numere. **cuantificare numerică**

Ce facem însă dacă, folosind un tipar standard, vrem să-l căutăm chiar pe `+`. În acest caz, vom scrie, de exemplu, `/\vf\+losofie/`. Evident, opţiunea `\v` am pus-o de dragul Vim. Oricum, în Vim, cum am văzut mai sus, putem să-l recunoaştem pe `+` direct.

Dacă vrem să găsim barele oblice într-un text, atunci trebuie să folosim `\\`, respectiv `\/`.

Tiparele de căutare pot fi foarte sofisticate. Putem indica, de exemplu, dacă tiparul trebuie căutat la început de rând. Pentru aceasta scriem `/\v^fil/`. Observaţi diferenţa faţă de `/\vfil/`. **început de rând**

Din nou, folosit între `[]` semnul `^` are cu totul alt rol. Dacă scriem `/[^0-9]/`, aceasta înseamnă că suntem în căutarea a orice altceva în afară de cifre. Dacă exeperimentaţi cu fişierul test.txt, vedeţi cum Vim subliniază tot, în afară de „37“. Spaţiile sunt şi ele semne! **excludere**

Dacă vrem doar cuvintele care încep cu `f`, trebuie să construim un tipar de genul `/\v<f[a-z]>/`. Practic cuvântul este pus în paranteze unghiulare. Dacă nu punem `\v`, atunci, în Vim, trebuie să punem bara oblică spre stânga ca în `\<` şi `\>`. Când experimentăm observăm că „filoçofie“ nu este evidenţiat. Iar `+` trece drept graniţă a unui cuvânt.[90] **cuvinte**

[89] Ucenicii vrăjitori se pot juca folosind comenzile `:set magic` şi `:set nomagic`; setarea recomandată este `magic`, dar nici ea nu este, cum am văzut mai sus, suficientă pentru o utilizare standard a tiparelor.

[90] Cu titlu de exerciţiu, puteţi încerca să aflaţi ce efect are `/\v<f[a-z,ç]>/`.

1.3.1.1.1 Conectori pentru şabloane Ca şi-n logica propoziţiilor, putem folosi conectori pentru obţine şabloane compuse. Un tipar poate avea, de pildă, două ramuri, precum `/\vfil|biblio/`. Bara verticală o putem citi „sau“. Dacă experimentăm acest tipar pe fişierul test.txt, observăm desigur cum cuvântul „bibliofil“ este evidenţiat în întregimea sa.

Trebuie să fim atenţi la spaţii. Spaţiile puse în jurul barei verticale vor fi interpretate ca fiind o parte a tiparului. Dacă ni se pare confuză expresia, putem pune paranteze rotunde în jurul tiparelor conectate.

1.3.2 Utilizarea şabloanelor

Dacă nu scriem programe avansate, în practică, n-avem nevoie de structuri ale tiparelor mai complicate decât cele descrise mai sus.

substituţie

La ce am putea, de altfel, folosi tiparele, în afară de căutări în fişiere? Tiparele sunt utilizate frecvent în operaţiile de substituire a unui semn sau şir de semne cu alt semn sau şir de semne. De pildă, pot înlocui pe „37“ cu „73“ în fişierul test.txt cu ajutorul următoarei comenzi: `:%s/37/73/`. Se observă acel `s` pus în faţa tiparului şirului care va fi substituit, precum şi felul în care este plasat şirul cu care se substituie.[91] Semnul `%` spune lui Vim să caute în tot fişierul.

Comanda `:%s/\vfi+l/fil/` nu duce chiar la efectul scontat! Vim caută în tot fişierul, dar nu substituie peste tot. Pentru aceasta trebuie să punem după şirul cu care se substituie o opţiune. Dacă vrem o substituire globală, punem `g`. Comanda noastră este atunci ceva de genul `:%s/\vfi+l/fil/g`. Putem apăsa Esc şi experimenta. Pentru a anula efectul comenzii apăsaţi, în modul comandă, `u` sau faceţi clic pe pictograma Undo[92].

Acum puteţi face un prim pas către programare. Programarea nu este, în fond, decât arta de a proiecta algoritmi, de a descoperi paşii care duc la rezolvarea unei probleme. Fie următoarea problemă: în `test.txt`, substituiţi pe „s“ din „filosofie“ cu „z“ şi pe „z“ din „filozofie“ cu „s“. N-ar fi rău să nu vă uitaţi o vreme la soluţia care urmează şi să experimentaţi propriile idei.

Exerciţiul nu este chiar banal dacă n-aţi scris niciodată programe. Dacă substituiţi direct pe „z“ cu „s“ sau invers, ajungeţi precum hoţii derutaţi de Morgiana în *O mie şi una de nopţi*. Hoţii au făcut un

[91] Evident, la substituire trebuie să indicăm un şir (posibil vid), nu tiparul unui şir. De altfel, dacă punem tiparul, Vim va substitui cu tiparul ca atare.

[92] Numele pictogramelor apare scris sub ele dacă deplasaţi cursorul Windows pe pictograma respectivă.

semn pe casa unde stătea Ali Baba, dar Morgiana l-a reprodus pe toate casele din cartier. Nu mai ştiţi unde a fost „z“ şi unde a „s“.

substituire intermediară

Soluţia este foarte simplă. Efectuaţi o substituire intermediară. De exemplu, daţi comanda `:%s/filosofie/filoÇofie/g`. Apoi daţi comanda `:%s/filozofie/filosofie/g`. Pasul al treilea, cel final, îl reprezintă `:/filoÇofie/filozofie/g`.

1.3.3 Utilizarea şabloanelor în programul `grep` şi-n TEXnicCenter

grep

Dacă vă întrebaţi ce programe, în afară de Vim, folosesc tipare pentru căutări, cel mai simplu ar fi să instalaţi o unealtă Unix numită „grep“ sau să folosiţi mediul integrat de dezvoltare de programe LaTeX.[93] Mergeţi cu ajutorul 2xExplorer în dosarul în care ţineţi `test.txt` şi chemaţi fereastra MS-DOS. Tastaţi următoarea comandă în fereastra MS-DOS: `grep filo[sz]ofie *`. Comanda are trei părţi: numele comenzii, un tipar după care se face căutarea şi numele fişierului în care se face căutarea. În locul numelui fişierului am pus `*` pentru a arăta că este vorba despre orice fişier (din dosarul respectiv).

Rezultatul este afişarea numelui fişierului în care grep a găsit ceva care corespunde tiparului şi a rândului în care apare şirul care corespunde tiparului. Dacă vreţi şi numărul rândului, puneţi opţiunea `-n` după `grep`. Pentru a vă convinge că grep caută efectiv prin mai multe fişiere, creaţi un alt fişier text în dosarul respectiv şi scrieţi acolo, între altele, „filosofie“ şi/sau „filozofie“. Dacă vreţi instrucţiuni ceva mai detaliate pentru grep, daţi comanda `grep --help > grep.txt`. Această comandă va crea un fişier cu explicaţii. Se observă c-am folosit pe `>`, care redirecţionează rezultatul comenzii. Acesta nu mai este afişat pe ecran, ci este scris într-un fişier.

TEXnicCenter

În TEXnicCenter căutaţi pictograma cu binoclul aşezat peste un dosar (Find in files). Când daţi clic pe această pictogramă apare o fereastră de dialog. Ar trebui să nu vă fie deloc greu să alegeţi tipul fişierelor. De asemenea, este relativ uşor să alegeţi dosarul unde se face căutarea (Directory). Bifaţi butonul de validare Regular expression.

Eu am avut, de exemplu, nevoie să caut contextele în care apărea una dintre formele expresiei „ucenic vrăjitor“. Am folosit şablonul `ucenic[i]* vr.jitor[i]*`. De ce? Voiam să găsesc şi cazurile când

[93] Pentru instalarea TEXnicCenter vezi aici §2.1.2.

expresia era folosită la plural; de asemenea, era mai simplu să nu mă încurc cu diacriticile.[94]

TEXnicCenter construieşte o listă cu toate contextele în care apare expresia. Dând clicuri pe lista respectivă deschideţi fişierul chiar în punctul dorit. Acest mod de căutare de tip `grep` este unul dintre marile avantaje ale TEXnicCenter.

În TEXnicCenter puteţi construi două liste cu contextele în care găsiţi două expresii diferite. Puteţi face comparaţii şi lucra eficient cu fişierele aflate într-un dosar sau chiar în subdosarele acestuia.

1.4 În căutarea surselor

În mod tradiţional răsfoiai cărţi în bibliotecă. Este o plăcere la care eu unul nu aş putea renunţa. Dar astăzi căutăm mai degrabă în biblioteci electronice şi mai ales în imensa bibliotecă electronică denumită Internet.

Să zicem c-am găsit pe Internet *Fraţii Karamazov* sub forma unui fişier text.[95] Deschid fişierul în Vim. Dau un clic pe pictograma cu o foaie şi o lupă pe ea. Apare o mică fereastră. Scriu cuvântul `Inquisitor` şi apăs ENTER. Vim derulează textul şi apare cuvântul-cheie pe care l-am tastat anterior pe un fond galben. Este chiar punctul din text în care Ivan îi spune lui Alioşa că a scris un poem despre Marele Inchizitor.

Evident, sunt foarte interesat de aspectele filosofice ale poemului lui Ivan. Pe moment, trebuie remarcată însă deosebirea principală din perspectiva căutării prin textul romanului: nu încerc să găsesc o pagină (acest concept nici n-are sens aici), ci locul în care apare un cuvânt-cheie.

căutare cu comandă în linie

Pot căuta însă şi cu ajutorul comenzilor în linie. Apăs tasta `Esc` urmată de `/` şi cuvântul-cheie `Peter`. Cursorul se mută ceva mai jos în text în punctul în care este vorba despre Petru cel Mare.

chirilice

În principiu, în textul în limba rusă se caută în acelaşi fel. Ceea ce complică lucrurile sunt literele chirilice. Când deschid textul apar semne incomprehensibile. Merg la meniul Edit şi apoi la Select Font pentru a schimba *script*-ul tipului de literă în Cyrillic. Acum pot citi textul. Mă mulţumesc să selectez un cuvânt din text şi să-l copii în

[94] Puteţi însă pune litere cu diacritice în şabloane. Atâta doar că trebuie să comutaţi pe tastatura cu literele cu diacritice înainte de a deschide caseta de dialog.

[95] Romanul lui Dostoievski este disponibil în rusă; de asemenea, putem găsi traducerea în engleză realizată de Constance Garnett.

fereastra de căutare a Vim, dar numai după ce am dus cursorul la începutul textului. Vim ascultă silitor comanda şi colorează fondul cuvântului respectiv prin tot textul.

Limitele lucrului cu ferestrele Windows se fac însă rapid simţite. Este mult mai sigur să procedez după cum urmează. Apăs Esc şi apoi v. Vim scrie pe ultimul rând `VISUAL`, semn că sunt în modul lui specific de selecţie. Ţin tasta Shift apăsată şi selectez cu săgeata către dreapta ca-n Windows. Când termin de selectat, copiez tot ca-n Windows selecţia (prin clic pe pictograma cu cele două foi suprapuse). Trec în modul comandă în linie, tastez / şi lipesc textul copiat anterior, tot ca-n Windows, cu clic pe pictograma corespunzătoare. Căutarea funcţionează. Pot să repet căutarea trecând în modul comandă în linie şi recurgând la istoria comenzilor păstrată de Vim (apăs săgeata îndreptată în sus).

selecţia vizuală Vim

Pot scrie cuvântul-cheie căutat direct în linia de comandă, dar pentru aceasta trebuie să dispun în Windows sau Vim de tastatura pentru limba rusă.

Puteţi afla mai multe despre căutările cu ajutorul Vim citind *help*-urile. Aici n-avem spaţiu decât pentru discuţia de principiu.

1.4.1 Construirea unei concordanţe

Este limpede că nu toţi cei care se ocupă cu filosofia au şi o pasiune specială pentru logica formală. Mai mult decât atât, s-ar putea să aibă nevoie de căutări amănunţite prin fişiere persoane care studiază texte vechi sau stilul unor filosofi. A le cere să folosească în practică tipare construite de la un capăt la altul de către ele ar fi probabil o utopie.

Pe de altă parte, din ceea ce ştiu din propria practică, atunci când studiezi un text filosofic ai nevoie de o listă cu apariţiile unui cuvânt-cheie în text. Dacă lista aceasta oferă şi contextul în care apare cuvântul-cheie, ai deja ceea ce se numeşte o „concordanţă“.

Puteţi găsi pe Internet un program gratuit de realizare de concordanţe scris de Zdenek Martinek şi Les Siegrist. Numele programului este „Wconcord“.[96]

Instalarea programului Wconcord este cât se poate de simplă. Dacă l-aţi găsit arhivat, îl dezarhivaţi într-un dosar potrivit şi puteţi deja lucra. n-ar mai fi necesară decât o scurtătură într-un dosar din

[96] A se vedea subsecţiunea 1.4.2 pentru o explicaţie a modului în care puteţi găsi programul pe Internet.

Start Menu. Puteţi crea, de exemplu, un dosar pentru `Editare` şi-n el unul pentru `Analiza textelor`.

Wconcord construieşte concordanţele pe baza fişierelor de tip text.[97] Prima operaţie care trebuie făcută cu programul este elaborarea unei liste a fişierelor în care se vor opera căutările.

frecvenţa cuvintelor

Programul are o interfaţă grafică şi este extrem de uşor de folosit. Cel mai simplu lucru este să construim o listă a cuvintelor care apar în text şi să o ordonăm în funcţie de frecvenţă. Am experimentat acest lucru cu textul romanului lui Dostoievki *Fraţii Karamazov*. Alioşa, de exemplu, este menţionat de 1243 de ori. Termenul „moarte“ apare de 121 de ori, în vreme ce „libertate“ apare de 53 de ori.

concordanţe pentru documente `pdf`

Mai interesante decât frecvenţele sunt utilizările în context ale unor termeni-cheie. Am folosit practic acest sistem pentru a analiza textul tratatului lui Ludwig von Mises *Human Action*.Textul disponibil pe Internet este în format `pdf`. Am extras cu ajutorul Adobe Acrobat Reader textul pur şi am folosit sistemul de investigare oferit de Wconcord. Regăsirea punctului din text în formatul `pdf` nu este o problemă dacă folosim în mod adecvat un fragment-cheie de text. Astfel putem stabili cu exactitate toate referirile unui autor la un anumit concept.

Foarte interesante sunt concordanţele în care este identificat contextul în care apar mai mulţi termeni-cheie. Wconcord poate lucra simultan cu cinci termeni-cheie. Putem specifica şi alternative la un termen-cheie. De asemenea, poate fi precizată ordinea în care apar cheile şi distanţa maximă dintre ele.

Wconcord poate lucra, de asemenea, cu formele gramaticale ale unui cuvânt. Pentru aceasta este nevoie să construim noi fişierele cu lista de forme gramaticale.

1.4.2 Arheologie pe Internet

Toate programele despre care este vorba în această carte sunt disponibile pe Internet. Este deci firesc să spunem câteva cuvinte despre modul în care putem găsi un document pe Internet.

Presupun că nu scrieţi eseul şi efectuaţi căutările pe Internet pe acelaşi calculator. Puteţi merge în laboratorul Facultăţii şi veţi găsi acolo o reţea de calculatoare care este conectată la Internet. De altfel, Internetul nu este decât o reţea de reţele de calculatoare care

[97] Extragerea textului este posibilă din orice fişier. De la un format la altul, trebuie văzut însă cum se procedează şi ce program poate să facă acest lucru.

acoperă tot globul. Fiecare reţea mai mică are un calculator care le deserveşte pe celelalte. Termenul englezesc pentru acest calculator este *server*. Dacă reţeaua este mai dezvoltată s-ar putea să existe un alt calculator care are instalate pe el programele care formează poarta către Internet. Termenul englezesc este *gateway*. Calculatoarele acestea nu se disting de celelalte neapărat prin modul în care construite, ci prin sistemul de operare de care dispun.

Categoric, Windows98 este nepotrivit pentru conectarea la Internet. Este vulnerabil la atacurile de pe Internet. Ceea ce-l face atât de plăcut când este folosit pe un calculator separat, uşurinţa cu care avem acces la fişiere devine un handicap când persoane rău intenţionate au şi ele cam acelaşi acces facil la fişiere.

e uşor să distrugi; greu e să construieşti

Oricine poate face pe ucenicul vrăjitor şi vedea cât de vulnerabil este Win98. Mergeţi cu 2xExplorer pe discul C. Nu faceţi însă modificarea care urmează decât dacă aveţi organizarea fişierelor sugerată în acest capitol.[98] Redenumiţi dosarul Program Files prin simpla eliminare a spaţiului din nume. Veţi primi un avertisment, dar atât. Sistemul nu vă împiedică să acţionaţi. Încercaţi acum să deschideţi un fişier cu extensia `html`. Dacă n-aţi modificat locul în care se află Internet Explorer, veţi primi un mesaj care spune că sistemul nu poate descoperi unde se află `iexplore.exe`. Redenumiţi Program Files pentru a-l aduce la forma pe care o ştie sistemul.

Aţi văzut deci că un pirat de pe Internet nu trebuie să fie prea inteligent. Este mult mai uşor să distrugi decât să construieşti.

Calculatoarele cu Win98 sunt cel mult bune în reţea în postura de *clienţi* ai *server*-ului. Acesta este cel care le serveşte cu ceea ce doresc şi are misiunea de a le apăra de atacuri.

În laboratorul Facultăţii, sistemul de operare al serverelor este de tip Unix (GNU/Linux şi Sun Solaris). Cea mai mare parte din clienţi au şi ei sisteme Unix. Pe aceşti clienţi, pentru a vedea fişierele `html` tipice pentru Internet se foloseşte programul Netscape.

Un client Windows foloseşte în mod tipic programul Internet Explorer. În orice caz, sub Win98, puteţi folosi acest program pentru a vizualiza fişiere de tip `html`. Acestea sunt tot fişiere de tip text, dar conţin o mulţime de comenzi folosite de programul de vizualizare pentru a produce imaginea frumoasă de pe ecran. Fişierele `html` conţin, de asemenea, trimiteri către alte fişiere aflate poate la mari distanţe în spaţiu.

În esenţă, chiar şi atunci când vreţi doar să vedeţi ceva pentru o clipă trebuie să descărcaţi fişiere. Este ca şi cum aţi suna pe cineva

[98] În orice caz, să n-aveţi 2xExplorer în Program Files.

la telefon, iar persoana respectivă v-ar dicta un text. Cam tot aşa, un calculator apelează numărul unui alt calculator şi îşi procură fişierul de care este nevoie.

Ca persoană umană ar fi greu să ţineţi minte numerele de pe Internet ale calculatoarelor. De aceea se folosesc adrese umanizate. De exemplu, adresa paginii de pe Internet a acestei cărţi este <`www.fil.unibuc.ro/~solcan/eft/`>.

Dacă vorbiţi la telefon trebuie să ştiţi şi limba persoanei cu care conversaţi. Tot aşa, adresa de mai sus nu este complet funcţională fără o indicaţie privitoare la limbă. Completată astfel, ea devine `http://www.fil.unibuc.ro/~solcan/eft/`.

descărcarea de fişiere de pe Internet

Să zicem că aţi ajuns la pagina de Internet a cărţii şi aţi descoperit un *script*, un program, pe care vreţi să-l descărcaţi pe calculatorul dumneavoastră. În Internet Explorer, puteţi duce cursorul Windows pe numele fişierului respectiv; obţineţi apoi un meniu prin clic pe dreapta. Folosiţi Save Target As... pentru a descărca fişierul. Dacă utilizaţi Netscape, procedura este asemănătoare, numai că recurgeţi la Save Link As....

De multe ori veţi găsi indicaţii cu privire la descărcarea de fişiere (numită în limba engleză *download*). Evident, citiţi condiţiile în care puteţi obţine fişierele. Citiţi licenţele aferente şi avertismentele.

În tot acest proces cel mai important lucru este să ajungeţi însă la pagina dorită. De unde aflăm ce să tastăm după acea magică Address din Internet Explorer? Există un fel de „grep“-uri ale Internetului. Ele se numesc „motoare de căutare“. Aşa cum sugerează şi numele, veţi fi deservite şi deserviţi de o maşinărie. Cu puţină ingeniozitate din partea dumneavoastră, rezultatele vor fi însă din cele mai bune.

motoare de căutare

Cartea de faţă oferă şi ea adrese de pe Internet. Acestea se pot însă schimba. Este mult mai simplu şi mai sigur să apelaţi la un motor de căutare. Tastaţi, de pildă, după Address sau Location, comanda `http://www.google.com` şi citiţi instrucţiunile de utilizare a motorului.

Tehnica de căutare se bazează pe cuvinte-cheie. Îi veţi oferi motorului un şir de cuvinte-cheie şi acesta vă va furniza adrese pe Internet şi o serie de informaţii despre ceea ce găsiţi la adresa respectivă.

Cygwin

Pentru a găsi, de exemplu, **Cygwin** am folosit cuvintele-cheie `cygwin Linux emulation download`. Căutarea a durat $0,24$ de secunde. Am primit, ce-i drept, o listă cu 3630 de adrese. Pare enorm, dar motorul le-a ordonat deja în mod automat. A doua adresă este <`http://www.cygwin.com/`>; a patra este <`http://www.redhat.com/download/cygwin.html`>. Pare rezonabil să citesc informaţiile

generale de la prima adresă şi să aflu la a doua adresă cum pot descărca Cygwin.

Acum se vede de ce este mai raţional sistemul căutărilor decât recursul la o singură adresă găsită în carte. Sunt destul de multe locurile de pe Internet unde putem găsi ceea ce căutăm. Atunci când locul respectiv este organizat sub forma unei colecţii de fişiere şi pentru transferarea lor se foloseşte limbajul `http`, în engleză, se vorbeşte despre un *site*. Cred că este potrivit să folosim în româneşte cuvântul „sit“. Folosim de multă vreme expresia „sit arheologic“. Aici este vorba de „sit pe Internet“. Căutarea pe un asemenea sit presupune câteodată veritabile înclinaţii pentru arheologie. Cele mai mici fragmente de informaţii trebuie exploatate. Vom ilustra acest lucru cu o serie de exemple.

sit

Următorul exemplu este absolut crucial pentru cartea de faţă. De data aceasta căutăm un loc de unde să descărcăm o distribuţie TEX. Cuvintele-cheie folosite sunt doar `TeX distribution Windows98`. Informaţiile despre a cincea adresă arată o legătură cu MikTEX. Găsim astfel un posibil capăt de fir de care să tragem pentru a ajunge, navigând de la o pagină la alta, la <`http://www.miktex.org/`>, pagina de web a distribuţiei TEX descrise şi-n această carte.

TEX

Ucenicii vrăjitori nu vor rezista probabil tentaţiei de a introduce cuvintele-cheie `C++ IDE SourceForge download`. Dacă vor căuta atent ce se potriveşte cu Win98, vor descoperi adresa `http://www.bloodshed.net/devcpp.html`. Cei mai curioşi se pot uita şi la pagina <`http://sourceforge.net/projects/dev-cpp/`>.

C++

Mai sunt câteva programe extrem de utile a căror căutare merită să o ilustrăm aici. Dacă folosim cuvintele-cheie `TeXnicCenter download`, găsim adresa de la care putem descărca mediul integrat pentru crearea de programe LATEX: <`http://www.toolscenter.org/products/texniccenter/download.htm`>.

TEXnic-Center

Dacă utilizăm cuvintele-cheie `weaverSlave HTML editor download`, descoperim adresa de Internet <`http://www.subjective.de/en/weaverslave/index.php`>. De aici se poate descărca un editor de fişiere `html`, `php` şi comenzi SQL, foarte util pentru crearea de pagini web.

editor pentru pagini web

Dacă sunteţi în căutarea unui program pentru statistică, găsiţi o soluţie cu ajutorul cuvintelor-cheie `ViSTA statistics`. Prima adresă din listă este chiar cea a paginii de Internet a Dr. Forrest Young, creatorul programului ViSTA,<`http://forrest.psych.unc.edu/research/vista-frames/welcome.html`>.

ViSTA

Un alt mod de a găsi documente sau programe pe Internet este

pagini cu trimiteri

reprezentat de paginile cu trimiteri către documente sau programe. Dacă navigaţi pe Internet la adresa <`http://www.uni-giessen.de/~ga1007/ComputerLab/concordance.htm`>, găsim o pagină în care se explică pe scurt ce este o concordanţă şi o listă cu programe de creat concordanţe. Pagina aceasta nu oferă însă direct posibilitatea de a descărca programe. La <`http://www.ujaen.es/dep/filing/profesores/alejandro_alcaraz.html`> găsim o pagină care oferă această posibilitate. Puteţi descărca direct arhiva Wconcord de la adresa <`http://www1.ujaen.es/~aalcaraz/HEL/wconcord.zip`>.

1.4.2.1 Programul `wget`

alternativa la *Save Target As...*

O adresă precum cea pentru Wconcord poate fi folosită şi cu una dintre cele mai bune unelte Unix transpuse sub Windows, programul `wget`. Acest program poate fi apelat fie sub Cygwin, fie ca program independent, în funcţie de modul în care l-aţi instalat. Este un program cu comandă în linie. Are o serie de avantaje însă faţă de Save Target As.... Nu trebuie să descarci interactiv zeci de fişiere care sunt menţionate într-o pagină.

Comenzile pentru `wget` sunt de forma `wget [opţiuni] [adresa-pe-Internet] [opţiuni]`. Adresa pe Internet este opţională; putem folosi comanda în forma următoare: `wget -i nume-fişier`. În fişierul folosit vom scrie adresa sau adresele care ne interesează. Astfel nu trebuie să tastăm adresele în linia de comandă.

Opţiunea `-nc` împiedică descărcarea repetată a aceluiaşi fişier. O opţiune extrem de utilă, pe care o putem pune în finalul comenzii este `-k`; ea îi spune lui `wget` să convertească trimiterile. În loc ca trimiterile să fie făcute pe Internet, ele vor fi locale. Fişierul va putea fi consultat fără probleme şi pe un calculator care nu este conectat la Internet.

Opţiunea `-p` este, de asemenea, utilă pentru a vedea pagina când nu suntem conectaţi. Ea îi spune lui `wget` să descarce tot ce este necesar (imagini, sunete) pentru a vedea pagina.

Opţiunile `-r` şi `-l` trebuie folosite cu grijă pentru că ele vor pune wget să descarce fişiere în mod recursiv până la un anumit nivel. Aceasta înseamnă că wget descarcă o pagină, apoi se uită la ce trimite pagina respectivă şi descarcă, până ajunge la nivelul la care i-aţi spus să se oprească. Metoda are un dublu dezavantaj: poate suprasolicita calculatorul de unde vreţi să obţineţi fişiere, forţându-l pe administratorul acestuia să vă blocheze accesul; de asemenea, dacă nu sunteţi prudenţi, vă puteţi trezi c-aţi descărcat un volum

uriaş de date care v-au lăsat fără spaţiu pe discul propriului calculator (şi fără bani, pentru că va trebui să plătiţi furnizorul de servicii Internet). Nivelul până la care merge `wget` trebuie reglat cu maximă grijă. De asemenea, un simplu `Ctrl+c` va stopa `wget`.[99]

O precauţie contra descărcării în exces de fişiere o reprezintă opţiunea `-A` urmată de o listă de extensii de fişiere (separate prin virgule). Acestea sunt singurele tipuri de fişiere care vor fi descărcate. Alternativ, puteţi pune opţiunea `-R` şi specifica tipurile de fişiere care nu vor fi descărcate.[100]

descărcare în reţele instabile

Programul `wget` este extrem de util pe o reţea instabilă. În acest caz, descărcarea de fişiere se întrerupe frecvent. Rămânem cu o bucată de fişier. Fişierele de mari dimensiuni nu pot fi practic descărcate. Folosind opţiunea `-c` putem continua descărcarea din punctul unde s-a rupt fişierul. Continuarea descărcării în acest mod nu depinde numai de `wget`, ci şi de sprijinul pentru această operaţie pe serverul de pe care descărcăm fişierul. Nu toate serverele oferă sprijin pentru continuarea unei descărcări întrerupte.[101]

1.4.2.2 Limitarea ariei căutărilor

Căutarea cu ajutorul cuvintelor-cheie este deosebit de eficientă. Un şir de cuvinte-cheie este asemenea unei conjuncţii de condiţii. Nu este nevoie să fii expertă sau expert în logică pentru a-ţi da seama că în acest fel se limitează lista cu adrese pe care o obţinem de la motorul de căutare şi, implicit, aria căutărilor noastre prin această listă.

reducerea ambiguităţii

Există însă o problemă a cuvintelor-cheie. Ca orice cuvinte, acestea pot avea o doză de ambiguitate. Există, de pildă, un limbaj de programare numit `python`. Ce se întâmplă dacă folosim acest cuvânt pentru o căutare pe Google? Vor veni adrese cu situri despre limbajul de programare, dar şi cu situri despre şarpele cu acelaşi nume.

[99] `Wget` este un program din lumea Unix, unde acesta este modul de a opri forţat rularea unui program. Programele MS-DOS sunt oprite, probabil, de `Ctrl+Break`. În Win98 apăsaţi `Ctrl+Alt+Del` şi opriţi procesul respectiv.

[100] Din practică ştiu că se întâmplă ca o persoană să aibă în pagina sa texte pe care aş vrea să le citesc şi care nu ocupă mult loc pe disc, dar şi fişiere enorme cu muzică sau filme, de care n-am nevoie şi care ar putea epuiza spaţiul de pe disc. Sub sistemele Unix sau WindowsNT, în mod normal, utilizatorii obişnuiţi au alocată doar o porţiune limitată de disc. S-ar putea ca un film să nici să nu încapă pe zona care le-a fost alocată.

[101] Pentru mai multe detalii despre `wget` şi pentru problemele ridicate de combinarea diverselor opţiuni consultaţi documentaţia programului.

Ce-i de făcut? Căutăm pe Google folosind tehnica exemplificată de secvenţa: `python -snake -monty`. Semnul minus pus în faţa unor cuvinte-cheie îi spune motorului de căutare să excludă de pe lista pe care o produce siturile despre şerpi sau „monty“.

Persoanele care sunt în căutarea şarpelui şi nu a limbajului de programare pot să pună `python -programming`.

O altă restrângere a căutărilor rezultă evident atunci când ne limită la situri care sunt numai într-o anumită limbă. Acest lucru se poate face explicit sau implicit: cuvintele folosite sunt, să zicem, în limba română.

În afara limbii, o altă posibilitate de a restrânge aria siturilor investigate o reprezintă folosirea unui motor de căutare specializat. Putem porni însă de la un motor de căutare general şi de aici să descoperim siturile cu motoare de căutare specializate. Google are chiar liste gata făcute cu asemenea situri specializate.

Dacă motorul de căutare permite acest lucru, putem desigur aplica şi tehnicile mai subtile ale expresiilor regulate.[102]

1.5 Corectura computerizată

Există multe metode şi programe de corectare, în special pentru limba engleză. Noi vom prezenta aici posibilitatea de a efectua o corectură ortografică în mediul integrat TEXnicCenter.

MySpell

TEXnicCenter foloseşte aceeaşi maşinărie pentru corectură ca şi OpenOffice. Numele tehnic al bibliotecii respective de funcţii este „MySpell“[103].

Pe situl <`www.openoffice.org`> puteţi găsi legăturile necesare pentru a descărca o sumedenie de dicţionare ortografice.

Soluţia pe care o descriem aici este una care poate fi dezvoltată independent de existenţa unui dicţionar gata făcut. În practică, am avut nevoie de ea pentru că mai demult nu exista pe situl OpenOffice un dicţionar ortografic pentru limba română.[104]

[102] Dacă n-aţi examinat încă anexa despre expresiile regulate, puteţi să o faceţi acum; vezi §1.3.

[103] Autorul MySpell este Kevin B. Hendricks. Algoritmii utilizaţi pentru a opera cu afixe sunt bazaţi pe cei ai lui Geoff Kuenning, autorul programului Ispell. Persoanele interesate de detalii pot citi cu folos sursele programului TEXnicCenter în partea lor care priveşte MySpell.

[104] La data de 22/10/2003 am găsit un dicţionar MySpell creat de Nicu Buculei pornind de la dicţionarul Ispell al lui Mihai Budiu <`http://www.cs.cmu.edu/~mihaib`>. Este un dicţionar destul de amplu, dar fără semnele diacritice din dicţionarul lui Budiu. RomanianOffice, care este un program de birou bazat pe

meşterirea unui dicţionar ortografic românesc

L-am improvizat în felul descris pe scurt în continuare. Primul pas constă în construirea unui fişier `ro_FI.dic` pe care trebuie să-l plasaţi în dosarul `Language` aflat pe calea folosită la instalarea TEXnicCenter. În fişierul acesta plasaţi o listă de cuvinte, corect ortografiate, puse fiecare la începutul unui rând şi sortate alfabetic. Primul rând al acestui fişier este rezervat pentru numărul de rânduri din fişier. Nu este greu să stabiliţi, cu ajutorul Vim, numărul de rânduri.

Mai trebuie să creaţi un fişier `ro_FI.aff` pe care-l puneţi tot în dosarul `Language` din dosarul unde este instalat TEXnicCenter. Raţiunea existenţei acestui fişier este foarte simplă. O voi explica cu ajutorul unui mic fragment dintr-un asemenea fişier:

```
SET ISO8859-2

PFX G Y 1
PFX G   0     meta         .
```

Primul rând specifică pentru corectorul ortografic codificarea folosită. Rândul al patrulea îi spune sistemului cum să ataşeze un prefix, foarte important pentru filosofie şi nu numai, prefixul *meta*. Se observă că aici a fost declarat un *steag*, cum se spune tehnic.[105] Steagul trebuie ataşat cuvintelor din fişierul de tip `dic` care pot fi prefixate cu *meta*. Iată două exemple evidente:

```
filosofie/G
limbaj/G
```

Sistemul va recunoaşte acum drept corecte şi cuvintele „metafilosofie“ şi „metalimbaj“.

Nu rămâne de făcut decât un mic reglaj în TEXnicCenter. În meniul Tools mergeţi la rubrica Options... şi apoi la panoul Spelling. Alegeţi `ro` în caseta Language. Observaţi cum apare automat `FI` („româna FIlosofică“) în caseta Dialect. Nu rămâne decât să bifaţi Check spelling while typing şi sistemul va colora cuvintele pe care nu le găseşte în listă sau nu le poate obţine din cuvintele din listă folosind metodele descrise în fişierul de tip `aff`.

OpenOffice, include un dicţionar cu diacritice, dar dicţionarul este proprietatea firmei INTERSOL. Chiar dacă versiunea 1 a produsului RomanianOffice este gratuită, nu puteţi folosi dicţionarul decât în cadrul acestui produs-program. În consecinţă, cred că este în continuare interesant de lucrat la un dicţionar plasat sub o licenţă de tip GPL.

[105] În limba engleză, termenul este *flag*.

În versiunile mai vechi ale TEXnicCenter nu se putea folosi decât Spelling... din meniul Tools. Începând cu versiunea 1 Beta 6.20 Beta, este posibil să dăm un clic pe butonul din dreapta al *mouse*-ului, pe cuvântul marcat ca incorect şi să folosim un meniu contextual.

Capitolul 2

Tehnoredactarea computerizată

Cuprins

2.1 LaTeX **64**
2.1.1 Utilizarea programului LaTeX: costuri şi beneficii 65
2.1.2 Instalarea TeX şi a programelor asociate 69
2.1.3 LaTeX într-o săptămână 75
2.2 BibTeX **114**
2.2.1 Primii paşi în lumea bazelor de date . . 114
2.2.2 Sistemul BibTeX 116
2.2.3 Stilurile bibliografice 119
2.3 Turnul Babel **121**
2.3.1 Literele româneşti 122
2.3.2 Vim şi adaptarea tastaturii 125
2.3.3 Limbile europene care folosesc alfabetul latin 132
2.3.4 Tehnica alegerii tipului de literă 134
2.3.5 Limba greacă veche 136
2.3.6 LaTeX şi unicod 141
2.3.7 Alte pachete cu simboluri în LaTeX . . . 143
2.4 Tabele şi formule **144**
2.4.1 Principiile de bază ale construirii tabelelor144
2.4.2 Câteva idei simple despre formule 148
2.4.3 Tehnicile avansate de scriere matematică 156
2.5 Indexarea electronică **156**

2.1 LaTeX

Donald Knuth

Programul de care ne-am folosit pentru a tehnoredacta cartea de faţă s-a născut din nevoi pur practice. Autorul său, Donald E. Knuth, era absorbit de munca la tratatul său despre *Arta programării calculatoarelor*[1]. Lucrul la tratat l-a început în 1962, pe vremea când cărţile se tipăreau în mod tradiţional. După 1970, în SUA, a început trecerea la producerea cărţilor cu ajutorul sistemelor computerizate. Donald Knuth a fost însă îngrozit de forma pe care o avea tiparul electronic la începuturile sale. El s-a decis să realizeze propriul său program de creat cărţi frumoase. Generos, el a plasat sistemul său zis TeX în domeniul public.[2]

Procesul prin care un text ajungea, în veacul trecut, să fie tipărit ar putea fi segmentat în următoarele faze: autorul crea un manuscris; manuscrisul era dactilografiat; textul dactilografiat era prelucrat într-o redacţie; după redactare, textul era tehnoredactat; după ce era tehnoredactat se dădea la cules, adică se crea pentru fiecare pagină o pagină din litere de plumb; textul cules era dat la corectură; varianta finală se tipărea folosind paginile de plumb.

Faza tehnoredactării însemna punerea pe manuscris a fel şi fel de semne pe care tipografii le citeau pentru a şti cum să meşterească paginile lor de plumb. Pe de o parte, trebuie să identificaţi existenţa unui limbaj al tehnoredactării. Pe de altă parte, trebuie să ţineţi cont că realizarea formelor în care erau turnate literele de plumb era o artă.[3]

Donald Knuth a fost extrem de dezamăgit de decăderea bruscă a artei tiparului sub impactul computerizării. El a rezistat tentaţiei de se crampona de vechile litere de plumb şi a folosit fantasticele sale calităţi de programator pentru a realiza un sistem de aşezare a textului în pagină probabil fără egal.

[1] Vezi Donald E. Knuth, *Arta programării calculatoarelor*, 3 volume (Bucureşti: Teora, 1999-2002). Aceasta este o carte celebră. Aşa cum reiese din textul reprodus pe ultima copertă a tuturor celor trei volume, Bill Gates a scris despre ea următoarele: „Dacă te crezi programator, ... citeşte *Arta programării calculatoarelor* de Knuth... Dacă poţi citi toată cartea, trimite-mi neapărat un C.V.“ Trebuie să ţineţi cont că aceste cuvinte vin nu doar din partea unuia dintre cei mai mari oameni de afaceri din toate timpurile, ci şi a cuiva care a publicat ca programator-cercetător (a se vedea B. Gates şi C. Papadimitriou, „Bounds for sorting by prefix reversals“, *Discrete Mathematics* 27: 47-57, 1979)

[2] O excelentă prezentare a bazelor sistemului TeX, în limba română, o găsiţi în cartea lui Moroşanu[8].

[3] Istoria acestei arte este povestită de S.Tóth[12].

2.1.1 Utilizarea programului LaTeX: costuri şi beneficii

Ce este LaTeX? Fără a intra în detalii, putem spune că LaTeX vă permite să tipăriţi (pe hârtie sau în format electronic), cu ajutorul în ultimă instanţă al TeX, eseurile dumneavoastră. Aceleaşi tehnici pot fi folosite pentru a tipări practic orice: articole, cărţi, afişe şi aşa mai departe. LaTeX a fost creat de către Leslie Lamport.[4]

Leslie Lamport

Cât este de greu de folosit sistemul LaTeX? Nu vă grăbiţi cu această întrebare! Mai bine faceţi un calcul cu privire la costuri şi beneficii.

De ce să nu folosim doar Vim pentru a da forma finală eseurilor? Vim este precum un stilou sau o maşină de scris. LaTeX este ca un tipograf.

Vestea bună ar fi că puteţi beneficia de posibilităţile unei edituri şi ale unei tipografii, folosind LaTeX şi programele asociate.

Şi vestea proastă? Este chiar proastă dacă aveţi nici un fel de cunoştinţe despre programarea calculatoarelor. LaTeX, ca şi TeX, este un limbaj de programare. Pentru a tehnoredacta eseul şi a-l tipări frumos trebuie să scrieţi un program.[5]

Care sunt acum rezultatele analizei cost-beneficii? Cred că sunt în favoarea LaTeX. Dacă scrieţi un eseu filosofic, nivelul pe care trebuie să-l atingenţi în domeniul programării în LaTeX este cât se poate de elementar. Practic, vă trebuie doar câteva zile ca să învăţaţi chiar dacă n-aţi programat niciodată. Costul acesta (care nu este nul)[6] este contrabalansat din plin de calitatea rezultatului obţinut.

programare elementară

La beneficiile unui text frumos tipărit se adaugă însă şi disciplina intelectuală pe care o impune utilizarea LaTeX. Acesta este un câştig mai puţin vizibil, dar extrem de consistent. TeX, LaTeX, BibTeX şi programele asociate sunt realizate de minţi strălucitoare din mediul academic. Dacă elaboraţi eseuri în mediul universitar, atunci spiritul acestor programe este cel de care aveţi nevoie.

disciplina intelectuală

[4] Prezentarea standard a sistemului este cea pe care o face chiar Leslie Lamport[4]. Cartea cuprinde un manual de utilizare şi o descriere sistematică a LaTeX. În limba română, există o prezentare a versiunii mai vechi a LaTeX (Pusztai şi Ardelean[10]). Blaga şi Pop[1] prezintă LaTeX2e.

[5] Moroşanu[8, p.10] subliniază foarte limpede caracterul de limbaj de programare al TeX.

[6] Cu un procesor de cuvinte puteţi scrie un text din prima clipă. Problemele vin abia când vreţi să faceţi lucruri mai complicate. În cazul LaTeX lucrurile stau exact invers: este mai greu la început, dar este mai uşor ulterior.

2.1.1.1 Avantajul de a face faţă complexităţii rescrierii textului

În aparenţă obiectivul este modest: aşezarea textului în pagină. În cazul Vim am fost preocupaţi de crearea de fişiere care conţin text, de căutarea de şiruri de semne într-un text şi de substituirea lor cu alte semne. Acum problema este de a aranja textul în pagină. Pentru a aşeza textul în pagină trebuie să ne angajăm însă deplin pe calea programării. Ca şi vechii tehnoredactori care-şi scriau pe manuscris mesajele către tipografi, noii tehnoredactori trebuie să trimită mesaje unui program care va aranja textul.

Mesajele către programele care vor prelucra textul pe computer sunt scrise într-un limbaj. Chiar dacă uneori pot să nu pară prea complicate, în spatele lor se află algoritmi, specificări ale paşilor care trebuie făcuţi pentru a rezolva o problemă.

capacităţile TEX

În ciuda obiectivului principal aparent modest, Knuth a pus bazele unui sistem care face o mulţime de lucruri. Între altele, sistemul poate gestiona structura unui articol sau a unei cărţi. Poate ţine evidenţa şi plasa automat în pagină notele de subsol sau notele marginale. Inserează imagini. Creează desene, scheme de care au nevoie autorii. Realizează indici. Ţine evidenţa listei bibliografice. Asigură posibilitatea de a face trimiteri interne sau la documente aflate pe Internet.

Ar fi greu de descris tot ce poate face o distribuţie TEX. Conceput iniţial pentru ştiinţa computerelor, sistemul a fost adoptat de către matematicieni. Din punctul de vedere al filosofiei, aceasta înseamnă că există în el o sumedenie de mijloace pentru a tehnoredacta texte de logică simbolică. Este însă cât se poate de util şi dacă vrem să scriem în greaca veche.

compilare

S-ar putea ca vestea proastă pentru mulţi să fie aceea că, pentru a lucra în TEX, trebuie să scrii o sursă care va fi supusă procesului de compilare. Programele obişnuite de birou tehnoredactează totul din mers, interpretând fiecare pas făcut de utilizator. Procesul de compilare este diferit. Sursa este prelucrată în întregimea ei. Erorile de programare pot duce la oprirea procesului de compilare ca atare. Procesul de realizare a produsului final este mai dificil, dar rezultatul obţinut are altă calitate.

Un sistem TEX, după cum se vede din cele scrise mai sus, este altceva decât o suită de programe pentru munca de birou.

rescrierea textului

Marea calitate a unui sistem TEX este că permite soluţionarea problemelor legate de rescrierea textului. Dacă n-ar trebui să rescriem un text, atunci sistemele uzuale ar fi destul de potrivite.

În realitate, munca intelectuală presupune aşternerea unor straturi succesive ale textului. Aici se vede, dacă treceţi peste dificultăţile începutului, superioritatea sistemului lui Donald Knuth.

Evident, frumuseţea aranjării finale în pagină nu este de neglijat nici ea. Dar, în practică, chiar dacă editorul m-a forţat să-i dau textul în formatul fişierelor unui editor de birou, tot a fost mai uşor să scriu în sistemul lui Knuth.[7]

2.1.1.2 Avantajele pentru cine vrea să aibă o perspectivă generală asupra ştiinţei computerelor

Sunt toate acestea doar simple unelte? Punctul de plecare al sistemului a fost de ordin practic, dar el a luat turnura necesară pentru a fi punctul de pornire şi al unor reflecţii cu caracter general. O anexă la un ghid de elaborare a eseurilor filosofice nu oferă cadrul necesar pentru a duce prea departe consideraţiile teoretice. Putem face doar scurte observaţii despre natura algoritmilor şi a proceselor algoritmice.

Începem cu o distincţie care aminteşte de distincţia dintre tip şi mostră. Dacă aranjăm în pagină şirul de semne `validitate`, câte litere „a“, „i“ sau „t“ trebuie să plasăm? Una sau două? Putem formula răspunsul în termenii distincţiei tip-mostră (sau probă de literă)[8]. În „validitate“ există şapte tipuri de literă, iar din tipurile „a“, „i“ şi „t“ avem câte două mostre sau probe de literă.

probă de literă vs. literă

În sistemul lui Knuth fiecare literă are un cod. La rândul lor, literele sunt grupate pe categorii. O probă, în sistemul lui Knuth, este desemnată printr-o pereche formată dintr-un cod şi o categorie.

Operarea cu o probă seamănă cu felul în care, în vechile tipografii, zeţarii[9] culegeau cu mâna literele din nişte cutii. Aveau cutii cu litere de unde luau ce le trebuia şi aşezau în pagină. Aceasta nu este altceva decât esenţa tiparului lui Gutemberg: literele mobile. Pe de

[7] Cel care m-a convins definitiv de superioritatea sistemului lui Knuth este cunoscutul logician Melvin Fitting. El mi-a explicat că, într-adevăr, îşi scrie scrisorile cu un editor de birou; cărţile şi articolele sale de logică sunt scrise însă cu un sistem TeX. Distincţia este limpede şi convingătoare: dacă scrisorile ar fi rodul unui proiect de cercetare, dac-ar avea o structură complicată, numeroase trimiteri etc., atunci ar trebui şi ele scrise cu un sistem gen TeX. O scrisoare are, de multe ori, o singură pagină şi este mult mai comod să aranjezi direct, vizual elementele ei în pagină decât să scrii un program. O carte are sute de pagini. Aici avantajul programării se face repede simţit.

[8] Tipografii colecţionează exemple din literele de care dispun şi alcătuiesc albume care se numesc „probare de litere“(v. *Dicţionarul limbii române*(Bucureşti: Editura Academiei, 1984), tom VIII, partea a 5-a, s.v.).

[9] În limba veche se numeau chiar „probari“.

altă parte, un „a“ poate avea categorii diferite. Aşa cum, în Vim, într-un mod este comanda de a plasa litera `a` în text, iar în alt mod o comandă de trecere la inserarea de text după poziţia curentă a cursorului.

Mai departe, putem rafina ideea de procesor. TEX are un număr limitat de comenzi de bază. Aceste comenzi pot fi folosite pentru a scrie alte comenzi mai complexe. Cu alte cuvinte, în inima sistemului conceput de Donald Knuth există un procesor virtual.

procesor

Când vorbesc despre procesoare oamenii se gândesc la procesoarele fizice. Putem face însă abstracţie de ele şi să ne concentrăm asupra a ceea ce face procesorul virtual.

În sistemul lui Donald Knuth, putem face o distincţie între două procese fundamentale: plasarea unor cutii virtuale pe pagină şi ceea ce este pus în cutiile respective. Sistemul nu plasează deci literele ca atare, ci cutiile în care sunt puse probe de litere. Putem astfel separa procesul de aranjare a cutiilor de procesul de desenare a literelor.

Aici ne vom concentra atenţia, în primul rând, asupra aranjării cutiilor şi alegerii unui conţinut potrivit pentru ele. La urma urmei, nu refacem desenul tipurilor de literă de fiecare dată când scriem un text. Folosim probele de litere existente şi variantele lor pentru a da un conţinut cutiilor. Limbajul care ne permite acest lucru ne interesează cu precădere.

Ca şi-n cazul procesoarelor reale, nu este tocmai uşor să programăm direct în limbajul procesorului virtual. În multe situaţii putem face abstracţie de ceea ce se întâmplă la nivelul procesorului. Ca şi atunci când folosim limbaje de nivel mai înalt, şi-n cazul sistemului lui Donald Knuth putem utiliza sisteme de comenzi mai complexe, care vor fi traduse automat în procesul compilării.

Compilarea înseamnă în sistemul lui Knuth generarea a unor fişiere care pot fi vizualizate şi tipărite. Pentru vizualizare şi tipărire vor fi folosite programe precum Ghostview sau Acrobat Reader.

limbaje de nivel înalt

Sistemul de comenzi complexe de nivel mai înalt pe care-l vom folosi este LATEX. Ca şi limbajele de nivel înalt, el facilitează enorm programarea. De asemenea, ne ajută să nu „reinventăm roata“. Dacă avem o problemă şi suntem începători, atunci este aproape sigur că altcineva a găsit o soluţie. LATEX ne permite să integrăm comod asemenea soluţii în propriul nostru program.

2.1.2 Instalarea TeX şi a programelor asociate

Distribuţia TeX la care ne referim constant în această carte se numeşte MikTeX. Puteţi găsi instrucţiunile de instalare la adresa <`http://www.miktex.org`>. Primul pas este obţinerea unei colecţii de arhive cu ajutorul unui program de descărcare-instalare. Presupun că veţi descărca mai întâi arhivele pe propriul computer. Cum volumul MikTeX este foarte mare, s-ar putea să fie mai rezonabil să obţineţi de la laboratorul Facultăţii un CD cu fişierele MikTeX.

Asiguraţi-vă apoi c-aţi instalat Ghostscript, Ghostview şi Adobe Acrobat.

Instalarea ca atare nu presupune nimic deosebit. Este suficient să urmaţi instrucţiunile. Recomandarea mea ar fi să nu economisiţi spaţiul de pe disc şi să instalaţi sistemul TeX în forma sa completă. Puneţi, de asemenea, sistemul TeX într-un dosar special şi pe o cale care nu conţine nume de dosare cu spaţii. Aţi putea folosi, de pildă, ceva de genul `D:\ed\tex`. Puneţi în dosarul `tex` atât dosarul `texmf`, dosarul principal al distribuţiei MikTeX, cât şi `localtexmf`. Tot în `tex` aş sugera să fie pus tot ce este legat de TeX.

Instalarea totală a distribuţiei MikTeX poate să ia destul de mult timp.[10] În momentul când scriu aceste rânduri, ultima versiune a MikTeX, versiunea 2.4, instalează *44850* de fişiere (929 de pachete). Pe computerul cu care este scrisă această carte dosarul `tex`, cu toate programele asociate, ocupă 654MB şi conţine 46758 de fişiere în 4566 dosare.

Dacă n-aveţi suficient spaţiu pe discul dur, instalaţi doar o versiune redusă a distribuţiei. Aveţi o opţiune în acest sens chiar în programul de instalare. Mai puteţi elimina, de asemenea, din documentaţia sau pachetele pe care nu le folosiţi foarte des. Folosiţi pentru aceasta programul MikTeX Package Manager.

Configurarea sistemului MikTeX nu este deosebit de complicată. Asiguraţi-vă de existenţa în AUTOEXEC.BAT, după `SET PATH`, a unei căi de genul `D:\ED\TEX\TEXMF\MIKTEX\BIN`. Aceasta este calea din Windows unde sunt puse executabilele MikTeX. Dacă ea nu există, adăugaţi calea adecvată sistemului dumneavoastră. Nu uitaţi că trebuie să separaţi căile prin punct şi virgulă (vedeţi modelul din subsecţinea 1.1.4.4).

[10] Se consumă timp şi cu verificarea integrităţii pachetelor. Acest proces este însă absolut necesar. S-ar putea ca unele fişiere să se fi descarcat parţial de pe Internet. De asemenea, procesul acesta de verificare vă fereşte de nedorite coruperi ale fişierelor pe parcursul circuitului lor pe Internet sau al scrierii pe CD.

N-ar fi rău să reorganizaţi şi Start Menu în aşa fel încât să nu fie toate scurtăturile puse în Programs. În orice caz, găsiţi scurtătura care se numeşte MikTeX Options şi care trimite către `mo.exe`, unul dintre executabilele MikTEX. Executaţi un clic pe Languages şi bifaţi limbile de care aveţi nevoie. În orice caz, bifaţi romanian. Daţi apoi clic pe General şi, pentru siguranţă, un clic pe Refresh Now. Această împrospătare a bazei de date este absolut necesară când adăugaţi noi pachete cu fişiere.

Sistemul este efectiv uriaş şi integrează o contribuţii realizate de către diverşi autori. Porniţi MikTeX Package Manager şi veţi avea o listă a pachetelor instalate, cu unele scurte informaţii despre ceea ce face fiecare.[11]

După configurare, Win98 trebuie repornit. Dacă totul a mers bine, ar trebui ca sistemul MikTEX să fie funcţional.

2.1.2.1 Sub ce sisteme de operare funcţionează TEX?

Întrebarea din titlu este foarte importantă. Dacă intraţi în sala calculatoarelor de la Facultatea de Filosofie, observaţi imediat computerele Sun şi PC-urile care funcţionează sub sistemul de operare Linux. Sistemele de operare sunt, în acest caz, de tip Unix. Alături de ele veţi găsi însă cel puţin un PC care foloseşte Windows.

Multe utilizatoare şi utilizatori de sisteme de calcul au acasă sau la serviciu WindowsXP, nu Windows98. Merge LATEX sub toate aceste sisteme de operare? Răspunsul este categoric da.

Instalarea sub WindowsXP este asemănătoare cu aceea de sub Windows98. Atenţie doar la setarea specifică a variabilei de mediu pentru calea pe care se află executabilele MikTEX.[12] Din experienţa noastră concretă am putea spune că MikTEX se instalează mai lesne sub WindowsXP. Într-unul dintre cazuri instalarea a fost foarte dificilă, pe acelaşi computer, sub Windows98 şi lină sub WindowsXP. Sub Windows98, computerul s-a blocat de câteva ori. WindowsXP gospodăreşte evident mai bine resursele calculatorului şi asigură o funcţionare mai stabilă.

Dar Linux? LATEX este la el acasă sub Linux. Orice distribuţie Linux importantă include TEX şi programele asociate. Numele distribuţiei respective, sub Linux, este TeTEX.

N-am o experienţă directă de utilizator de LATEX sub alte sisteme

[11] Acest program există începând cu versiunea 2.3 a MikTEX, dar lipseşte din versiunile anterioare.

[12] Vezi aici explicaţiile de la pagina 20, rândul 18.

Unix. Din câte ştiu, lucrurile ar trebui să decurgă fără probleme. Aceeaşi este situaţia şi-n cazul Mac.

2.1.2.2 Ce se întâmplă dacă n-aveţi computer sau ai unul foarte vechi?

Aş recomanda însă LaTeX studentelor şi studenţilor care nu au acasă un computer sau care au un model foarte vechi.[13] Ce poţi face cu un computer foarte vechi? Poţi scrie fişiere de tip text! Poţi folosi o versiune mai veche a Vim, eventual fără interfaţa grafică.

Pentru sursele LaTeX nu este nevoie decât de fişiere de tip text. O carte întreagă încape fără probleme pe o singură dischetă. Sursele respective pot fi apoi compilate şi corectate sub Linux, într-un laborator care dispune de computere performante.

Puteţi folosi pentru a crea surse LaTeX chiar şi un calculator care n-are disc dur! Un PC foarte vechi, de la începutul anilor '80 ai secolului trecut, cu doar două dischete poate fi folosit cu mult succes. Un calculator de tip Spectrum, dar care are şi sistemul de operare CP/M (cum este calculatorul românesc HC2000), poate servi şi el la producerea fişierelor text care alcătuiesc sursa LaTeX.

Nu uitaţi că LaTeX sau programele asociate pot produce şi desene pornind de la fişiere de tip text!

Achiziţionarea de sisteme scumpe doar pentru a scrie eseuri pentru examene, lucrări de diplomă, articole, teze de doctorat, cărţi este efectiv o eroare. Investiţia trebuie făcută în direcţia învăţării programării, nu a utilizării de computere foarte scumpe.

Singurul lucru de care este nevoie, pe lângă computerul ieftin care produce fişierele de tip text, este accesul la un laborator cu computere pe care este instalat LaTeX.

La limită, puteţi sta şi scrie chiar în laborator (sau la un Internet café). Rezultatul depinde de mintea celei sau celui care scrie, nu de preţul plătit pe calculator.[14]

[13] Astăzi, un model vechi de computer pe care să meargă bătrânul MS-DOS sau o versiune veche de Windows poate fi achiziţionat la un preţ care oscilează între salariul minim şi cel mediu din România. Un sistem foarte vechi are un preţ chiar mai mic.

[14] În practică, se întâmplă, uneori, să primeşti eseuri scrise în limba română, dar fără diacritice! Ce aţi zice dacă vi se arată, prin contrast, un text cu dicritice şi citate frumos realizate, în greaca veche?

2.1.2.3 Mediul integrat de dezvoltare TEXnicCenter

În configuraţia pe care o aveţi în acest moment trebuie să folosiţi comenzile în linie într-o fereastră MS-DOS pentru a compila sursele LATEX. Aş sugera că este absolut utilă instalarea unui mediu integrat de dezvoltare de programe LATEX.[15]

Există şi un script Vim, creat de Fritz Mehner, care creează meniuri, inclusiv pentru compilare şi vizualizare.[16] Scriptul trebuie configurat pentru a lucra adecvat.[17] S-ar putea să doriţi să folosiţi acest script sub Linux pentru a avea acces rapid la compilare şi vizualizare.

Am arătat deja cum putem găsi pe Internet locul de unde poate fi descărcat TEXnicCenter.[18] Recomandarea ar fi să instalaţi acest program într-un dosar plasat în dosarul în care aveţi şi MikTEX.

Instalarea ca atare n-ar trebui să vă creeze probleme, dacă urmaţi instrucţiunile. Versiunea care a fost folosită pentru a compila sursele cărţii de faţă are însă o particularitate care s-ar putea să vă atragă atenţia; se numeşte „1 Beta 6.01“. Ce semnificaţie are „Beta“? Programele sunt texte şi lor li se aplică din plin principiul rescrierii. Ele sunt rescrise până se ajunge la o versiune matură. O versiune beta este încă într-un stadiu în care mai sunt erori. Este însă suficient de bună pentru a fi utilizată, iar autorii aşteaptă mesajele utilizatorilor cu privire la eventuale deficienţe.

În ciuda acelui „beta“ din versiunea folosită de către noi, mediul integrat funcţionează destul de bine. Este totuşi recomandat să-l folosiţi exclusiv pentru a compila sursele.[19] Pentru scrierea surselor ar fi mult mai sigur să folosiţi Vim.

Configurarea TEXnicCenter este ceva mai problematică decât in-

[15] Este vorba aici mai ales de utilizatorii sistemului de operare Windows. Linux transformă computerul într-o puternică staţie de lucru. Cine este capabilă sau capabil să lucreze în Linux se poate descurca folosind `Makefile`-uri. De asemenea, comanda în linie din Linux este incomparabilă ca putere cu ceea ce oferă sistemul Windows ca atare.

[16] Vezi <http://lug.mfh-iserlohn.de/vim/vim-latex/vim-latex.html> pentru explicaţii, exemple de utilizare şi descărcarea scriptului lui Mehner.

[17] N-am testat scriptul sub Linux, dar acesta pare conceput sub un sistem Unix.

[18] A se vedea aici pagina 57, rândul 30.

[19] Modul acesta de lucru este, de altfel, imperios necesar dacă n-aveţi la dispoziţie în mod curent un calculator pe care este instalat LATEX. Creaţi sursele programelor separat şi apoi executaţi ciclul compilare-vizualizare-corectare până obţineţi rezultatul dorit.

stalarea. Când porneşte pentru prima oară TEXnicCenter lansează automat un Wizard[20] care configurează mediul integrat.

Dacă mergeţi pe ruta Build→ Define Output Profile... şi daţi un clic pe butonul Wizard, puteţi porni oricând doriţi programul de configurare.

Dacă pe computer este instalat editorul Acrobat, editorul de fişiere `pdf` al firmei Adobe, atunci s-ar putea să obţineţi automat o configurare în care fişierele de tip `ps` (fişierele PostScript) sunt deschise de către programul `distiller`, care converteşte un fişier de tip `ps` într-unul `pdf`. Dacă doriţi doar să vizualizaţi fişierul PostScript, trebuie să modificaţi manual configuraţia TEXnicCenter sau să faceţi în aşa fel încât Ghostview să fie programul care deschide automat fişierele `pdf` sub Windows.

Ce trebuie să ştiţi pentru a configura mediul integrat? În primul rând, trebuie să ştiţi unde sunt plasate executabilele MikTEX[21] Căile către Ghostscript şi Acrobat Reader ar trebui să fie identificate automat. Evident, este bine însă să le cunoaşteţi dinainte.

Singurele probleme serioase cu configurarea automată, sub Windows98, le-am avut în cazul Acrobat Reader. Dacă vedeţi, pe parcurs, că Acrobat Reader nu porneşte când îl chemaţi din mediul integrat, mergeţi pe ruta Build→Define Output Profile..., selectaţi LaTeX=>PDF şi daţi clic pe Viewer. Veţi vedea acolo trei comenzi. Dacă este o comandă DDE, la primele două puneţi [FileOpen ("%bm.pdf")]. Acum ar trebui să puteţi deschide Reader-ul. Alternativ, alegeţi comanda în linie şi puneţi `"%bm.pdf"` la primele două comenzi. Problema care rămâne este legată de rescrierea fişierului `pdf` deschis în Reader. Reader-ul nu permite modificarea fişierului pe care l-a deschis. Acesta ar trebui închis. Dacă n-o face mediul integrat, închideţi manual fişierul direct în Reader.

Sub WindowsXP, TEXnicCenter a configurat în mod automat vizualizarea cu Acrobat Reader în modul descris mai sus. Posibilitatea de a închide automat fişierul `pdf` nu există. Dacă nu-l închideţi, veţi primi un mesaj de eroare de la compilator.

Versiunea mai nouă, 1 Beta 6.20, integrează mai bine Reader-ul. Ea are şi un corector ortografic mai funcţional, precum şi o serie de alte îmbunătăţiri.

[20] Cuvântul acesta înseamnă în engleză *vrăjitor*. n-are însă sens să-l traducem. Aici este folosit ca o simplă etichetă.

[21] Pentru conceptul de „executabile MikTEX“ se vedea aici explicaţia de la pagina 69.

2.1.2.3.1 Integrarea Vim în TEXnicCenter TEXnicCenter are propriul său editor de texte. Are, de asemenea, meniuri care permit introducerea direct în mediul integrat a comenzilor din limbajul LATEX.

Este posibil însă să integraţi editorul Vim în TEXnicCenter. Mergeţi la meniul Tools→ Customize şi daţi un clic pe Tools, iar apoi un clic pe pictograma New. Daţi un clic pe butonul din dreptul casetei Command şi procedaţi ca şi cum aţi deschide fişierul `gvim.exe` (principalul executabil din dosarul unde este instalat Vim). La argumentele comenzii puneţi `+%l %pc`.[22] La directorul iniţial puneţi `%dc`. Aţi indicat astfel că vreţi să deschideţi documentul curent din TEXnicCenter în Vim.

deschideţi Vim la rândul curent

Vim şi TEXnicCenter colaborează bine şi, dacă aţi modificat textul în Vim, TEXnicCenter vă va întreba dacă să modifice la rândul său textul. Invers, dacă aţi modificat textul în TEXnicCenter, la revenirea în Vim daţi clic pe Load File în caseta de dialog care va apărea automat. Preluaţi astfel automat schimbările. Altfel, le pierdeţi! Veţi primi, ce-i drept, nişte mesaje din partea Vim.

Pentru a verifica dacă totul merge fără probleme este imperios necesar să exersaţi pe fişiere test. Nu treceţi direct la scrierea eseului pe care trebuie să-l predaţi mâine. S-ar putea să aveţi surprize neplăcute.

Multe cititoare şi cititori se vor fi întrebând însă ce rost are să folosim Vim.Mediul integrat are o mulţime de pictograme care ne permit să scriem comenzile LATEX fără efort. Limbajul LATEX este extrem de subtil. Ar fi imposibil pentru cineva să creeze exact comenzile de care aveţi nevoie. Recomandarea noastră este să creaţi propriile dumneavoastră meniuri. Alternativ, dacă n-aveţi interfaţă grafică sau nu vă plac meniurile, folosiţi scripturi Vim. Puteţi crea uşor scripturi Vim folosind doar al treilea bloc de cod din exemplele noastre cu meniuri Vim.[23]

creaţi meniuri proprii

La urmă, dar nu în cele din urmă, cred că trebuie accentuată recomandarea de a folosi separat Vim de mediul integrat cât timp construim sursele. Cu alte cuvinte, editorul Vim (sau unul similar)

[22] Litera `l` vine de la *line*; nu o confundaţi cu cifra 1. Studiaţi opţiunile la invocarea în linia de comandă a lui Vim cu ajutorul comenzii `gvim.exe -help`.

[23] Cum porniţi un script Vim? Tastaţi, în modul normal, comanda `:source` sau, mai scurt, `:so` şi apoi calea unde se găseşte scriptul, respectiv doar numele scriptului vim, când acesta se află în dosarul curent. De asemenea, istoria comenzilor în linie este de mare ajutor. Căutaţi şi un fişier `_viminfo`. S-ar putea să fie în `home`, dacă aveţi aşa ceva. Acolo găsiţi istoria comenzilor. Ce vi se pare mai reuşit puteţi păstra în alt fişier în vederea construirii de script-uri Vim.

trebuie folosit pentru a introduce textul ca atare. Folosim mediul integrat doar pentru a compila sursele şi pentru a le corecta. În acel moment s-ar putea să avem nevoie de funcţionarea integrată şi a editorului Vim.[24]

2.1.2.3.1.1 Vim şi programul de vizualizare a fişierelor `dvi` După compilarea sursei LaTeX rezultă un fişier `dvi` care este vizualizat cu ajutorul programului Yap din distribuţia MikTeX. La instalarea TeXnicCenter, Yap este setat în aşa fel încât un dublu clic ne readuce în editorul mediului integrat.

Mergeţi în Yap pe ruta View→ Options.... În panoul Inverse Search modificaţi comanda după modelul următor:

```
D:\use\Vim\vim62\gvim.EXE +%l %f
```

Fiţi atente şi atenţi să nu repuneţi în drepturi TeXnicCenter: nu selectaţi nimic în caseta pentru programe! Secretul comenzii către Vim este `+%l` care-i spune să deschidă fişierul-sursă la linia `l`. Puneţi evident calea potrivită pentru felul în care este instalat Vim.

S-ar putea să fie necesar să verificaţi şi registrul Windows, dacă nu merge totul cum trebuie. Cu toate că pare mai dificil de operat, schimbarea aceasta este foarte utilă când vreţi să operaţi corecturi mai complicate în fişier.

2.1.3 LaTeX într-o săptămână

LaTeX este un limbaj de programare. Trebuie învăţat ca orice limbaj de programare: făcând exerciţii. Exerciţiile propuse de către noi aici constau în construirea unor meniuri Vim foarte simple.[25]

De data aceasta meniurile Vim nu sunt un scop în sine, ci un mijloc de a tasta mai rapid componentele diverselor construcţii posibile în LaTeX. Atenţia trebuie să fie concentrată asupra comenzilor LaTeX ca atare, nu asupra modului în care sunt definite meniurile în limbajul Vim.

Independent de meniuri, comenzile LaTeX pot fi introduse cu ajutorul oricărui editor de texte prin simpla tastare a textului comenzilor.

[24] Recomandarea aceasta este făcută şi pentru că mediul integrat recomandat aici este într-o versiune beta. N-am avut accidente majore cu editorul său intern. Impresia mea este totuşi că nu manevrează bine fişiere mari. Vim este mult mai sigur şi mai flexibil.

[25] Vezi aici §1.2.2.1.1.

2.1.3.1 Prima zi

schema comenzilor LaTeX

Prima idee care ar trebui învăţată este cea de comandă LaTeX ca atare. Scheletul unei comenzi LaTeX este următorul: `\{}`. Între bara oblică inversă şi prima acoladă se pune numele comenzii. Între acolade se pune argumentul comenzii. Argumentul nu este altceva decât materialul pe care comanda îl prelucrează în conformitate cu algoritmul aflat în spatele ei.

Unde sunt plasate comenzile? În textul eseului ca atare! Ele sunt aidoma însemnărilor pe care le face tehnoredactorul pe manuscris. Aceste însemnări le foloseşte apoi tipograful pentru a tipări textul.

semne rezervate

Cum de nu se confundă comenzile ca atare cu elemente similare ale textului? Există o serie de semne rezervate exclusiv pentru comenzi. Aţi văzut deja trei dintre aceste semne mai sus. Lista completă a semnelor rezervate este următoarea: $ & # % _ { } ~ ^ şi, la urmă, dar nu ultima în ordinea importanţei, bara oblică \. În total 10 semne. Dacă vrem să tipărim aceste semne ca atare, atunci trebuie să dăm nişte comenzi prin care să cerem acest lucru.

Cineva s-ar putea să fie nedumerit. Ce face LaTeX dacă întâlneşte semnul, să zicem, `s` (o literă uzuală din alfabet)? Într-un fel şi semnul aceasta este o parte a unei comenzi. LaTeX este ca un tipograf. Aşa cum tipograful trebuie să ştie la ce cutie cu litere din plumb să se ducă, tot aşa LaTeX trebuie să ştie ce literă-ca-tip şi ce tip de literă trebuie să tipărească pe ecran sau pe foaia de hârtie. Scepticul s-ar putea să clatine din cap şi să mormăie în sinea sa că este prea complicat. Dar şi-n cazul unui editor uzual de birou trebuie să alegeţi tipul de literă!

meniu Vim pentru scheletul unui program LaTeX

Exemplul practic s-ar putea să vă convingă însă că este mai simplu decât credeţi. Creaţi un fişier de tip `vim` şi introduceţi următoarele linii de cod Vim pentru a realiza un meniu:

```
1 :imenu ltx1.antet \documentclass[a4paper,11pt]{}<Left>
2 :imenu ltx1.corp \begin{document}<CR><CR>\end{document}<Up><Home>
```

Cele două puncte ne arată că este vorba despre comenzi în linie pentru Vim. Din `imenu` reiese destul de uşor că este vorba despre un meniu activ în modul insert. Numele meniului care apare pe bara principală cu meniuri este `LaTeX`. Punctul pus după numele meniului este urmat de numele elementului pe care vrem să-l introducem în meniu.

Dacă nu vreţi sau nu puteţi lucra în mod grafic cu Vim, experimentaţi următorul tip de comandă într-un script Vim.

```
normal i\documentclass[a4paper,11pt]{}
```

Ideea este aceeaşi ca şi-n primul element al meniului de mai sus. De data aceasta însă, cuvântul-cheie `normal` îi spune lui Vim să treacă din modul comandă în linie în modul comandă normal. Urmează apoi comanda `i`, pentru a insera text, şi textul ca atare.

Puteţi da comanda de mai sus direct în modul comandă în linie. Aceasta ne arată ce editor formidabil este Vim. Ideile pe care le folosim în construcţia interfeţei grafice pot fi utilizate şi-n absenţa acesteia, într-un mod cât se poate de eficient.

Ce sens au elementele de meniu? În primul element de meniu, ideea este de a scrie o comandă care să spună despre ce tip de document este vorba. Când dăm clic pe primul element de meniu, Vim scrie în fişier textul cerut şi mută cursorul între acolade. Acolo trebuie să scriem despre ce tip de document este vorba. Scrieţi, pentru început, `article`. Orice eseu de dimensiuni mai mici este de tipul `article`. De asemenea, pentru a nu vă complica viaţa de la bun început, consideraţi că orice eseu mai lung (lucrare de diplomă, disertaţie, teză de doctorat sau carte) este de tipul `book`. În realitate, LaTeX este mult mai sofisticat şi puteţi chiar defini propriile dumneavoastră tipuri de documente.[26]

Ce sunt parantezele drepte? După cum se vede mai sus, nu fac parte dintre semnele rezervate. În comenzile LaTeX se pun între paranteze drepte opţiuni. Evident, LaTeX are nişte opţiuni standard şi aţi putea omite cu totul acel element. Noi am vrut să arătăm care este conţinutul mai important al acestor opţiuni: formatul foii de hârtie şi mărimea literelor. Unităţile de măsură ale literelor se numesc puncte[27]. Ar fi inutil să facem teoria acestor unităţi de măsură aici. Experimentând veţi vedea ce rezultă pe ecran şi pe hârtie.

Opţiunea cea mai importantă pe care s-ar putea să vreţi să o controlaţi este cea a tipăririi pe o singură faţă a colii de hârtie (`oneside`)[28] sau pe ambele feţe (`twoside`).

Comanda care specifică tipul de document este elementul obli-

[26] Definirea de tipuri de documente nu este însă o operaţie pe care o pot face începătorii sau utilizatorii obişnuiţi LaTeX. De altfel, recomandarea ar fi să respectaţi disciplina impusă de LaTeX chiar şi atunci când ştiţi destul de multe elemente ale limbajului.

[27] Scrieţi doar `pt` după numărul care specifică dimensiunea literelor.

[28] În mod tradiţional, de exemplu, lucrările de licenţă sunt tipărite pe o singură parte a foii de hârtie. Dacă le construiţi folosind tipul de document `book` va trebui să specificaţi explicit opţiunea `oneside`. La articole lucrurile stau exact invers.

gatoriu al *antetului* unui program LaTeX. Programul trebuie să aibă însă şi ceea ce se numeşte un *corp*.

În al doilea element de meniu, i se spune, de fapt, editorului Vim cum să creeze corpul unui program LaTeX. Observaţi că, de data aceasta, este vorba despre o pereche de comenzi. Trataţi-le ca pe nişte paranteze. Dacă una lipseşte, atunci programul conţine o eroare.

Acţiunea perechii de comenzi se exercită numai asupra a ceea ce se află între ele. Lucrul acesta dă mari bătăi de cap novicilor. Creaţi un fişier cu extensia `tex`. Să zicem că numele acestui fişier este `prim.tex`. Deschideţi `prim.tex` cu Vim. Dacă totul este în regulă, puteţi rula scriptul Vim creat şi aveţi un meniu cu două elemente în el. Folosiţi-le pentru a scrie un prim program LaTeX, ceva în genul a ceea ce vedeţi mai jos.

program LaTeX minimal

```
\documentclass{article}
\begin{document}
Salut cititoarele/cititorii!
\end{document}
```

Salut cititoarele/cititorii!

În exemplul de mai sus vedeţi în partea stângă un program minimal şi-n partea dreaptă rezultatul obţinut după compilare.

Ce este **compilarea**? Un program conţine comenzile pe care vrem să le execute computerul. În cazul compilării, programul este prelucrat, ca să spunem aşa, în întregimea sa. Nu obţinem rezultatul dorit pe bucăţi, comandă cu comandă.

Adevărul este că LaTeX este suficient de flexibil pentru a lucra şi bucată cu bucată. Scrieţi în `prim.tex` doar antetul. Într-o fereastră MS-DOS, cu promptul chiar în dosarul unde este `prim.tex`, putem duce un dialog cu TeX de genul celui care urmează:

```
E:\test\zi1>latex prim.tex
This is e-TeX, Version 3.141592-2.1 (MiKTeX 2.4)
entering extended mode
(prim.tex
LaTeX2e <2001/06/01>
Babel <v3.7m> and hyphenation patterns for english loaded.
)
*\begin{document}
(D:\ed\ltx\texmf\tex\latex\base\article.cls
Document Class: article 2001/04/21 v1.4e Standard LaTeX
document class
(D:\ed\ltx\texmf\tex\latex\base\size11.clo)) (prim.aux)
*Salut lume!
*\end{document}
[1] (prim.aux)
```

```
Output written on prim.dvi (1 page, 228 bytes).
Transcript written on prim.log.
```

Promptul > este cel al sistemului de operare, care aşteaptă o comandă în linie. Steluţa este promptul sistemului TEX, care aşteaptă comenzi. Mesajele indică faptul că totul s-a terminat cu bine.

Pentru a compila fişierul `prim.tex` vă trebuie un document complet, după modelul indicat mai sus. Când daţi acum comanda în linie `latex prim` nu mai trebuie să duceţi nici un dialog cu TEX. Puteţi vedea rezultatul dând un dublu clic pe `prim.dvi`.

unde scriem textul eseului?

Atenţie, nu scrieţi textul care vreţi să apară pe ecran în afara corpului programului! În antet, ar fi o eroare. După `\end{}` n-ar avea nici un efect. Aceasta din urmă este eroarea tipică a novicilor. Este semnul că n-aţi înţeles rolul perechii de comenzi. Nu este nici o problemă! Exersând vă daţi seama care este funcţia corpului programului.

Folosiţi mediul integrat pentru compilare. Daţi dublu clic pe fişierul `prim.tex` şi acesta va fi deschis în TEXnicCenter. Căutaţi pictograma Build current file (Ctrl+F7). Daţi un clic. În fereastra de jos a mediului integrat veţi vedea o serie de mesaje. Dacă sunt 0 Error(s), pentru o primă încercare, totul este în ordine. Pentru a vedea rezultatul, daţi un clic pe pictograma View output.

importanţa `dvi`

Este recomandabil să menţineţi caseta Output profile pe opţiunea LaTeX=>DVI cât timp lucraţi la primele versiuni ale eseului dumneavoastră. Acestea sunt fazele în care, înainte de era computerelor, aţi fi scris cu mâna. Abia când pregătiţi versiunea pentru printer sau pentru ecran are sens să compilaţi un fişier `PostScript` sau `pdf`. Deocamdată verificaţi doar dacă totul este în ordine pentru toate cele trei tipuri de rezultat (`dvi`, `ps` şi `pdf`).

Vizualizatorul fişierelor `dvi` este setat în aşa fel la instalarea mediului integrat încât puteţi da dublu clic într-un punct al textului şi veţi reveni în editorul de texte din mediul integrat în punctul corespunzător din sursă.[29] În acest fel puteţi parcurge lesne ciclul compilare–vizualizare–corectare până obţineţi rezultatul dorit.

Citiţi deci sursa pentru a vedea dacă n-aveţi erori din perspectiva limbajului LaTeX. Citiţi rezultatul în programul de vizualizare pentru a vedea dacă el corespunde cu intenţiile dumneavoastră. Corectaţi sursa şi recompilaţi pentru a vedea noul rezultat.

[29] Pentru deschiderea Vim din Yap v. §2.1.2.3.1.1

2.1.3.1.1 Proiectul LaTeX Dacă elaboraţi un document de dimensiuni mai mari şi puneţi tot ce este legat de documentul respectiv într-un singur fişier, acesta devine extrem de greu de inspectat. Principiul sursei deschise ar putea funcţiona şi la nivelul unui fişier foarte amplu, dar caracterul deschis al sursei nu ne-ar fi de mare folos. În plus, câştigăm în planul structurării logice a documentului dacă-l secţionăm.

Ce facem însă cu sumedenia de fişiere care ar putea rezulta? Le punem într-un singur dosar. În dosarul respectiv nu punem nimic care să nu fie legat de proiectul la care lucrăm.

TEXnicCenter este de mare ajutor când este vorba despre lucrul cu proiecte. Deschideţi TEXnicCenter şi mergeţi la meniul File. De aici deschideţi fereastra de dialog New Project.... Alegeţi o cale (Project path) în care va sta dosarul cu proiectul dumneavoastră. Daţi un nume proiectului (de pildă, „eseu“) şi TEXnicCenter va crea automat un dosar cu numele proiectului (în cazul nostru, dosarul `eseu`). În versiunea 1 Beta 6.01 a TEXnicCenter nu este posibil decât să creaţi un proiect vid (`empty project`).

Deocamdată nu ştiţi nimic despre BibTeX şi MakeIndex, aşa că puteţi lăsa nebifate rubricile lor.

Dacă exploraţi dosarul `eseu`, veţi vedea că TEXnicCenter a creat trei fişiere: `eseu.tcp`, `eseu.tps` şi `eseu.tex`. Primele două fişiere le modifică doar TEXnicCenter. Pot fi însă lesne vizualizate cu Vim. Să vedem ce se găseşte în `eseu.tcp`:

```
1 [FormatInfo]
2 Type=TeXnicCenterProjectInformation
3 Version=2
4
5 [ProjectInfo]
6 MainFile=eseu.tex
7 UseBibTeX=0
8 UseMakeIndex=0
```

Pe rândurile 7 şi 8 se găsesc două opţiuni evidente. Dacă la crearea proiectului n-aţi bifat Uses BibTeX şi Uses MakeIndex, valoarea acestor opţiuni este 0. Când veţi vrea să folosiţi programele respective, nu este dificil să puneţi 1 în loc de 0. Nu faceţi însă alte modificări.

Pe rândul 6 din fişierul `eseu.tcp` ni se spune că `eseu.tex` este fişierul principal al proiectului. Aceasta este o informaţie importantă. Acest fişier poate fi modificat şi dinafara mediului integrat, cu ajutorul editorului Vim. Fie în mediul integrat, fie cu ajutorul Vim, puneţi în fişierul principal următoarele rânduri de program LaTeX:

```
1 \documentclass[a4paper,12pt]{article}
2 \author{Prenume Nume}
3 \title{Primul meu eseu}
4 \begin{document}
5 \maketitle
6 \tableofcontents
7 \include{text}
8 \end{document}
```

Atenţie în special la rândul 7 din fişierul principal. Acest rând cuprinde o comandă care cere includerea unui fişier de tip `tex`, numit `text.tex` în proiect. Nu puneţi extensia fişierului în comanda `\include`. Fişierul trebuie creat de către dumneavoastră. O puteţi face atât în mediul integrat, cât şi-n afara acestuia. Trebuie însă neapărat să puneţi acest fişier în dosarul proiectului.

Ce cuprinde fişierul `text.tex`? Deocamdată o singură linie de text pur. Este suficient atât pentru prima zi. În perspectivă, în acest fişier poate sta un întreg eseu.

Atenţie! Nu puneţi antet sau corp de document în fişierul acesta sau în altă parte decât în fişierul principal.

Dacă cercetaţi fişierul principal, vedeţi că-n antet (în preambulul programului) se află o comandă care indică sistemului numele autoarei sau autorului. De asemenea, o altă comandă arată care este titlul. Aceste comenzi sunt necesare desigur pentru comanda `\maketitle` din corpul documentului.

De ce lipsesc acoladele după `\maketitle` şi `\tableofcontents`? Puteţi să le puneţi. Nu se întâmplă nimic rău. Aceste comenzi îşi culeg însă singure materialul. Ne vom mai întâlni, de altfel, cu comenzi cu argumentul vid.

Versiunea 1 Beta 6.20 a mediului integrat are în meniul Project rubrica Create with active file as main file. Cu ajutorul ei procesul descris mai sus se simplifică. Puteţi crea direct fişierul principal şi, pe baza lui, întregul proiect.

Cum compilaţi proiectul? Căutaţi pe bara cu instrumente pictograma `Build Output`(F7). Daţi clic şi compilaţi. **F7**

Are rost să construim însă un proiect pentru un simplu articol, pentru un eseu de dimensiuni mici? Are foarte mult sens să facem acest lucru. Este un pas înainte în separarea procesului de elaborare a textului ca atare şi a procesului de compilare. Puteţi scrie fişierul `text.ttt` pe cu totul alt computer decât cel pe care compilaţi şi doar să-l integraţi ca pe un modul în proiect. De asemenea, separaţi comenzile privitoare structura de ansamblu a documentului de tot ce este legat de conţinutul diferitelor părţi ale proiectului.

În cazul unei cărţi, este recomandabil ca fiecare capitol să fie pus într-un fişier aparte. Evident, în fişierul principal al proiectului, trebuie să schimbaţi tipul documentului din `article` în `book`.

S-ar putea ca unii să fie îngroziţi de sumedenia de fişiere dintr-un proiect. După compilare rezultă noi fişiere. Unele dintre ele, când vă satisface rezultatul final, pot fi şterse. În TEXnicCenter, puteţi curăţa proiectul de fişierele auxiliare folosind Build→ Clean Project. Este recomandabil însă să aveţi undeva (pe dischetă sau CD ar fi cel mai bine) tot proiectul salvat, înainte de a face curăţenie.

Chiar şi curăţat de fişierele auxiliare, un proiect are numeroase fişiere. Cum îl distribuim pe Internet sau îl trimitem prin poşta electronică? S-ar putea ca destinatarii să aibă probleme cu descărcarea fişierelor. Soluţia cea mai bună este să arhivaţi tot proiectul. Selectaţi dosarul proiectului. Deschideţi meniul contextual. Dacă aveţi 7-zip sau alt program de arhivare instalat, chemaţi programul de arhivare, selectaţi opţiunile dorite şi construiţi o arhivă a proiectului. Acum aveţi un singur fişier, lesne de transportat.[30]

2.1.3.1.2 Vim şi sintaxa limbajului LaTeX Ca orice limbaj de programare, LaTeX are o sintaxă care trebuie strict respectată. Daţi un nume de comandă pe care LaTeX nu-l ştie şi veţi primi la compilare un mesaj de eroare. Nu se poate însă face ceva pentru a depista erorile de sintaxă înainte de faza compilării.

Dacă folosiţi Vim cu interfaţă grafică, atunci aţi observat deja faptul că Vim colorează într-un mod special expresiile din limbajul LaTeX. Comenzile sunt evidenţiate prin culori. Acest lucru vă ajută să identificaţi erorile de sintaxă.

De asemenea, folosiţi, în modul comandă normală, tasta % pentru a testa închiderea corectă a parantezelor. Dac-aţi integrat şi scriptul `matchit`, puteţi verifica şi corectitudinea perechilor de comenzi de genul `\begin{document}--\end{document}`.

Sistemul culorilor are limitele sale, dar – în practică – n-am simţit nevoia utilizării unui program special de verificare a corectitudinii sintaxei Vim. Dacă n-aveţi interfaţă grafică, un asemenea program este însă util.

Editorul mediului integrat foloseşte aceeaşi metodă a colorării textului pentru a evidenţia sintaxa programului LaTeX. Vim mi se

[30] Nu vă jucaţi cu toate posibilităţile oferite de arhivare. Există posibilitatea de a transforma arhiva într-un executabil, care se deschide printr-un simplu clic la destinaţie. Unele firme care asigură serviciile de poştă electronică sunt însă suspicioase şi returnează toate scrisorile care au ataşate programe executabile.

pare însă mai flexibil şi de mai mare ajutor atunci când este vorba despre depistarea unei erori subtile de sintaxă.

2.1.3.2 Ziua a doua

Orice program LaTeX se traduce, până la urmă, în comenzi pe care le execută procesorul TeX. Ar fi foarte util să ne amintim că TeX nu este un editor de texte asemenea lui Vim. Este un sistem de aranjare a textului în pagină în vederea tipăririi.

TeX pune pe foaia de hârtie cutii (*boxes*) în care apoi pune litere.[31]

2.1.3.2.1 Modurile LaTeX Oricine s-a jucat cu cutii ştie însă că le poţi fie pune unele lângă altele, fie unele peste altele. Pe foaia de hârtie, care este bidimensională, „cutiile“ au doar un sens metaforic. Evident, sunt doar nişte patrulatere care pot fi dispuse pe orizontală sau pe verticală.

Dacă sistemul dispune cutiile pe verticală, atunci spunem că el se află în modul vertical. Dacă sistemul aşază cutii pe orizontală (pe rânduri), atunci este în modul orizontal.

Atât modul vertical, cât şi modul orizontal au două variante:[32]

vertical obişnuit În acest mod se pot pune una peste alta oricâte cutii, deoarece se trece de la o pagină la alta şi aşa mai departe;

vertical intern Cutiile sunt puse vertical într-o altă cutie şi există o limită dată de dimensiunile cutiei în care se pun alte cutii;

orizontal obişnuit În acest mod se pot înşirui oricâte cutii, deoarece se trece de la un rând la altul;

orizontal strict Cutiile sunt înşiruite în altă cutie şi limita este dată de cutia respectivă, dar nu se produce o trecere de la un rând la altul.

Când ne referim la „vertical“ sau „orizontal“ fără altă precizare avem în vedere variantele obişnuite ale acestor moduri.

Distincţiile din sistemul lui Knuth sunt foarte naturale, în ciuda enunţurilor abstracte de mai sus. Evident, resursele sistemului TeX

[31] Aici doar am reamintit un principiu de bază pentru înţelegerea modului în care funcţionează procesorul TeX. Vezi mai sus pagina 67, rândul 27.

[32] Am folosit terminologia şi explicaţiile din Seroul[11, p.46].

pun o limită în calea înşiruirii de *oricâte* cutii; aici era vorba însă de nişte distincţii de principiu.

Cel de al treilea mod este cel matematic. Într-un eseul filosofic este perfect posibil să întâlniţi formule logice. Acestea trebuie scrise în modul matematic. Acest mod de lucru al TEX nu poate fi învăţat însă în două zile.[33]

text vs. non-text

Dacă am înţeles felul în care lucrează TEX, atunci am înţeles şi modurile de lucru ale LATEX. Putem, de asemenea, să facem şi o distincţie simplă, binară, între un mod „text“ şi un mod „non-textual“. Textul ca atare este aranjat orizontal şi vertical. Veţi vedea şi o serie de comenzi pentru text. Când trebuie specificat explicit că aceste comenzi sunt pentru text LATEX, ele sunt de forma `\text...{}`.

Când porneşte, LATEX este în modul vertical. Cum se schimbă acest mod? Prin comenzi care implică modul orizontal într-una din versiunile sale şi la întîlnirea unuia dintre semnele care nu sunt rezervate.

Prin urmare, la întâlnirea unui semn care este reprodus ca atare în textul tipărit, LATEX trece în modul orizontal.

Pentru a vedea modurile TEX puteţi duce următorul dialog într-o fereastră MS-DOS:[34]

```
>tex
This is TeX, Version 3.141592 (MiKTeX 2.4)
**\tracingcommands=1
*\vbox{}
*a
*\hbox{}
*$x$
*\end
[1]
(see the transcript file for additional information)
Output written on texput.dvi (1 page, 260 bytes).
Transcript written on texput.log.
```

Chiar dacă nu ştiţi comenzile, numele lor este sugestiv: `vbox` sugerează o cutie legată de modul vertical; iar `hbox` este o cutie care are legătură cu modul orizontal. Citirea fişierului de tip `log` este oricum instructivă:

```
This is TeX, Version 3.141592 (MiKTeX 2.4)
**\tracingcommands=1
*\vbox{}
```

[33] A se vedea aici anexa 2.4.

[34] Exerciţiul este pentru ucenicii vrăjitori şi este sugerat de Seroul[11, p.47].

```
{vertical mode: \vbox}
{internal vertical mode: end-group character }}
{vertical mode: blank space  }
*a
{the letter a}
{horizontal mode: the letter a}
{blank space  }
*\hbox{}
{\hbox}
{restricted horizontal mode: end-group character }}
{horizontal mode: blank space  }
*$x$
{math shift character $}
{math mode: the letter x}
{math shift character $}
{horizontal mode: blank space  }
```

Fişierele `log` conţin mesajele sistemului pe parcursul interpretării sau compilării unui program. Este bine să ne obişnuim să le citim. Din nou, TEXnicCenter oferă un mare avantaj: toate mesajele sistemului apar într-o fereastră şi putem naviga de la un mesaj la altul cu ajutorul unor pictograme. Puteţi, de asemenea, da un clic pe programul care vă interesează şi veţi naviga automat către linia (în sens logic) din program care a pricinuit apariţia mesajului.

2.1.3.2.2 Alineatele Dacă aţi făcut tentativa de a scrie din prima zi un text cu mai multe alineate în sistemul LaTeX, aţi avut probabil o mare surpriză. Fără să ştiţi aţi intrat în modul orizontal şi, dacă n-aţi avut noroc, n-aţi putut ieşi din el. Apăsarea (o singură dată!) pe tasta Enter nu vă scoate din modul orizontal.

Dacă nu ieşim din modul orizontal, nu putem construi alineate distincte. Ieşim din modul orizontal dacă apăsăm de două ori tasta Enter. Altfel spus, un rând alb ne readuce în modul vertical.

apăsaţi de două ori Enter pentru a începe un alineat

Există şi o comandă specială pentru crearea de alineate: `\par{}`. N-are rost să creez aici meniuri pentru comenzi atât de simple. Este suficient să aveţi un meniu pentru scheletul unei comenzi LaTeX.[35] Puneţi textul la locul potrivit şi aţi scris comanda.

Unde punem textul alineatului? Putem să-l punem între acoladele comenzii `\par{}`? Se poate, dar n-are sens din perspectiva modurilor TEX. Comanda `\par{}` trebuie pusă la sfârşitul alineatului. Ea spune sistemului că trebuie să treacă în modul vertical. Putem omite acoladele.

Din perspectiva clarităţii sursei programului LaTeX, recursul la

[35] A se vedea aici §1.2.2.1.1.3.

rândul alb pentru a crea alineate mi se pare preferabil. Marele avantaj al faptului că apăsarea pe tasta ENTER nu este echivalentă cu trecerea la un nou alineat[36] este însă altul. Putem pune fiecare idee pe un rând logic distinct.

Un alineat este cărămida de bază a unui text. Fiecare alineat cuprinde un mănunchi de idei care gravitează în jurul unei idei-cheie. Este extrem de avantajos să putem pune fiecare idee pe un rând distinct.

În acest punct se vede cel mai bine de ce soluţia aleasă curent în editoarele de birou este inadecvată pentru munca intelectuală. Editoarele de birou[37] ne forţează să scriem ideile, ca să zic aşa, în modul orizontal, punându-le unele după altele. Această modalitate de scriere este avantajoasă dacă vrem să vedem imediat cum va arăta rezultatul final. Ce-aţi zice însă dac-ar cere cineva ca o clădire să fie construită fără a se recurge la schele, pe motiv că acestea nu ne permit să vedem pe parcurs apropierea de rezultatul final? Dacă respectăm această cerinţă, rezultatul final va semăna mai mult cu o colibă decât cu o clădire frumoasă.

2.1.3.2.3 Textul invizibil Un alineat are o idee-cheie. Putem face ceva, în sursa programului LaTeX, pentru a evidenţia cumva ideile-cheie? Da. Editoarele de birou recurg la colorare. Această metodă este însă rezervată în sursele LaTeX pentru marcarea sintaxei limbajului. Există însă şi o altă posibilitate: textul invizibil (în documentul tipărit în final).

comentarii LaTeX

În surse este bine să fie puse comentarii. Comentariile pot cuprinde, de pildă, ideea-cheie a alineatului. Comanda pentru comentarii este realizată cu ajutorului semnului rezervat %. Tot ce se află între simbolul procent şi primul sfârşit de rând logic devine comentariu şi este ignorat de către compilator. De asemenea, putem pune comentariile pe un rând separat.

Dacă examinăm începuturile de rând din sursa alineneatului de la pagina 86, rândul 9 vedem un comentariu (pe rândul al doilea în extrasul de mai jos) care nu apare în text după compilare (rolul lui este de a-mi permite să sesizez rapid esenţialul alineatului):

```
1 În acest punct se vede cel mai bine de ce soluţia aleasă (...)
2 %editoarele de birou fac obscură logica alineatului
```

[36] Ideea de a nu apăsa de două ori ENTER, preluată de pildă de Funeriu[3, p.282] din Parker, este una dintre dintre absurdităţile pe care le induce în mintea oamenilor utilizarea editoarelor de birou.

[37] Cel puţin cele pe care le ştiu eu.

```
Editoarele de birou ne forţează să scriem ideile (...)
Această modalitate de scriere este avantajoasă dacă (...)
Ce-aţi zice însă dac-ar cere cineva ca (...)
Dacă respectăm această cerinţă, rezultatul final (...)
```

Până la urmă, din punctul de vedere al textului invizibil, diferenţa între editoarele de birou şi LaTeX este una mai mult de abordare decât de substanţă. Editoarele de birou au posibilitatea de a insera text invizibil la tipărire. Vim sau alte editoare adaptate cerinţelor LaTeX colorează într-un mod aparte comentariile.

2.1.3.2.4 Secţiunile Este bine să folosim cât mai mult comentariile. În cele ce urmează vom explica felul în care pot fi create secţiunile unui eseu într-un document LaTeX. Titlul lor nu ne spune însă, după o vreme, chiar totul despre ce am vrut să punem în secţiunea respectivă. Un comentariu ne ajută enorm din acest punct de vedere.

În LaTeX, articolele sunt divizate în secţiuni, subsecţiuni, subsubsecţiuni, paragrafe şi subparagrafe. Nu există capitole în cazul unui articol.

Care sunt comenzile pentru a crea aceste cinci tipuri de secţiuni? Comenzile pot fi desprinse din următoarele linii de cod pentru meniuri Vim:

```
imenu ltx2.section \subsection{}<CR>\label{}<Up><Right>
imenu ltx2.subsection \subsubsection{}<CR>\label{}<Up><End><Left>
imenu ltx2.subsubsection \paragraph{}<CR>\label{}<Up>
<End><Left>
imenu ltx2.paragraph \subparagraph{}<CR>\label{}<Up><End><Left>
imenu ltx2.subparagraph \subparagraph{}<CR>\label{}<Up><End><Left>
```

Puneţi titlul secţiunii între acoladele primei comenzi din perechea de comenzi. Sensul celei de a doua comenzi îl vom explica mai jos.

Termenii din limba engleză pentru ultimele două tipuri de secţiuni sunt susceptibili să genereze confuzii. În limba engleză, un *paragraph* este mai degrabă un alineat. Numele comenzilor sunt mai puţin importante însă. Semnificativ este că acestea sunt toate secţiuni ale unui text. Cea mai mică secţiune grupează un număr de alineate, centrate în jurul unei idei-cheie.

Comenzile de secţionare pot fi date şi-n mediul integrat TeXnicCenter. Recomandarea noastră este să NU faceţi acest lucru. Mediul integrat construieşte automat argumentul comenzii `\label{}` şi pune o secvenţă `sec:` la începutul oricărui argument. Acest obicei

creaţi secţiunile cu Vim

mai mult încurcă însă lucrurile. Cel mai bine ar fi ca argumentul din comanda `\label{}` să fie un unic cuvânt. Dacă puneţi două puncte în argument, atunci nu se mai respectă această cerinţă.

La ce slujeşte comanda `\label{}`? Scrieţi comenzi de secţionare în fişierul `text.tex` al proiectul `eseu`. Nu le scrieţi în fişierul principal. Scrieţi comenzi pentru toate cele cinci tipuri de secţiuni, cu titluri şi ceva text în fiecare secţiune. Transformaţi însă în comentarii rândurile unde se află comenzile `\label{}`. După ce compilaţi, studiaţi efectul obţinut. În textul generat astfel sunt vizibile secţiunile. Primele trei tipuri au şi numere puse automat. Eliminarea comenzilor `\label{}` n-a stricat însă nimic.

label

În limba engleză, *label* înseamnă „etichetă“. Secţionarea textului nu depinde de etichete. Etichetele au cu totul alt rol decât acela de a secţiona textul. De fapt nu au, în sine, nici o legătură cu secţiunile ca atare. Ele sunt marcaje, sunt punctele pe care le caută alte comenzi din program. Dacă vrem să trimitem la o secţiune din text, avem nevoie de aceste marcaje speciale.

Cum ar fi bine să arate aceste etichete? Voi folosi un exemplu chiar din textul de faţă:

```
\subsubsection{Învăţarea Vim într-o zi}
\label{InvatareaVimIntroZi}
```

Cred că acum sunteţi convinse şi convinşi că este o doză de ironie în spatele titlului luat drept exemplu. Vim nu se învaţă într-o zi şi, cu atât mai puţin, LaTeX. Lucrul acesta este însă aici mai puţin semnificativ. Ce se vede însă foarte bine este felul în care în etichetă au fost eliminate spaţiile, diacriticile şi, de fapt, orice alt element care ar putea genera confuzii la căutarea prin text.

De ce pledez atât de mult pentru etichetele simple, compacte? Practic, gândiţi-vă c-aţi construi un şablon, o expresie regulată pentru a căuta etichete într-un program LaTeX. Nu este prea greu să izolezi conţinutul aflat între acoladele comenzii `\label{}`. Este însă mult mai simplu să lucrezi după aceea cu el dacă este un simplu cuvânt.

TeXnicCenter dispune de o fereastră pentru vizualizarea structurii proiectului. De altfel, face acest lucru foarte bine, cu sau fără etichete. Mediul integrat nu se bazează pe ele pentru a explora structura proiectului.

În TeXnicCenter, numai dacă avem un proiect putem explora structura documentului, indiferent de numărul de fişiere. Este un alt argument în favoarea creării unui proiect chiar şi-n cazul unui simplu articol.

În cazul unei cărţi există două tipuri suplimentare de secţiuni. Le vom prezenta tot sub forma codului pentru meniuri Vim:

```
imenu ltx2.part \part{}<CR>\label{}<Up><Right>
imenu ltx2.chapter \section{}<CR>\label{}<Up><Right>
```

Pentru a nu primi mesaje de eroare, trebuie să modificaţi în antetul fişierului principal al proiectului tipul de document. Acesta trebuie să fie `book`.

O carte poate avea părţi şi capitole. Acesta este sensul acestor tipuri de secţiuni ale unui document.

Aici preferăm să ne referim la toate aceste diviziuni ale unui document folosind un număr precedat de simbolul §. Pentru a produce acest semn este nevoie de o comandă specială: `\textsection`.

§

2.1.3.3 Ziua a treia

Atunci când lucrăm cu un editor de birou executăm o serie întreagă de operaţii uzuale de prelucrare a unui text. Acestor operaţii propun să le consacraţi ziua a treia de învăţare a LaTeX.

Pentru început studiaţi câteva meniuri simple:

```
imenu ltx3.emph \emph{}<Left>
imenu ltx3.textit \textit{}<Left>
imenu ltx3.underline \underline{}<Left>
imenu ltx3.textbf \textbf{}<Left>
imenu ltx3.texttt \texttt{}<Left>
imenu ltx3.textsc \textsc{}<Left>
imenu ltx3.textsf \textsf{}<Left>
imenu ltx3.textsl \textsl{}<Left>
```

Prima dintre comenzile de mai sus este probabil cea mai importantă. În ciuda aparenţelor, LaTeX face mult mai multe lucruri în mod automat decât un procesor de cuvinte. Prima comandă îi spune sistemului LaTeX să evidenţieze porţiunea de text cuprinsă în acoladele comenzii (argumentul comenzii), în funcţie de context. LaTeX alege metoda potrivită, nu utilizatoarea. Este metoda recomandabilă de evidenţiere a textului. În anumite puncte ale textului, alte metode nu funcţionează sau nu funcţionează cum vă aşteptaţi.

Pentru a vedea cum funcţionează comenzile de mai sus cel mai potrivit este să dăm un exemplu:

stilurile literelor

```
\emph{Eseul filosofic}\\
\textit{Eseul \emph{filosofic}}\\
\underline{Eseul \emph{filosofic}}\\
\textbf{Eseul \emph{filosofic}}\\
\texttt{Eseul \emph{filosofic}}\\
\textsc{Eseul \emph{filosofic}}\\
\textsf{Eseul \emph{filosofic}}\\
\textsl{Eseul \emph{filosofic}}\\
\textrm{Eseul \emph{filosofic}}
```

Eseul filosofic
Eseul filosofic
Eseul *filosofic*
Eseul *filosofic*
`Eseul filosofic`
ESEUL *filosofic*
Eseul *filosofic*
Eseul filosofic
Eseul *filosofic*

Vă întrebaţi acum, de bună seamă, ce rol au cele două bare oblice inverse. Ele îi spun sistemului că trebuie să forţeze trecerea pe alt rând.

Ce facem dacă vrem, din contră, să evităm ruperea rândului. Dacă este vorba despre un spaţiu alb, atunci punem în locul spaţiului alb o tildă, semnul `~`.

Este momentul acum să arătăm şi cum se obţin semnele rezervate. La urma urmei, ele sunt semne uzuale pentru un editor de birou. Dacă studiaţi exemplul următor, vedeţi că nu sunt greu de obţinut nici în LaTeX:

semne rezervate

```
100\${}\\
XYZ\&{}co.\\
\#{}\\
100\%{}\\
meniu\_{}vim
```

100$
XYZ&co.
#
100%
meniu_vim

Tehnica folosită este cea a transformării simbolurilor rezervate în nume de comenzi LaTeX. Chiar dacă argumentele acestor comenzi rămân vide, are sens să păstrăm acoladele. În acest fel comunicăm sistemului LaTeX unde se termină numele comenzii.

Terminarea numelui unei comenzi poate fi indicată şi printr-un spaţiu alb. Sursa programului LaTeX devine însă mai lizibilă prin folosirea acoladelor.

În cele ce urmează vom vedea şi o excepţie de la regula transformării semnelor rezervate în nume de comenzi.

semne rezervate (continuare)

```
\{{}de aici\\
până aici\}{}\\
\^{}Z\\
\texttt{\~{}}\\
\textbackslash{}label\{\}
```

{de aici
până aici}
^Z
~
\label{}

Bara oblică inversă este obţinută, în modul text, cu comanda `\textbackslash{}`.

LaTeX este sensibil la diferenţa de stil între semnele folosite în modul matematic şi-n modul orizontal, obişnuit sau strict. Acest lucru se vede din prezenţa unor comenzi specifice textului. Unele le-am menţionat deja. Altele pot fi studiate în exemplele care urmează.

semne speciale

```
\textgreater{}\\
\textless{}\\
\textsection{}\\
\textparagraph{}\\
\textcopyright{}
```

>
<
§
¶
©

Pentru a uşura comparaţiile am inclus în cazul ghilimelelor şi versiunile scurte ca să zicem aşa.[38]

ghilimele

```
\textquotedblleft{}\\
``\\
\textquotedblright{}\\
''\\
\textquoteleft{}\\
`\\
\textquoteright{}\\
'
```

“
“
”
”
‘
‘
’
’

Mai există şi alte comenzi importante în LaTeX pe care s-ar putea să le folosim relativ frecvent:

```
\ldots{}\\
-\\
--\\
---
```

...
-
–
—

Din exemplul de mai sus rezultă implicit cum putem spune sistemului să nu rupă punctele de suspensie. Ce facem însă în cazul ortogramelor româneşti? La trecerea de pe un rând pe altul s-ar putea să ne ciocnim de ruperea ortogramei într-un punct nedorit. Felul în care putem proceda ne este sugerat de exemplele următoare:

ortograme

```
într-o într-o într-o într-o într-o
într-o parte într-o zi zi
într-o într-o
într-o
într-o
```

într-o într-o într-o într-o într-o într-o parte într-o zi zi într-o într-o într-o într-o

```
într-o într-o într-o într-o \hbox{într-o}
într-o parte într-o zi zi
\hbox{într-o} \hbox{într-o}
\hbox{într-o}
\hbox{într-o}
```

într-o într-o într-o într-o într-o într-o parte într-o zi zi într-o într-o într-o într-o

Există o subtilitate în utilizarea `\hbox{}`. Conţinutul cutiei este în mod orizontal strict, dar comanda nu face trecerea la modul ori-

[38] Atenţie! Acestea sunt ghilimele standard în limba engleză. Ghilimelele româneşti trebuie construite, după cum vom arăta mai jos.

zontal obişnuit în locul în care este plasată. Putem compara alte două exemple:

```
\hbox{într-o}
\hbox{într-o}
\hbox{într-o}
```

într-o
într-o
într-o

```
\mbox{într-o}
\mbox{într-o}
\mbox{într-o}
```

într-o într-o într-o

În manualul său[4], creatorul LaTeX, Leslie Lamport, recomandă folosirea comenzii `\mbox{}` pentru evitarea ruperii textului.[39]

2.1.3.3.1 Blocurile Există, după cum am văzut, comenzi fără acolade puse după ele. Sunt comenzi care fie nu au nevoie de nici un material (argumentul lor este efectiv vid), fie îşi iau singure materialul pe care-l prelucrează.

O comandă precum `\today` afişează data curentă. Putem să ne dispensăm de acoladele puse după această comandă.

Este oare posibil să folosim, în limbajul LaTeX, acolade care nu sunt precedate de bara oblică inversă şi numele comenzii? Răspunsul este afirmativ. Acoladele separă un *bloc* din programul LaTeX.

Care este rostul blocurilor? Ideea de bază este cea a localizării anumitor proprietăţi ale textului. Cea mai simplă ilustrare este legată de evidenţierea unei porţiuni de text. Să examinăm un exemplu foarte simplu:

```
Eseul{\em filosofic}
```

Eseul*filosofic*

Rolul acoladelor este să delimiteze o porţiune de *text*. Imediat după prima acoladă urmează o declaraţie. Declaraţia este de fapt tot o comandă. Aţi putea pune chiar şi acolade după ea, dar cred că aici acest stil de a scrie programul nu-şi are rostul. Spaţiul alb este preferabil.

Declaraţiile au efect numai în limitele unui bloc de program LaTeX.[40]

Este extrem de uşor să construim un meniu Vim pentru a insera blocuri şi declaraţii. Dăm aici doar rândul de cod pentru evidenţierea textului:

```
1 imenu ltx3.em {\em }<Left>
```

[39] A se vedea chiar listele volante cu comenzi uzuale, pe care le găsim în carte.

[40] Termenul „declaraţie“ este cel folosit chiar de Leslie Lamport. A se vedea Lamport[4, p.27].

A se observa poziţia la care se întoarce cursorul Vim. Rămâne un spaţiu alb între declaraţie şi ce punem în blocul respectiv.

Ce am mai putea declara? Putem specifica forma (stilul) tipurilor de litere.[41] Dacă faceţi exerciţii cu exemplul care urmează, veţi descoperi lesne rostul următoarei serii de declaraţii:

```
{\itshape Eseul filosofic}\\
{\scshape Eseul filosofic}\\
{\slshape Eseul filosofic}\\
{\upshape Eseul filosofic}\\
{\bfseries Eseul filosofic}\\
{\mdseries Eseul filosofic}\\
{\ttfamily Eseul filosofic}\\
{\sffamily Eseul filosofic}\\
{\rmfamily Eseul filosofic}
```

Eseul filosofic
Eseul filosofic
Eseul filosofic
Eseul filosofic
Eseul filosofic
Eseul filosofic
`Eseul filosofic`
Eseul filosofic
Eseul filosofic

declaraţii

Se observă că există o corespondenţă între comenzile prin care se dă o anumită formă literelor din text[42] şi declaraţii.[43]

2.1.3.3.2 Revizia textului programelor LaTeX Se-ntâmplă de multe ori să lucrez şi să uit să subliniez o literă sau să scriu cursiv un cuvânt. Când fac revizia textului descopăr aceste probleme. Ce-i de făcut? Mi-am construit meniurile care urmează tocmai pentru a rezolva genul acesta de probleme.

```
imenu ltx3.cuvantInAcolade {<Esc>ea}
imenu ltx3.cuvantInComanda \{<Esc>ea}
imenu ltx3.cuvantTextsc \textsc{<Esc>ea}
imenu ltx3.cuvantTexttt \texttt{<Esc>ea}
imenu ltx3.cuvantTextit \textit{<Esc>ea}
imenu ltx3.cuvantSubliniat \underline{<Esc>ea}
imenu ltx3.litSubliniata \underline{<Esc>la}
imenu ltx3.litInAcolade {<Esc>la}
```

Felul în care funcţionează meniurile este destul de evident. Treceţi Vim în modul `insert`. Puneţi cursorul în faţa cuvântului sau a literei. Daţi comanda şi asta este tot. Restul este doar exerciţiu.

Comenzile de mai sus au însă un rol destul de limitat. S-ar putea să am cu Vim divergenţe în privinţa ideii de cuvânt. Să modific ceea ce înţelege Vim prin cuvânt? Uneori ar fi absurd să fac aşa ceva. Pentru mine, `text.tex` reprezintă un singur cuvânt. Din păcate, eu folosesc înţelesul termenului respectiv. Vim se orientează după

[41] Acelaşi *font* poate fi normal sau cursiv, aldin şi aşa mai departe.
[42] Vezi mai sus pagina 90, rândul 1.
[43] Observaţia aceasta am preluat-o de la Lamport[4, p.37].

punctul pus în numele fişierului. N-am însă nici un interes să schimb felul în care Vim înţelege rolul punctului.

Toată lumea ştie că editoarele de birou au o soluţie simplă la problema de mai sus: selectezi textul şi apeşi pe un buton virtual; porţiunea selectată îşi schimbă forma pe ecran. Nu este chiar aşa de greu să construim meniuri Vim care ne permit să lucrăm cu porţiuni de text selectate.

```
menu ltx3n.selInAcolade xi{<Esc>pa}
menu ltx3n.selInComanda xi\{<Esc>pa}
menu ltx3n.selTextsc xi\textsc{<Esc>pa}
menu ltx3n.selTexttt xi\texttt{<Esc>pa}
menu ltx3n.selTextit xi\textit{<Esc>pa}
```

Aceste meniuri trebuie studiate ceva mai atent decât cele de mai sus. Ele nu sunt active în modul `insert` şi este firesc să fie aşa. Atunci când selectaţi text, Vim îşi schimbă modul. Când exersaţi, acordaţi atenţie şi deosebirilor dintre efectul comenzilor în modul `(insert)SELECT` şi modul `vizual`, precum şi rolului punctului din care începeţi selecţia.

Cred că acum vestea proastă şi vestea bună în legătură cu LaTeX (şi Vim!) au prins deja contur. Vestea proastă este c-ar trebui să dobândiţi un minimum de deprinderi de programatoare sau programator. Vestea bună este legată de flexibilitatea sistemului şi de posibilitatea de a face (în principiu) orice cu textul tipărit.

Felul în care sunt percepute cele două veşti depinde în cea mai mare parte de câtă programare ştiţi deja. Dacă ştiţi programare, s-ar putea să fiţi uimită sau uimit să vedeţi câte lucruri se pot face fără structuri condiţionale, funcţii, obiecte şi altele. Dacă nu ştiţi, s-ar putea să nu apreciaţi corect cât de multe lucruri face sistemul pentru dumneavoastră şi cât de puţin investiţi, de fapt, în raport cu ceea ce primiţi.

2.1.3.4 Ziua a patra

Prin ce ai deosebi dintr-o privire un eseu academic (filosofic sau nu) de o lucrare de factură neacademică (un articol de ziar, de exemplu)? Notele de subsol, uneori foarte abundente, disting imediat eseul academic de lucrările neacademice.

notele de subsol

Producerea impecabilă de note de subsol este unul dintre atuurile LaTeX. Comanda prin care sunt realizate notele de subsol este foarte simplă. Iată codul pentru un meniu Vim în modul `insert`:

```
imenu ltx4.Nota\ de\ subsol  \footnote{}<Left>
imenu ltx4.Nota\ marginala \marginpar{}<Left>
```

După cum se vede din a doua linie de cod de mai sus, comanda pentru nota care apare pe margine este asemănătoare cu aceea pentru nota de subsol. Dacă este vorba despre un document tipărit pe ambele feţe ale foii (o carte), atunci LaTeX plasează automat nota pe marginea exterioară. O subtilitate a notei marginale o reprezintă posibilitatea de a folosi două valori pentru argumentul comenzii: una pentru marginea din stânga, alta pentru cea din dreapta. Puteţi studia diferenţa plasând comanda `\marginpar[stânga]{dreapta}` în diverse puncte ale textului, în aşa fel încât s-apară când pe pagini pare, când pe pagini impare. Nu lăsaţi una dintre paranteze fără conţinut; sistemul va interpreta paranteza goală ca pe o comandă şi nu va afişa nimic.

2.1.3.4.1 Acolo unde scriau tipografii Închipuiţi-vă că sunteţi într-o tipografie veche. Bătrânul meşter tipograf culege textul literă cu literă. Caută prin cutiile din faţa sa litere şi formează rânduri. La capăt de rând desparte-n silabe, dacă este cazul, şi trece pe rândul următor. Şi tot aşa până termină un alineat şi apoi întreaga pagină de carte. Apare însă o problemă. Textul rândurilor nu se termină exact în marginea din dreapta. Arată mai mult ca textul dactilografiat decât ca un text tipărit.

Într-o altă formă, problema apărea şi-n cazul articolelor de ziar. Articolului i se rezerva un anumit spaţiu în pagina de ziar. Se întâmpla însă ca articolul să fie un pic prea scurt şi să nu umple bine locul rezervat. Nu-i nimic, spuneau tipografii. *Aici scriem noi!*

Tipografii „scriau“, cu spaţii albe. Se punea „albitură“. În zilele noastre un program de aşezare în pagină a textului, cum este LaTeX,scrie şi el precum vechii tipografi.

În mod normal, LaTeX produce o aliniere a textului atât la marginea din stânga, cât şi la cea din dreapta. Uneori nu reuşeşte să încadreze exact textul între margini sau în spaţiul vertical alocat şi compilatorul se plânge de existenţa unor `bad boxes`. Acesta este este cel de al treilea tip de mesaje pe care le primim de la compilator. Putem primi mesaje cu privire la erori (de sintaxă LaTeX), avertismente (de exemplu, ni se atrage atenţia că trebuie să recompilăm sursa) sau aceste mesaje cu privire la textul care nu încape bine în cutii.

Dacă are prea mult text pe un rând, LaTeX spune că a apărut o `overfull \hbox.`. Dacă are prea puţin text pe un rând, LaTeX

spune că a apărut o `underfull \hbox.`. Să studiem puţin exemplul care urmează.

```
Cuvinte magice pentru ucenicii
 vrăjitori când
 nu reuşesc să compileze
 programul:ab:rasintaxadabra
 ab:racompiladabra
```

bad box(es)

Cuvinte magice pentru ucenicii vrăjitori când nu reuşesc să compileze programul:ab:rasintaxadabra ab:racompiladabra

Iniţial sistemul s-a plâns c-au apărut cutii orizontale prea puţin umplute. El nu ştie unde vreau să folosesc două puncte pentru a introduce o enumerare şi unde cele două puncte fac parte din expresia magică. Am pus un spaţiu alb după două puncte, acolo unde introduc enumerarea. Rezultatul nu a fost total reuşit. Sistemul nu ştia cum să despartă-n silabe cuvinte magice. Comanda `\-` îi spune unde **poate** separa silabele. Acum situaţia s-ar îndrepta automat dacă sistemul ar şti că textul este în română. În prima săptămână, este mai bine să lăsaţi sistemul setat doar pentru limba engleză. Dacă faceţi exerciţii cu un exemplu similar, forţaţi despărţirile în silabe după regulile limbii române. Aveţi mai jos exemplul cuvântului „`uceni\-cii`“.

```
Cuvinte magice pentru
 uceni\-cii
 vrăjitori când
 nu reuşesc să compileze
programul: ab:rasin\-taxadabra
 ab:racompiladabra
```

Cuvinte magice pentru ucenicii vrăjitori când nu reuşesc să compileze programul: ab:rasintaxadabra ab:racompiladabra

Cele două casete arată total diferit. Prima este dezastruoasă. Cuvintele sunt dispuse cu vădită stângăcie. A doua este mult mai echilibrată.

Putem scrie, precum tipografii, cu spaţii albe pe orizontală sau pe verticală? Desigur că da. La urma urmei, în LaTeX, putem programa orice acţiune pe care ar executa-o un tipograf de modă veche.

Comenzile folosite sunt `\hspace{}` şi `\vspace{}`. Între acolade trebuie indicată o valoare numerică, fie în unităţi de măsură uzuale, fie în unităţi de măsură tipografice.

Dacă studiaţi puţin vizualizatorul Yap al mediului integrat, vedeţi că este capabil să afişeze atât date cu privire la sursă, cât şi cu privire la paginile pregătite pentru tipar. Mişcaţi cursorul şi, de asemenea, schimbaţi pagina la care vă aflaţi pentru a sesiza felul în care se schimbă aceste date. Poziţia cursorului pe pagină este indicată pe coordonatele carteziene. Unităţile de măsură folosite sunt numite „puncte“. `Yap` foloseşte prescurtarea `pt`.

Acum putem ilustra pe un exemplu introducerea de spaţiu alb

pe orizontală sau pe verticală. Practic, scriu un text, compilez, vizualizez, măsor cu ajutorul Yap şi introduc comenzile pentru a scrie „precum tipografii“.

spaţii albe

```
Eseul filosofic Eseul filosofic\\
Eseul\hspace{68pt} filosofic\par
\vspace{10pt}
Eseul filosofic
```

Eseul filosofic Eseul filosofic
Eseul filosofic

Eseul filosofic

Modul de lucru descris mai sus NU este însă recomandabil. El foloseşte partea cea mai discutabilă din WYSIWYG: ideea de a ne plimba cu cursorul pe ecran şi de a aprecia „din ochi“ poziţia în care plasăm textul.

LaTeX oferă posibilitatea unor calcule precise. Ar trebui să ne facem un proiect de aranjare în pagină. S-ar putea însă ca ziua a patra să aibă o valoare simbolică: suntem încă între lucrul aproximativ (şi comod) şi acribia LaTeX.

2.1.3.4.2 Mediile LaTeX Ce facem însă dacă vrem ca textul să fie aliniat doar la marginea din stânga? Sau dacă dorim să-l aliniem doar la dreapta? Sau să-l centrăm. Trebuie să-i spunem tipografului cum să aranjeze textul.

Pentru a rezolva problema de mai sus putem folosi *mediile LaTeX*. Mediile nu sunt altceva decât o pereche de comenzi `\begin{}` `\end{}`. Perechea aceasta de comenzi se comportă exact ca o pereche de paranteze. Numele mediului este trecut între acolade şi trebuie să fie acelaşi după `begin` şi după `end`.

La ce slujeşte un mediu? Să zicem că vrem să avem un alineat sau mai multe cu text centrat. Atunci împarantezăm textul pe care vrem să-l centrăm între comenzile unui mediu.

Primul exemplu de medii va fi cel pentru centrarea textului, respectiv alinierea la stânga sau la dreapta.

```
\begin{center}Eseul\end{center}
\begin{flushleft}
filosofic
\end{flushleft}
\begin{flushright}
filosofic
\end{flushright}
```

Eseul

filosofic

filosofic

Dacă studiaţi atent exemplele de mai sus, vedeţi că toate cele trei medii, când se încheie efectul lor, fac trecerea la modul vertical. Nu trebuie însă să trageţi de aici concluzia că orice mediu face trecerea la un nou alineat atunci când se termină.

Cum am putea construi meniuri Vim pentru a scrie mai uşor comenzile pentru medii? Mai jos sunt oferite două genuri de soluţii.

```
imenu ltx4.mediuOriz  \begin{}\end{}<Esc>3ba
imenu ltx4.mediuVer \begin{}<CR><CR>\end{}<Esc>2<Up><Right>a
imenu ltx4.mediu <Esc>:let nume=inputdialog("Mediul?\n itemize
\n enumerate\n description\n flushleft\n center\n quote
\n quotation\n verse\n verbatim ")<CR>:let mediu="\\begin{".nume."}
\n\n\\end{".nume."}"<CR>:put=mediu<CR><Down><Home>i
```

Prima soluţie are două versiuni. Mai întâi este creat un meniu care ne ajută să inserăm un mediu într-o linie de program. A doua versiune a primului gen de meniu ne permite să scriem însă într-un mod mult mai lizibil codul aferent unui mediu.

Problema primului gen de meniuri este aceea că trebuie să scriem de două ori numele mediului. Aceasta este adesea o sursă de erori. Uneori uiţi să scrii numele mediului în acoladele lui `end`. Alteori greşeşti ceva într-una dintre instanţele numelui.

meniuri cu casetă de dialog

Al doilea gen de meniuri încearcă să rezolve problema de mai sus. Preţul plătit este un cod Vim ceva mai complicat. Am redat doar varianta de scriere pe verticală a mediului în a doua versiune. Acest meniu face să apară o casetă de dialog. Există chiar şi o listă cu nume de medii. Ea are doar un caracter orientativ. Puteţi să scrieţi ce nume vreţi în caseta de dialog. Trebuie dat apoi un clic pe butonul OK al casetei şi mediul va apărea în fişier. Nu rămâne decât să puneţi ceva între începutul şi sfârşitul mediului.

Nu uitaţi că rândurile *3-6* reprezintă un singur rând logic Vim. Mărimea paginii de hârtie, ca şi ecranul, ne obligă să fragmentăm acest rând în mai mute rânduri vizuale.[44]

Cu ajutorul mediilor pot fi create diverse tipuri de liste. Sintaxa comenzilor este lesne de-nţeles dacă studiaţi exemplul care urmează.

[44] Cred că este mai comod să folosiţi situl cărţii, de unde puteţi prelua codul ca atare, fără să fiţi nevoite sau nevoiţi să retastaţi totul.

```
\begin{itemize}
\item Primul pe listă
\item Al doilea pe listă
\item Al treilea pe listă
\end{itemize}
\begin{enumerate}
\item Primul pe listă
\item Al doilea pe listă
\item Al treilea pe listă
\end{enumerate}
\begin{description}
\item [Primul] pe listă
\item [Al doilea] pe listă
\item [Al treilea] pe listă
\end{description}
```

- Primul pe listă
- Al doilea pe listă
- Al treilea pe listă

1. Primul pe listă
2. Al doilea pe listă
3. Al treilea pe listă

Primul pe listă

Al doilea pe listă

Al treilea pe listă

Alte medii ne permit introducerea de citate în text sau a versurilor. Pentru citate scurte este potrivit mediul `quote`. Pentru citatele mai lungi mediul `quotation` este mai adecvat; el formatează citatul ca pe un text de sine stătător.

LaTeX, care este standardul *de facto* când este vorba despre tipărirea de cărţi de informatică, are desigur şi posibilităţi de reproducere a progamelor sau fragmentelor de program. Liniile de program pe care le vedeţi în această carte sunt realizate cu mediul `verbatim`. Mediul acesta poate fi folosit însă ori de câte ori vrem să dăm unei porţiuni de text aerul de pagină dactilografiată.

2.1.3.4.3 Modul matematic LaTeX este vestit mai ales pentru capacităţile sale de a produce formule matematice. Oricât de repede am trece în revistă sintaxa limbajului LaTeX,nu se poate să nu spunem câteva cuvinte despre modul matematic.

Chiar şi-n texte care nu sunt de factură matematică, vrem adesea să folosim indici. Pentru a face acest lucru în LaTeX trebuie să trecem sistemul în modul matematic. Dacă folosim mediul `math` putem insera fragmentul de text matematic într-un alineat uzual. Mediul `math` face mai întâi trecerea de la modul vertical sau orizontal la modul matematic şi, la ieşire, revine în modul orizontal.

Să vedem practic cum putem scrie indici:

```
Indice suprascris
\begin{math}x^i\end{math}\\
Indice subscris
\begin{math}x_j\end{math}\\
Doi indici
```

Indice suprascris x^i
Indice subscris x_j
Doi indici x_{ij}

Arsenalul de simboluri care pot fi folosite în modul matematic este impresionant. Săgeţile, de pildă, la care am recurs şi noi pe parcurs sunt inserate cu ajutorul modului matematic.

```
A\rightarrow{}B\hspace{50pt}
A\Rightarrow{}B\\
B\leftarrow{}A\hspace{50pt}
B\Leftarrow{}A
```

$$A \rightarrow B \qquad A \Rightarrow B$$
$$B \leftarrow A \qquad B \Leftarrow A$$

Există şi un mod prescurtat de a nota modul matematic care serveşte la inserarea de formule în rândurile obişnuite de text. Este important să menţionăm această notaţie prescurtată pentru că ea dezleagă misterul încă unuia dintre cele zece simboluri speciale din LaTeX. În loc de `\begin{math}\end{math}` putem scrie doar `$$`.

Modul matematic ne va dezvălui secretele încă unuia dintre semnele rezervate. Semnul `&` serveşte la realizarea de tabele. Iar fără tabele nu putem, de fapt, scrie un text matematic ceva mai complicat. O matrice, de exemplu, trebuie construită ca un tabel. Exemplul unei matrici foarte simple încheie, de altfel, incursiunea aceasta în zona scrierii matematice.

matrice

```
\begin{displaymath}
\begin{array}{ccc}
1 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1
\end{array}
\end{displaymath}
```

$$\begin{array}{ccc} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{array}$$

Exemplul necesită câteva comentarii. N-am primi de la compilator un mesaj de eroare dac-am folosi mediul `math`. Pentru afişarea de sine stătătoare a unei formule trebuie să folosim însă mediul `displaymath`. Textul matematic nu mai este încastrat în alineat, lucru care nu prea ar avea, de altfel, sens în cazul matricii.

Mediul `array` nu poate fi invocat decât în modul matematic. Cele sau cei care au învăţat limbajul Pascal la informatică ştiu desigur că `array` are sensul tehnic de tabel.

În cazul mediului `array` trebuie să indicăm nu doar numele mediului, ci şi formatarea coloanelor. De aici prezenţa imediat atunci când începe mediul a unei perechi de acolade. Litera `c` indică faptul că textul de pe coloană trebuie centrat. Sunt trei litere pentru că sunt trei coloane.

Separatorul coloanelor este semnul `&`. Folosirea celor două bare oblice inverse este aidoma celei din modul text. Ele comandă trecerea la un nou rând.

2.1.3.5 Ziua a cincea

Ziua anterioară a fost încărcată, aşa c-am rezervat pentru astăzi elementul care încoronează posibilitatea de a scrie eseuri în stil academic: sistemul trimiterilor. Trimiterile fac posibilă lectura neliniară a eseurilor mai ample.

2.1.3.5.1 Trimiterile în limbajul LaTeX Comanda `\label{}` am amintit-o deja, în contextul discuţiei despre comenzile de secţionare. Rolul ei nu a fost însă discutat.

În limba engleză, `label` înseamnă *etichetă*. Comanda `\label{}` nu este singurul mod de a construi etichete în lumea mai largă a sistemului LaTeX. Vă veţi întâlni cu etichete şi atunci când folosiţi baze de date de tip BibTeX. Rolul unei etichete este de a marca un loc în text. Locul respectiv se află într-o anumită secţiune a textului, pe o anumită pagină sau pe un anumit rând. Ne putem folosi de el pentru a trimite la secţiunea, la pagina sau la rândul respectiv.

De ce nu folosim numărul secţiunii? Motivul este foarte simplu. Numărul secţiunii se poate schimba, dacă modificăm structura textului. Folosim etichete unice. Chiar dacă locul lor în text se schimbă, sistemul poate stabili unde se află în momentul compilării. Atenţie doar la faptul că, în cazul etichetelor, este nevoie de cel puţin două compilări. La o primă compilare sistemul stabileşte locul etichetelor în text. La a doua compilare sistemul face legăturile între trimiteri şi locul către care se face trimiterea.

Care sunt comenzile pentru trimiteri? Pentru a trimite la o secţiune din text se foloseşte comanda `\ref{}`. Argumentul comenzii este o etichetă. Comanda returnează numărul secţiunii (unde este plasată eticheta). În faţa numărului trebuie să puneţi expresia dorită. Puteţi folosi, de pildă, pe `\textsection{}`, care va produce simbolul §. `\ref{}`

Pentru a ne referi la o pagină trebuie să folosim o altă comandă, `\pageref{}`. Argumentul comenzii este tot o etichetă. Spre deosebire de `\ref{}`, rezultatul lui `\pageref{}` este un număr de pagină. `\pageref{}`

Sistemul descris mai sus are un caracter general. Pe măsură ce învăţaţi mai multe despre bibliografie, figuri, tabele, veţi vedea că acelaşi mecanism este folosit pentru a ne referi la intrări din bibliografie, figuri sau tabele.

Cu tot zgomotul în jurul hipertextului şi a paginilor de web, acestea nu fac decât să exploateze pe larg ideea trimiterilor. Ceea ce era folosit doar în lucrările academice a ajuns să fie utilizat de

către toată lumea. Senzaţia produsă de aceste trimiteri pe Internet este una mai ciudată, pentru că, în vreme ce o trimitere de factură tradiţională la un text presupunea eforturi de deplasare şi căutare prin biblioteci, acum textul la care se face trimitere sau imaginea apar cu mare repeziciune pe ecran.

2.1.3.5.1.1 Vim şi navigarea prin intermediul *tag*-urilor

Dac-aţi lucrat cu Vim sau cu alte editoare pentru programare aţi observat în meniuri sau la explicaţiile pictogramelor că există posibilitatea de a construi *tag*-uri. Pe bara cu instrumente din Vim vedeţi o pictogramă cu un ciocan pus peste o etichetă de forma celor care se leagă de valize la aeroporturi. Le voi zice *tag*-uri, pentru a nu le confunda cu etichetele (*labels*).

Vim n-are inclus în *kit*-ul de instalare programul `ctags` care creează *tag*-uri. Acest program trebuie descărcat separat de la adresa de Internet <`http://ctags.sourceforge.net`>. Programul `ctags` este scris de către Darren Hiebert. Acest program poate crea *tag*-uri pentru o sumedenie de limbaje de programare. Versiunea lui Hiebert nu include însă şi LaTeX.

***tag*-uri pentru LaTeX**

Programul lui Hiebert poate fi extins pentru a include şi *tag*-uri pentru LaTeX. Mergeţi la adresa <`http://www.unb.ca/chem/ajit/vim.htm`> şi veţi găsi o versiune pentru Windows a `ctags` care produce şi *tag*-uri pentru LaTeX. Sursele suplimentare necesare pentru a recompila programul lui Hiebert cu sprijin pentru LaTeX sunt scrise de către Ajit J. Thakkar.

Ce faceţi cu `ctags`? Îl puneţi într-un dosar aflat pe un `path`, o cale pe care caută sistemul Windows. Asta este tot. Vim are deja pregătite comenzile pentru crearea de *tag*-uri.

După ce aţi creat *tag*-urile puteţi să examinaţi fişierul `tags`, aflat în dosarul proiectului dumneavoastră. În proiectul cărţii de faţă fişierul respectiv are, de pildă, următorul rând:

```
1 ComenzileNoi anexe05tex.tex /^\\label{ComenzileNoi}$/;"   l
```

Pe acest rând putem desluşi mai întâi numele unei etichete, apoi cel al unui fişier şi o expresie regulată. Litera `l` din finalul rândului ne spune că este vorba despre o etichetă (*label*). Expresia regulată permite Vim să caute eticheta cu numele respectiv.

Acum putem explica raţiunea pentru care am recomandat ca numele etichetelor să nu conţină două puncte. Dacă le conţine, comanda de căutare generată de `ctags` nu va funcţiona în Vim.[45]

[45] Atenţie, de asemenea, la respectarea diferenţei dintre majuscule şi minus-

Dacă daţi, în Vim, un clic pe pictograma cu *tag*-ul galben, atunci când cursorul Vim este plasat pe numele unui *tag*, cursorul se va deplasa la rândul unde este declarată eticheta respectivă sau alt element căruia `ctags` i-a ataşat un *tag*. În meniul `Tools` din Vim găsiţi comanda necesară pentru a reveni la punctul de plecare.

`Tag`-urile sunt atât de utile încât am considerat necesar să modific bara de instrumente Vim în aşa fel încât (în locul pictogramei pentru compilarea de surse) am pus o pictogramă pe care dacă dau un clic cursorul revine la punctul de plecare. Mişcările acestea sunt extrem de utile pentru a verifica dacă argumentul unei comenzi gen `\ref{}` este corect ales. Dacă greşesc numele etichetei, compilatorul emite un mesaj de eroare. Reparaţiile în faza compilării sunt însă mult mai complicate.

verificarea argumentelor comenzii `\ref{}`

Cu ajutorul *tag*-urilor puteţi şi naviga prin sursele LaTeX. Puteţi pune trimiteri şi-n textul invizibil (la compilare) al sursei. De pildă, dacă puneţi numele unei etichete într-un comentariu, puteţi naviga de la acel punct la locul unde este declarată eticheta. Puneţi doar numele, nu redeclaraţi eticheta. Puteţi pune numele etichetei şi imediat după comanda `%` prin care sunt introduse în sursă comentariile.

Programul `ctags` este foarte util şi s-ar putea să vreţi să-l extindeţi. Eu, de exemplu, am adăugat în fişierul `latex.c` scris de Ajit J. Thakkar următoarele rânduri de cod:

```
addTagRegex (language,
"\\linelabel\{[ \t]*([^} \t]+)[ \t]*\}",
"\\1", "l,linelabel",   NULL);
```

Modificarea sursei programului este banală, dar folositoare. Doream să pot găsi şi etichetele care sunt puse pentru a afla numărul rândului de pe pagina generată de către LaTeX.

Compilarea sub Windows s-a dovedit a fi însă mai dificilă decât credeam. Nu ştiu exact cum au reuşit Darren Hiebert şi Anjit J. Thakkar să compileze sursele şi să obţină un program DOS. Cert este că eu n-am reuşit decât compilarea sub Cygwin. Dacă aveţi emulatorul de Unix instalat, daţi obişnuitele comenzi `./configure` şi, apoi, `make`. Partea delicată este că executabilul `ctags` compilat sub Cygwin depinde tot timpul de fişierul `cygwin1.dll`.

Versiunea `ctags` compilată cu cygwin apelează programul `sort` de tip Unix! Trebuie să ştergeţi cu totul programul `sort.exe` din `\Windows\command` sau să modificaţi căile pe care caută Windows

`ctags` şi `sort`

cule. LaTeX şi Vim îşi au rădăcinile în lumea Unix, iar „Unix nu-i ca Windows“. Unix este sensibil la diferenţa dintre „eseu“ şi „Eseu“.

în aşa fel încât să găsească mai întâi pe `sort.exe` de tip Unix. Programul Unix `which` vă poate spune pe ce cale se află un executabil şi, implicit, ce executabil va fi apelat cu prioritate.[46]

`sh.exe` Versiunea compilată cu Cygwin apelează şi programul de tip Unix `sh.exe`. Eu folosesc versiunea instalată de Msys pentru `sh.exe` şi pe cea din GnuWin32 pentru `sort.exe`[47].

Pentru început versiunea programului `ctags` creată de către Anjit J. Thakkar este absolut recomandabilă. Explicaţiile de mai sus au fost doar pentru ucenicii vrăjitori.[48] Experienţa descrisă arată, de altfel, limitele lucrului cu unelte Unix sub Windows. Dacă vreţi să compilaţi programe C/C++ cu sursă deschisă, se pare că Linux este incomparabil mai bun. Atâta vreme cât n-aveţi aceste interese de ucenic vrăjitor experimentat, Windows cred că rămâne o alegere mult mai rezonabilă.

2.1.3.5.2 Comenzile noi Ca la orice limbaj de programare, secretul producerii la nesfârşit de noi expresii rezidă şi-n LaTeX în posibilitatea de a combina elementele deja introduse pentru a realiza noi construcţii.

Aţi remarcat desigur faptul că unele comenzi LaTeX au nume foarte lungi. Se spune, de exemplu, `\textbackslash{}`. Numele acesta lung nu este fără sens. El ne oferă deja o explicaţie a rostului comenzii; ne atrage, de asemenea, atenţia asupra faptului că este o comandă pentru modul text. Dezavantajul este că e dificil de tastat. Nu este greu să definim o nouă comandă, cu un nume mai scurt.

Unii folosesc comenzile noi pentru a abrevia denumiri de organizaţii sau orice fragment de text ceva mai lung şi a economisi timp la tastare. Comenzile noi se introduc după modelul următor:

```
1 \newcommand{\bs}{\textbackslash}
2 \newcommand{\ub}{Universitatea din Bucureşti}
```

Comanda pentru construirea unei comenzi noi are două argumente:

[46] Ordinea în care sunt listate căile în comanda PATH contează!

[47] Programele GnuWin32 au prioritate în `path` faţă de cele Msys, iar Cygwin este instalată autonom, fără posibilitatea de apelare din linia de comandă MS-DOS. Doar fişierul `cygwin1.dll` este evident pus pe o cale unde poate fi găsit de către Windows.

[48] S-ar putea ca unul dintre ucenici să descopere lesne cum se compilează sursele `ctags` fără Cygwin. Eu am încercat cu Mingw şi cu compilatorul C/C++ oferit gratuit, pe Internet, de către firma Borland, dar n-am reuşit. Fără discuţie, nu se pune problema ca ucenicii să pirateze unul dintre compilatoarele comerciale. Acest lucru ar trebui exclus din principiu.

unul pentru numele noii comenzi; altul pentru descrierea, în LaTeX, a acţiunii care este întreprinsă de către sistem în momentul în care este dată comanda respectivă. Efectul comenzilor noi de mai sus este lesne de ilustrat:

```
\bs{}NumeleComenzii\\
\ub
```

\NumeleComenzii
Universitatea din Bucureşti

Comenzile noi (cel puţin pentru versiunea limbajului LaTeX inclusă în distribuţia MikTeX 2.4) pot fi definite oriunde în document. Recomandarea noastră ar fi să nu le plasaţi decât în antetul documentului principal sau într-un fişier special[49].

O altă recomandare ar fi aceea de a nu abuza de prescurtări gen `bs` sau `ub`. Ele scad gradul de lizibilitate al sursei. Putem foarte lesne folosi meniuri Vim pentru a insera în text cu un singur clic texte lungi care se repetă frecvent.

2.1.3.5.2.1 Comenzi noi cu argumente Cum procedăm însă atunci când vrem ca noua comandă să aibă şi ea o parte sau mai multe părţi variabile, unul sau mai multe argumente? Să zicem că vrem o comandă pentru a tipări comenzile LaTeX sub forma unui text. În orice caz ne trebuie o parte variabilă pentru numele comenzii.

S-ar putea ca multă lume să ştie din logică ideea de a compara o variabiă cu o lacună într-un text. În LaTeX asemenea lacune sunt notate cu simbolul `#`. În mod obligatoriu acest simbol trebuie să fie urmat de un număr. Aici numărul este ca şi un nume pentru lacuna respectivă.

Acum s-a dezlegat şi misterul ultimului simbol rezervat. Simbolul `#` serveşte deci la marcarea, în definiţia unei noi comenzi, a locului argumentelor.

Să vedem cum ar arăta practic comanda noastră pentru transformarea comenzilor LaTeX în text:

```
\newcommand{\cmd}[1]{\texttt{\bs{#1}\{\}}}
```

Se observă că, după numele comenzii noi, urmează o paranteză dreaptă în care este specificat numărul de argumente. Noi avem nevoie de un singur argument, pentru numele comenzii. Exemplul care urmează ilustrează folosirea noii comenzi.

```
\cmd{section}\\
\cmd{label}\\
\cmd{ref}
```

```
\section{}
\label{}
\ref{}
```

2.1.3.5.3 Mediile noi Putem extinde nu numai mulţimea comenzilor LaTeX, ci şi pe aceea a mediilor LaTeX. Exemplul nostru este, din nou, foarte simplu. El ilustrează însă principiul general folosit pentru a construi medii noi.

În unele cărţi există pasaje scrise cu litere mai mici. Aceste pasaje discută uneori detalii ale unei probleme. Alteori sunt aprofundate aspecte mai dificile ale chestiunii discutate. Pentru a realiza un mediu în care scriem asemenea pasaje putem folosi următoarea definiţie:

```
1 \newenvironment{caNotaDeSubsol}{\footnotesize }{\normalsize}
```

Comanda prin care definim un mediu nou are trei argumente: numele mediului, o specificare a ceea ce se întâmplă când începe mediul şi o specificare a ceea ce se petrece când mediul se încheie.[50]

Să folosim acum mediul nostru:

```
Text normal.\\
\begin{caNotaDeSubsol}
Text care poate fi omis
la o primă lectură.
\end{caNotaDeSubsol}\\
Din nou text normal.
```

Text normal.
Text care poate fi omis la o primă lectură.
Din nou text normal.

Se observă că mediul face exact ce i-am spus să facă. De pildă, la ieşire, nu rupe rândul, nu creează nici aliniat; doar trece la o literă cu corpul normal în textul respectiv.

Ca şi comenzile, mediile pot fi definite în orice punct al textului. Lucrul acesta poate să aibă sens în cazul exemplului de mai sus, pentru că nu folosesc acest mediu decât în această secţiune. Nu sens să procedăm aşa când vrem să folosim sistematic un mediu.[51]

[49] Pentru fişierul special, care are extensia `sty` a se vedea §2.1.3.6.2

[50] Termenii *begin* şi *end* indică tocmai ceea ce arată şi sensul lor în engleză: un început şi un sfârşit.

[51] Programul `ctags` (vezi aici §2.1.3.5.1.1) este de mare ajutor în reperarea definiţiilor comenzilor şi mediilor. Cu tot sprijinul pe care l-am primi din partea *tag*-urilor în regăsirea definiţiilor, cred că este absolut recomandabil să construim în mod disciplinat sursa LaTeX.

2.1.3.6 Ziua a şasea

LaTeX ne oferă toate avantajele unui limbaj de programare. Desigur, el este prea specializat pe tipărirea de texte pentru a-l folosi comod în alte scopuri.[52] Când este vorba despre tipărirea de texte avantajul TeX şi LaTeX este acela de a ne permite să punem textul în pagină aşa cum dorim.

2.1.3.6.1 Pachetele LaTeX S-ar putea ca multe dintre problemele care ne frământă pe noi să fi fost rezolvate de multă vreme de către alţii. Comunitatea persoanelor care folosesc LaTeX este puternică şi aproape sigur veţi găsi pe cineva care a dat deja o soluţie.

Putem desigur îmbunătăţi rezolvările date de către alţii. Le putem mula după dorinţele noastre. n-are însă rost să „reinventăm roata".

Rezultatele obţinute deja sunt „împachetate" şi doar trebuie instalate, integrate în sistemul LaTeX pe care-l folosiţi. MikTeX instalează o mulţime de astfel de pachete.

Sistemul trebuie anunţat când vreţi să folosiţi un anumit pachet. Persoanele care ştiu un limbaj de programare gen Pascal sau C/C++ sunt probabil intrigate de faptul c-am tot insistat asupra calităţilor ce limbaj de programare ale TeX/LaTeX. Unde sunt structurile condiţionale? Puneţi în antet[53] comanda următoare:

```
1 \usepackage{ifthen}
```

Comanda cere sistemului să utilizeze pachetul `ifthen`.

contoare

Sistemul are o serie de contoare în care ţine date importante pentru aşezarea în pagină. Aşa este, de pildă, contorul `page`, unde este ţinut numărul paginii curente.

[52] Leslie Lamport vorbeşte tot timpul despre LaTeX ca despre un sistem de *macro*-uri TeX. Este o doză de modestie exagerată în acest fel de a vorbi. Pe de altă parte, dacă ţinem cont că *macro*-urile ne permit definirea altor *macro*-uri, vedem că este vorba de un limbaj. În sens strict, un *macro* este o serie de instrucţiuni care pot fi apelate apăsând o tastă sau folosind un cuvânt-cheie. Procesoare de cuvinte din complexele de programe pentru birou dau posibilitatea utilizatorilor de a crea *macro*-uri. Termenul „macro" sugerează ideea că ne aflăm la un nivel superior celui *micro*, unde operează limbajul de programare propriu-zis. Dacă tratăm LaTeX în acelaşi fel, atunci limbajul propriu-zis este TeX. Noi preferăm să tratăm LaTeX ca pe un limbaj de nivel înalt şi TeX ca limbajul procesorului de texte.

[53] Dacă puneţi această comandă în corpul documentului, veţi primi un mesaj de eroare de la compilator.

Putem chiar afişa numărul paginii scriind `\thepage{}` în text. Prefixul `the` se pune în faţa oricărui contor, atunci când vrem să-i aflăm valoarea. Putem afla valoarea din contor cu ajutorului lui `\value{page}`, dar această comandă o putem folosi doar în argumentul unei alte comenzi, cum ar fi `\isodd{}`, care ne spune dacă un număr este impar sau nu.

Acum puteţi scrie o comandă al cărei rezultat ne va spune dacă suntem pe o pagină pară sau pe una impară:

```
\ifthenelse{\isodd{\thepage}}
{Pagina este impară!}
{Pagina este pară!}
```

Pagina este pară!

Comanda aceasta, cu unele îmbunătăţiri,[54] vă poate fi de real folos dacă vreţi ca aspectul unei pagini pare să difere de cel al unei pagini impare. De exemplu, s-ar putea să vrem să plasăm diferit o imagine. Nu trebuie să-mi fac griji dacă modific documentul. Structura condiţională va asigura modificarea aspectului paginii.

Comanda `\ifthenelse{}{}{}` are trei argumente: o condiţie, un argument care este executat atunci când condiţia este adevărată şi unul care este executat când ea este falsă.

Unde se află, de fapt, `ifthen`? Cum l-am putea găsi? Căutaţi în dosarul unde este instalată distribuţia MikTEX un fişier denumit `ifthen.sty`. Îl găsiţi pe o cale `tex\latex\base`. Face parte din nucleul sistemului LATEX. Acest nucleu, în întregimea sa, este tratat, la instalare, ca un pachet unic, denumit `ltxbase`. Nu puteţi însă apela direct acest pachet. Trebuie să apelaţi componentele sale.

Cum putem afla ce pachete suplimentare a instalat sau poate instala MikTEX? În versiunile 2.3 şi 2.4 folosiţi MikTEX Package Manager. Cu ajutorul acestui program puteţi găsi informaţii despre pachete.

Trebuie să treacă însă ceva timp până puteţi folosi pachete ceva mai complicate. În orice caz trebuie citită documentaţia aferentă şi testat totul pe fişiere destinate exerciţiilor, înainte de a trece la utilizarea propriu-zisă.

Nu modificaţi în nici un caz fişierul `ifthen`. Dacă simţiţi nevoia

[54] Rezultatul comenzii `\thepage` depinde de generarea numărului paginii; în partea de sus a paginii veţi găsi în `page` încă numărul paginii anterioare (a se vedea explicaţia în Lamport[4, p.135]). Soluţia sugerată de către Lamport[4, p.196] este să declarăm o etichetă de genul `aceastaPagina` şi apoi să dăm condiţiei forma `\isodd{\pageref{aceastaPagina}}`. Atenţie şi la necesitatea de a da mai multe compilări pentru a obţine rezultatul dorit. Citiţi atent avertismentele din mesajele compilatorului!

să experimentaţi, faceţi o copie a fişierului, redenumiţi-o şi puneţi-o în dosarul `localtexmf`[55].

Nu aş sfătui totuşi pe nimeni să experimenteze cu `ifthen`. Un exemplu mai practic este dat în secţiunea care urmează.

2.1.3.6.1.1 Letrinele Unora le place să înceapă secţiunile mai importante, cum ar fi capitolele unei cărţi, cu o literă supradimensionată, care are înălţimea a două-trei rânduri. Această literă se numeşte *letrină*.

Am putea meşteri, urmând indicaţiile din literatură,[56] o comandă pentru letrine:

```
\newcommand{\letrina}[3]{\noindent\hangindent=#1
\hangafter=-#2\hskip-#1\smash{\hbox to #1
{#3\hfill}}\ignorespaces}
```

```
\letrina{15pt}{2}
{\Huge E}seul filosofic
```

Eseul filosofic

Exemplul acesta este dat special pentru a arăta ce util ar fi un pachet special dedicat rezolvării problemei cu care ne confruntăm.

Pentru exerciţiul real încărcaţi pachetul `lettrine`.

```
\lettrine[lines=1]{E}seul
 filosofic
```

Eseul filosofic

Soluţia lui Daniel Flipo, autorul pachetului `lettrine`, este evident mai uşor de folosit. Dacă studiaţi documentaţia aferentă pachetului, veţi descoperi o mulţime de posibilităţi de a crea letrine.

2.1.3.6.2 Fişierul de tip `sty` Comenzile de utilizare a pachetelor, definiţiile de noi comenzi şi de noi medii sunt reutilizabile. Dacă le-am pune într-un fişier, am putea muta lesne fişierul respectiv dintr-un proiect într-altul. În plus, fişierul respectiv ar face ca antetul documentului principal să fie mult mai concis.

Extensia fişierului în care punem pachetele pe care le utilizăm şi definiţiile comenzilor şi mediilor este `sty`.

Sintaxa fişierelor de tip `sty` diferă de cea a din fişierele de tip `tex`. Diferenţele faţă de ceea ce am văzut până acum sunt uşor de sesizat, dacă cercetaţi cu atenţie exemplul de mai jos. Am extras chiar o parte din fişierul `sty` folosit pentru această carte.

[55] Acesta este, de regulă, numele dosarului unde stau fişierele create local sau pachetele instalate suplimentar de către dumneavoastră. Nu uitaţi să folosiţi programul `MikTeX Options` pentru a reîmprospăta baza de date a sistemului TeX.

[56] Aici folosim ideile lui Seroul[11, p.83]

```
1 \ProvidesPackage{eseu}
2 \RequirePackage{ifthen}
3 \RequirePackage{lettrine}
4 \RequirePackage{url}
5 %==========================
6 \setcounter{secnumdepth}{5}
7 %==========================
8 \newcommand{\bs}{\textbackslash}
```

Pe primul rând se arată modul în care se comunică LaTeX care este pachetul furnizat. Apoi se vede cum, în loc de `\usepackage{}` trebuie folosită comanda `\RequirePackage{}`

În fişierul `sty` putem defini comenzi sau medii noi. Putem, de asemenea, modifica elemente ale sistemului: valori ale unor variabile, definiţii ale unor comenzi şi medii. Pe rândul 6 din exemplul de mai sus este modificată valoarea până la care LaTeX dă numere secţiunilor unei cărţi.

Dacă redefiniţi o comandă trebuie să faceţi acest lucru în mod explicit: folosiţi `\renewcommand{}`. Dacă redefiniţi un mediu, folosiţi `\renewenvironment{}`.

Utilizarea fişierului `sty` creat este foarte simplă. În cazul fişierului `eseu.sty`, în fişierul principal al proiectului introducem o comandă `\usepackage{eseu}`. Acest mod de a proceda este foarte logic; la urma urmei, fişierul `sty` creat constituie pachetul cu soluţiile specifice proiectului respectiv.

2.1.3.7 A şaptea zi

Dac-aţi ajuns până aici, nu doar cu lectura, ci şi cu exerciţiile, este foarte bine. Aproape c-am putea da comanda TeX `\relax`. Această comandă este dată atunci când argumentul trebuie să fie o acţiune, dar – în fapt – sistemul n-are nimic de întreprins.[57]

Oricine utilizează un procesor de cuvinte obişnuit, la nivelul cerut de un eseu filosofic curent (fără tabele, formule, imagini), poate să obţină acum cu LaTeX tot ce ar obţine şi cu editorul de birou. Cu unele excepţii! Ce facem dacă vrem să schimbăm structura paginii, de pildă? Acest lucru este uşor de făcut cu editorul de birou.

[57] Seroul[11, p.288] explică utilitatea deosebită a acestei comenzi. Fără a intra în detalii, putem spune că ea serveşte la eliminarea unor ambiguităţi care ar putea fi generate prin combinarea comenzilor TeX şi a textului documentului. Sistemul trebuie să ştie unde să se oprească. În documentele scrise în LaTeX nu veţi vedea această comandă. Dacă vă uitaţi prin fişiere `sty` sofisticate aveţi destul de repede şansa să vă întâlniţi cu ea.

Răspunsul la întrebarea de mai sus îl vom da în continuare. Ar trebui precizat însă că, mai important, în acest moment, este să treceţi în revistă întreg complexul de resurse oferit de către LaTeX.

2.1.3.7.1 Structura paginii Dacă scrieţi o lucrare de seminar sau o lucrare de diplomă, recomandarea noastră fermă este să nu modificaţi dimensiunile standard furnizate automat de către LaTeX. Sistemul este special conceput pentru a ne bate cât mai puţin capul cu forma textului. Autoarea sau autorul răspund de conţinut, LaTeX se îngrijeşte de formă.

Ce ne facem însă dacă intră în joc alte constrângeri? O carte are, de exemplu, un format cu particularităţile sale. În acest caz, este acceptabilă modificarea structurii paginii.

LaTeX n-ar fi deloc un limbaj comod dacă am vrea să scriem programe care să rezolve ecuaţii. Nu aceasta este menirea sa. Formatul comenzilor, numele lor ample şi explicite sunt însă extrem de utile pentru tehnoredactare. Rândurile de mai jos sunt extrase din fişierul `sty` al cărţii de faţă. Ele aproape nu necesită comentarii.

```
\oddsidemargin=2.2mm
\evensidemargin=10mm
\topmargin=0mm
%\headwidth  %vezi mai jos
\headheight=5.5mm
\headsep=4mm
\textheight=200mm
\textwidth=112mm
\marginparsep=3mm
\marginparwidth=20mm
\footskip=5.5mm
\marginparpush=3mm
\hoffset=17.5mm
\voffset=12mm
\paperwidth=210mm
\paperheight=295mm
%=======================================
\addtolength{\headwidth}{\marginparsep}
\addtolength{\headwidth}{\marginparwidth}
```

Ultimele două comenzi măresc lăţimea antetului paginii, pentru a acoperi şi coloana destinată notelor marginale. Altfel, `headwidth` coincide cu lăţimea textului.

2.1.3.7.2 Ce mai ştie LaTeX? LaTeX lucrează cu cutii (*boxes*) pe care le aşează pe orizontală şi verticală. Dacă umplem o cutie cu

cerneală şi aşezăm lângă ea alte cutii şi aşa mai departe ar ieşi un desen. LaTeX poate executa desene în acest mod.

Recomandarea noastră ar fi să folosiţi LaTeX doar pentru a trasa linii şi forme geometrice elementare. Partea forte a LaTeX o constituie aranjarea textului în pagină, nu desenarea de figuri. Figurile pot fi create cu alte programe şi apoi pot fi integrate în text.

Mari probleme creează desigur tipurile de litere, tabelele, imaginile. LaTeX are soluţii, pe care le prezentăm pe scurt în anexe separate.

Pe lângă compilatorul de LaTeX ca atare, orice distribuţie LaTeX include două programe absolut formidabile: un program pentru lucrul cu bibliografia şi unul pentru crearea de indici. Împreună cu aceste programe LaTeX vă permite să creaţi lucrări care au cu adevărat forma unor lucrări academice. Şi aceste programe sunt prezentate aici în anexe separate.

Pe lângă programe, există în distribuţia MikTeX şi un alt element preţios: documentaţia. Pentru început, identificaţi modul în care puteţi accesa documentaţia şi răsfoiţi câteva titluri.

documentaţia MikTeX

Căutaţi în dosarul MikTeX calea `texmf\doc\guides`. Veţi găsi acolo un dosar `lshort-english`. În el se află o carte scrisă de Tobias Oetiker, Hubert Partl, Irene Hyna şi Elisabeth Schlegl *The Not So Short Introduction to LaTeX2e*. Este o introducere foarte bine făcută şi puteţi să o folosiţi cu mare succes pentru a consolida cunoştinţele de bază în materie de LaTeX. Documentaţia include versiuni ale acestei cărţi în mai multe limbi.

Sursele cărţii lui Oetiker et al. se află în dosarul `\texmf\source \lshort-english`. Dezarhivaţi-le şi creaţi, cu ajutorul TeXnicCenter un proiect având fişierul `lshort.tex` ca fişier principal. Dacă aveţi o instalaţie completă, după două-trei compilări, obţineţi un fişier de tip `dvi` acceptabil. Cartea este concepută pentru a fi transformată în `pdf`. Prefer `dvi`-ul în combinaţie cu TeXnicCenter pentru că astfel pot fi studiate în condiţii optime sursele. Vizualiza-ţi `dvi`-ul; daţi un dublu clic în punctul care vă interesează şi studiaţi sursa.

Sunt o mulţime de elemente interesante în documentaţie. Ne limităm aici doar la două recomandări suplimentare. Restul puteţi descoperi şi singure sau singuri.

Citiţi articolul lui Guido Gonzato „LaTeX for Word Processor Users“ din dosarul `latex4wp`. Veţi înţelege cum puteţi produce cu ajutorul LaTeX tot ceea ce puteţi realiza cu ajutorul unui procesor de cuvinte. Cu alte cuvinte, aflaţi de ce n-aveţi nici un motiv să regretaţi trecerea de la editorul de birou la LaTeX.

Un articol scurt, dar extrem de interesant, este cel al lui Piet van Ostrum, „Page Layout in LaTeX“. Îl găsiţi, în dosarul `fancyhdr`, pe calea `\texmf\doc\latex` . Articolul oferă explicaţii pertinente cu privire la structura paginii.

2.1.3.7.3 Forţa cooperării dintre Vim, TeXnicCenter şi Yap Să zicem c-aţi compilat cartea scrisă de Oetiker et al. Aţi găsit însă o comandă ciudată în surse. Deschideţi fişierul respectiv în Vim. Creaţi fişierul `tags`. Căutaţi acum comanda respectivă.

Dacă vreţi să aflaţi, de exemplu, cum a fost definit mediul `code`, puteţi face acest lucru cu ajutorul *tag*-urilor. Definiţia este, de altfel, uimitor de simplă: este vorba de folosirea unui mediu standard pentru citate.

Dar dacă vreţi să aflaţi ce sens are comanda `\verb`. n-are *tag*! Semn că este un element standard din LaTeX. Reveniţi în TeXnicCenter. Puneţi cursorul pe cuvântul respectiv şi apăsaţi tasta F1. **F1** În câteva clipe vă este afişată într-o fereastră specială explicaţia comenzii.

Vim şi cu TeXnicCenter constituie o combinaţie foarte puternică. Aveţi atât acces direct la surse, cât şi deplasarea între sursă şi programul de vizualizare (în ambele direcţii). În Vim puteţi crea pentru fiecare document sau gen de document la care lucraţi meniuri adaptate cerinţelor dumneavoastră. Puteţi naviga prin surse cu ajutorul *tag*-urilor. Puteţi ajunge în Vim şi din programul de vizualizare.[58]

De ce n-am folosi totuşi un program de tip WYSIWYG? Există atât programe gratuite, cât şi programe comerciale pentru lucrul în LaTeX în acest mod (cel al editoarelor uzuale de birou).

Lyx este un program utilizat, de regulă, sub Linux pentru a edita în maniera WYSIWYG documente LaTeX. Lyx poate fi funcţiona şi sub Windows, cu ajutorul Cygwin. Trebuie să aveţi instalat însă şi emulatorul de X Window. X este interfaţa grafică tipică pentru sistemele Unix.

Forţa combinaţiei dintre un editor precum Vim şi un mediu integrat pentru compilare aşa cum este TeXnicCenter ni se pare însă net superioară. Posibilităţile de lucru sunt mai bune. Separarea dintre conţinutul şi forma textului este clară. În plus, prelucrarea fişierelor în Vim este extrem de sigură. Orice sistem integrat are fragilitatea sa şi, la lucrul cu fişierele mari, s-ar putea să vă producă surprize neplăcute.

[58] Vedeţi explicaţiile din §2.1.2.3.1.1.

2.2 BibTeX

Dacă o persoană scrie un program de calculator, primul lucru la care se gândeşte este cum ar proceda „manual“. Dacă ştii să rezolvi „cu mâna“ problema, atunci poţi trece la faza în care extragi algoritmul, ideea din spatele soluţiei, şi o transferi către programul de calculator. Principiul care stă la baza înţelegerii elaborării şi tehnoredactării computerizate a textelor este acelaşi.

2.2.1 Primii paşi în lumea bazelor de date

Modul tradiţional de alcătuire a unei bibliografii presupunea o colecţie de fişe bibliografice şi o listă alcătuită la finalul lucrării pe baza acestor fişe. Elaborarea unei bibliografii computerizate merge pe aceeaşi idee.

Din punctul de vedere al programării lucrurile se complică însă. Până acum am avut de a face cu o sursă care poate ori să fie interpretată, în care caz se execută pas cu pas comenzile din sursă, ori să fie compilată, în care caz sursa este transformată într-un fişier de alt tip.

bibliografiile

Bibliografiile ne pun în faţa unei situaţii diferite. Chiar din perspectiva tradiţională, fişele bibliografice erau o *bază de date*. În baza de date, pe fişe, câmpurile cu informaţii pot fi dispuse în cu totul alt mod decât în lista bibliografică. De asemenea, în listă putem prezenta doar o parte din informaţiile din baza de date.

Prin urmare, în computer, ne vom confrunta cu o diversificare a fişierelor şi a programelor care lucrează cu ele. Unele fişiere vor stoca baza de date ca atare. Altele vor fi surse (programe) pe baza cărora vor fi extrase şi prezentate date.

Din punctul de vedere al programelor care lucrează cu bazele de date, principiul tare al surselor deschise simplifică lucrurile. Fişierele care conţin datele sunt fişiere de tip text. Noi le vom prelucra cu ajutorul Vim.

O fişă bibliografică tradiţională are însă câmpuri! Într-un fişier de tip text nu putem însă trage linii pentru a separa câmpurile. Se folosesc, în schimb, separatorii. Separatorul este un semn cu ajutorul căruia distingem câmpurile.

Un exemplu ne va lămuri lesne despre ce este vorba. Iau trei cărţi dintre cele care sunt în bibliografia cărţii de faţă şi construiesc un fişier cu următoarele câmpuri: o prescurtare făcută din numele autoarei sau autorului, numărul total de pagini, numărul de pagini din

fişier `csv`

partea introductivă a cărţii[59], numărul de pagini din partea principală a cărţii, numărul de pagini alocate indicelui. Ca-n orice înşiruire aceste date sunt separate prin virgulă. Fiecare rând corespunde unei fişe tradiţionale:

```
Bazerman 1989,527,14,501,12
Fischel 1984,299,10,280,9
Hairston 1974,364,13,344,7
```

Un asemenea fişier se numeşte fişier de tip `csv`[60].

În funcţie de interesele noastre, separatorul ar putea fi şi altceva decât virgula: un spaţiu alb, un semn mai deosebit. Extensia fişierului este însă întotdeauna `csv`.

Există o serie întreagă de operaţii care pot fi făcute cu un fişier `csv`. Fişierul de mai sus poate fi, de pildă, sortat în funcţie de numărul total de pagini. Operaţia este absolut similară sortării fişelor tradiţionale de hârtie.

Tot aşa cum din baza tradiţională de date putem extrage anumite informaţii, din fişierul de mai sus am putea construi liste cu numărul de pagini al părţii introductive sau liste cu numărul de pagini al indicilor.

Bazele de date pot fi punctul de plecare pentru calcule. Pornind de la informaţiile de mai sus am putea, de exemplu, calcula procentul din total al numărului de pagini al indicelui. Folosind fişiere similare am putea face, pornind de la datele lor, calcule statistice mai complicate sau le-am putea folosi pentru a testa ipoteze statistice.

Un program simplu de operare cu fişiere `csv` este `CSVdb`, scris de Sam Francke.[61] Programul este *cardware*[62].

Este recomandabil să folosiţi acest program sau un program similar pentru a vă face practic o idee despre fişierele `csv`. Cu programul lui Sam Francke puteţi sorta, schimba ordinea coloanelor, elimina fişele[63] care se repetă. Este posibilă, de asemenea, utilizarea a diverşi separatori.

[59] Ceea ce în engleză se numeşte *frontmatter*.

[60] De la sintagma englezească *comma separated values*.

[61] Pagina de web a programului este `http://home.hccnet.nl/s.j.francke/software/software.htm`

[62] Dacă folosiţi programul şi vă place, atunci aveţi obligaţia (morală) de a-i trimite autorului o vedere la adresa indicată în documentaţia programului.

[63] În jargonul computerelor fişelor tradiţionale li se spune „înregistrări“, de la englezescul *records*.

2.2.2 Sistemul BibTeX

Un fişier `csv` are însă o structură foarte incomodă din punctul de vedere al fişelor bibliografice. Fişele bibliografice sunt de diverse tipuri. Fiecare tip are câmpurile sale specifice. Câmpul pentru titlul revistei n-are sens la o carte, de exemplu.

Am putea lăsa necompletate câmpurile care nu au sens. Soluţia nu este nici elegantă, nici logică. Logic ar fi să distingem explicit tipurile de fişe în formatul electronic.

Probabil că atunci când aţi instalat TeXnicCenter aţi observat că acesta integrează şi un compilator botezat BibTeX. Acesta este capabil, pe baza unor programe, să producă fişiere care pot fi utilizate de LaTeX pentru a genera liste bibliografice.

fişier `bib`

Pentru început, trebuie studiate fişierele care alcătuiesc baza de date propriu-zisă. Aceste fişiere au extensia `bib`. Vim are capacitatea de a le colora într-un mod care ne ajută să ne dăm seama dacă fişiere sunt construite corect sau nu.

În formatul `bib` tradiţionala fişă bibliografică a unei cărţi are un aspect de genul următor:

```
@BOOK{latex,
  author={Leslie Lamport},
  title={\LaTeX{}: a document preparation system},
  address={Reading, Massachusetts},
  publisher={Addison Wesley Longman,Inc.},
  year={1994}
}
```

Structura de mai sus are forma `@TIP{eticheta, câmpuri}`. Câmpurile sunt separate prin virgule. Fiecare câmp are un nume, urmat de semnul egal, după care este pus conţinutul propriu-zis al câmpului.

Rolul etichetelor este exact acelaşi ca şi-n cazul unui argument al comenzii `\label{}` din LaTeX sau al unui *tag*. Eticheta unei înregistrări ne permite să regăsim înregistrarea respectivă într-un fişier.

Numele câmpurilor sunt suficient de sugestive pentru a nu necesita explicaţii. În cazul cărţii date drept exemplu aceste câmpuri asigură informaţia bibliografică minimă în sensul specificat aici.

Pentru a facilita crearea de fişiere `bib` putem extinde meniurile Vim. Iată un exemplu pentru o intrare de tip BOOK:

```
:imenu Bib.&Book @BOOK{,<CR>author={},<CR>title={},
<CR>address={},<CR>publisher={},<CR>year={},
<CR><Home>}<Esc>6<Up><End>i
```

Articolele sunt adesea prezente în bibliografiile eseurilor filosofice. Prin urmare, este important să oferim şi aici un exemplu de fişă bibliografică electronică pentru un articol. Această fişă ia forma unei înregistrări într-un fişier `bib`, ca-n exemplul care urmează:

```
@ARTICLE{FlewTCE,
  author={Antony Flew},
  title={Issues in Teaching Contemporary Ethics},
  journal={Teaching Philosophy},
  month={Summer},
  year={1975},
  volume={1},
  number={1},
  pages={55--60},
}
```

Cum este şi normal, în cazul unui articol, trebuie menţionată publicaţia periodică cu datele aferente.

Spaţiul nu ne-ar permite să descriem pe larg diverse tipuri de intrări într-un fişier `bib`. Din fericire, principiul sursei deschise oferă o excelentă posibilitate de documentare. Dacă mergeţi la dosarul unde este instalat sistemul MikTeX, veţi găsi în `\texmf\bibtex\bib` o mulţime de exemple de fişiere `bib`.

În cazul în care nu există o descriere standard, trebuie să scriem una. Pentru documentele de pe Internet am construit structura de mai jos. Ea nu poate fi utilizată decât împreună cu `simplu.bst`, o versiune pentru limba română a lui `plain.bst`. Fişierul `simplu.bst` este creat special pentru această carte.[64]

```
@WEBBED{,
  author={},
  title={},
  notatxt={},
  url={},
  year={},
  download={},
  note={},
  annote={},
}
```

Câmpurile clasice sunt primele două. Ca orice document, un text aflat pe Internet a fost scris de către cineva sau o organizaţie şi-a asumat răspunderea pentru el. De asemenea, are un titlu. Dacă aceste elemente lipsesc, nu cred că are sens să figureze în bibliografia

[64] Pentru `simplu.bst` v. situl cărţii.

unui eseu filosofic. Ar putea fi cel mult menţionat în note, cu titlu de sursă a unui exemplu sau de sursă a unei opinii sau afirmaţii.

În câmpul `notatxt` punem o notă despre situl pe care se află textul sau despre caracterul textului.[65] Conţinutul acestui câmp fiind mai flexibil putem să-l reglăm în aşa fel încât să scăpăm de cutiile cu litere care depăşesc zona alocată textului bibliografiei(*bad boxes*).

În câmpul `url` trebuie pusă adresa de Internet. Acesta este un câmp absolut necesar, dar nu suficient pentru o astfel de fişă. Menţionarea anului în care documentul a fost publicat pe Internet ar fi o completare utilă.

Documentele de pe Internet au o natură dinamică. Se schimbă des. De multe ori este greu să fie identificată versiunea. Nu de puţine ori însă, după ce au fost disponibile o vreme, dispar. De aici necesitatea câmpului `download`, în care trebuie trecută data la care a fost descărcat[66] documentul. Recomandarea noastră ar fi să fie folosit un format al datei acceptabil pentru LaTeX şi care nu recurge la numele de luni, pentru a nu crea confuzii cu formatul folosit în bibliografie pentru data publicării documentului. Un astfel de format ar fi, de pildă, implicit în: 14/09/2003.

În sfârşit, structura de mai sus este prevăzută cu un câmp pentru o notă şi unul pentru adnotare. Nota poate să fie folosită pentru un supliment de informaţii privind documentul. Adnotarea are rolul uzual, acela de a oferi o scurtă descriere a documentului.

2.2.2.1 Utilizarea de sine stătătoare a BibTeX

Un fişier bib poate fi utilizat şi independent de sistemul LaTeX. Soluţia care mi se pare cea mai bună, sub Windows, este cea a utilizării programului BibDB, creat de către Eyal Doron.[67]

Chiar şi o bibliografie cu peste 16000 de înregistrări, cum este cea creată de către Richmond Thomason pentru lucrările de filoso-

[65] Aceste informaţii le-am reţinut pe descrierea din fişa bibliografică). Pentru folosirea în citări de sine stătătoare a acestei note a se vedea reguli ale citării tradiţionale.

[66] Nu uitaţi că şi atunci când aveţi sentimentul că doar vedeţi documentul, el tot este descărcat temporar.

[67] <`http://www.tcisoft.com/tcisoft/bibdb.html`> este pagina de web a programului BibDB. BibDB este un program cu surse deschise. Limbajul surselor este Pascal.

fia limbajului, semantică, inteligenţă artificială şi domenii conexe,[68] este manevrată cu mare uşurinţă de către BibDB.[69]

Folosirea interfeţei grafice a BibDB este relativ facilă. Dacă v-aţi însuşit şi utilizarea expresiilor regulate, atunci căutarea în baza de date va decurge foarte uşor.

2.2.3 Stilurile bibliografice

Datele dintr-un fişier `bib` pot fi extrase şi prezentate în diverse moduri, indiferent de felul sau ordinea în care apar ele în fişierul respectiv. Modurile acestea de prezentare a datelor bibliografice se numesc „stiluri bibliografice“.

Descrierile stilurilor bibliografice sunt plasate în fişiere care au extensia `bst`. Ele sunt scrise într-un limbaj special, inventat de către creatorul programului BibTeX, Oren Patashnik. Dacă nu ştiţi ceva mai multă programare sunt puţine şanse să înţelegeţi limbajul fişierelor `bst`. În orice caz, nu modificaţi conţinutul fişierelor `bst` standard! Dacă vreţi să operaţi modificări, faceţi o copie într-un dosar din `localtexmf` şi redenumiţi copia, în aşa fel încât să nu se producă nici un fel de confuzii în sistem.

fişierele bst

Pentru început puteţi investiga fişierele `bst` cele mai obişnuite din dosarul `base` plasat pe calea `\texmf\bibtex\bst`. Repet, nu modificaţi nimic aici! Faceţi doar copia în modul indicat mai sus şi examinaţi copia.

Fişierul cu care este bine să începeţi este `plain.bst`. Căutaţi, de pildă, secvenţa `FUNCTION {book}` şi cercetaţi definiţia modului de afişare a datelor bibliografice pentru o carte.

Limbajul `bst` are o sintaxă aparte. Dacă în LaTeX aţi văzut o structură condiţională de forma „comandă, condiţie, acţiune, acţiune“, în `bst` structura condiţională are forma „condiţie, acţiune, acţiune, if$“. Este ca şi cum aţi scrie „2 2 +“. Există motive temeinice pentru a proceda aşa, dar explicarea lor nu-şi are rostul aici.[70]

Fiecare stil bibliografic are drept efect moduri diferite de sepa-

[68] Căutaţi fişierul `rht.bib` la adresa de <`http://www.csse.monash.edu.au/mirrors/bibliography/Ai/rht.html`>. Dacă nu găsiţi acolo fişierul `rht.bib`, încercaţi o căutare cu ajutorul cuvintelor-cheie.

[69] Am reuşit să încarc chiar bibliografii mai ample de 43193 intrări şi de 64307 intrări, în mai puţin de un minut, respectiv în jur de un minut. Testul l-am făcut cu un fişier obţinut prin concatenarea unei colecţii de fişiere `bib`. În orice caz, nu aş folosi pentru biblioteca de acasă altceva decât fişiere `bib`. Putem, de asemenea, medita şi la ideea că o bibliotecă obişnuită, precum cea a unei facultăţi, are cam 50000 de volume.

[70] Puteţi citi explicaţiile lui Oren Patashnik în `btxhak` din documentaţia

rare a elementelor intrării bibliografice, precum şi unele diferenţe în modul de evidenţiere a lor. Ordinea faptelor privitoare la publicare diferă şi ea. Dacă vă familiarizaţi cu funcţiile din `plain.bst`, veţi descoperi desigur că acesta pune mai întâi editura şi apoi localitatea şi anul. Majoritatea stilurilor bibliografice preferă să indice întâi localitatea şi apoi editura.

Stilurile bibliografice definite în fişierele `bst` sunt replici electronice ale stilurilor tradiţionale. Un element important de diferenţiere îl reprezintă modul în care se realizează conexiunea dintre text şi intrările din bibliografie. Multe stiluri folosesc un cuplu „autor-an". `Plain` foloseşte un număr, numărul intrării bibliografice.

Oren Patashnik argumentează în favoarea stilului `plain` contra sistemului editurii Universităţii din Chicago.[71] Stilul acesta era dificil de folosit pe vremea tiparului tradiţional. Cu BibTeX şi LaTeX situaţia se schimbă. Nu trebuie să avem grija numerelor din bibliografie.

Pentru cartea de faţă am adaptat `plain.bst` la cerinţele limbii române. Am făcut şi unele modificări, în funcţie de practica editurilor din România în domeniul bibliografiilor.[72]

2.2.3.1 Listarea unei bibliografii în LaTeX

Generarea bibliografiei în LaTeX este foarte simplă. În fereastra de dialog cu proprietăţile proiectului TeXnicCenter bifaţi caseta `Uses BibTeX`. În fişierul principal puneţi în punctul în care trebuie să apară bibliografia rânduri de program LaTeX după modelul următor:

```
1 \backmatter
2 \nocite{*}
3 \bibliography{eseu}
4 \bibliographystyle{plain}
```

Comanda de pe rândul 1 spune LaTeX că este vorba despre partea finală a cărţii. Comanda de pe rândul 2 cere afişarea întregului conţinut al fişierului `bib`, indiferent de citarea sau nu a intrărilor în cuprinsul lucrării. Argumentul comenzii de pe linia 3 este numele fişierului `bib`, fără extensie. BibTeX adaugă automat extensia.[73]

MikTeX. Ţineţi însă cont că acesta este un text pentru ucenicii vrăjitori, nu pentru utilizatorii obişnuiţi!

[71] A se vedea documentul `btxdoc` din documentaţia MikTeX.

[72] Nu prezentăm aici fişierul `plainro.bst`. A se vedea pe situl cărţii o prezentare a acestui fişier.

[73] Dacă aş scrie `eseu.bib`, atunci BibTeX ar căuta un fişier `eseu.bib.bib`. Acest fişier nu există şi bibliografia nu poate fi generată. Atenţie şi la faptul că,

Argumentul comenzii de pe rândul 4 este numele fişierului de tip `bst` care conţine programul de prelucrare a bazei de date. Din nou, extensia NU trebuie inclusă în argument.

Recomandarea cât se poate de categorică pentru începători este să folosească mediul integrat TEXnicCenter pentru a genera bibliografia. BIBTEX este un compilator. Apelarea sa în linie de comandă cere o cunoaştere destul de bună a funcţionării sale.

2.2.3.2 Trimiterile la bibliografie în LATEX

După cum am explicat deja,[74] etichetele înregistrărilor din fişierele `bib` sunt folosite ca argumente pentru trimiteri. Comanda pentru trimiteri este `\cite{}`. În argumentul ei este pusă eticheta înregistrării la care vrem să trimitem.

Comanda `\cite{}`, dacă folosim stilul `plain`, generează un număr pus în paranteze drepte. Acesta este numărul intrării din bibliografie la care vrem să trimitem.

Forma `\cite[]{}` a comenzii pentru trimiteri la bibliografie include şi o opţiune. Argumentul opţional recomandat este un număr de pagină, precedat de „p.“, în cazul limbii române. Am putea pune şi un alt argument opţional, cum ar fi un număr de secţiune. Cartea sau articolul la care trimitem trebuie să aibă însă secţiunile numerotate. În cazul documentelor de pe Internet, putem indica un fişier.

Comanda `\cite{}` are diverse versiuni, în funcţie de stilul bibliografic folosit. Aceste versiuni sunt descrise în documentaţia aferentă stilului bibliografic respectiv. Folosirea versiunilor comenzii scade însă portabilitatea documentului de la un stil bibliografic la altul.

2.3 Turnul Babel

Din punctul de vedere al calităţilor tipografice ale textului, departe de a fi o binecuvântare, calculatorul este adesea un adevărat blestem. Cea mai vie ilustrare o constituie modul în care sunt tratate literele româneşti. Nenumărate eseuri studenţeşti sunt scrise fără diacritice. Ghilimelele tradiţionale româneşti tind să dispară. Într-un cuvânt, calculatorul pare a ne întoarce într-o epocă de sălbăticie intelectuală.

Motivele acestei stări de lucruri cred că se reduc, până la urmă,

în LATEX, fără a folosi pachete speciale, numărul de înregistrări din fişierul `bib` nu poate trece de 3000. Nu văd însă de ce un eseu filosofic ar avea nevoie de un fişier atât de mare.

[74] A se vedea mai sus pagina 116, rândul 23.

la ignorarea importanţei programării unui calculator. Consumatorii tind să achiziţioneze „maşini puternice“. Programele sunt însă adesea piratate sau instalate fără o reglare atentă. Efectul este, fără îndoială, dezastruos.

2.3.1 Literele româneşti

Ignorarea importanţei programării începe probabil de la felul în care este percepută tastatura. Simbolurile scrise pe taste au o valoare pur orientativă. Totul poate fi schimbat prin programare. Evident, nu trebuie neapărat să *scrieţi* programele respective. Ajunge să-i spuneţi sistemului de operare ce program vreţi să utilizaţi.

litere româneşti în Windows

Să zicem că folosiţi Windows98, sistemul de operare la care ne referim în mod preponderent în această carte. Daţi un clic pe butonul Start. Urmaţi ruta Settings→ Control Panel→ Keyboard. Fereastra care se deschide vă permite să controlaţi proprietăţile tastaturii. Alegeţi panoul Language şi daţi un clic pe butonul Add.... Selectaţi din lista derulantă Romanian. Daţi un clic pe OK. Apoi daţi un clic pe Apply.

După ce-aţi făcut operaţiile de mai sus sistemul vă va cere, probabil, CD-ul de pe care aţi instalat Windows98. Lucrurile se vor petrece ceva mai lin dacă sistemul este preinstalat. De cele mai multe ori însă, oricum nu este nevoie de CD. Fişierul de care aveţi realmente nevoie este KBDRO.KBD şi se găseşte poate deja în dosarul `c:\windows\system`.

Verificaţi, după aceea, dacă pe *taskbar* va apare indicatorul ce ne spune care este tastatura pe care o folosim. De asemenea, vedeţi care este combinaţia de taste care vă convine cel mai mult atunci când comutaţi de la o tastatură la alta.

Acum aveţi o tastatură care include literele româneşti. Fişierul KBDRO.KBD de pe CD-ul de instalare produce o tastatură pe care eu unul nu o găsesc comodă. Evident, este mai bună decât una fără litere româneşti. Din păcate, tastele Y şi Z îşi inversează în mod inutil rolul. Semnele de punctuaţie sunt greu de găsit. Dacă daţi peste un suflet caritabil în stare să modifice în mod competent fişierul KBDRO.KBD, ar fi bine să aduceţi tastatura la o formă mai comodă.

Vim şi literele româneşti

La urmă, dar nu în cele din urmă în ordinea importanţei, nu uitaţi că trebuie să alegeţi un tip de literă care conţine litere româneşti. Ca exerciţiu, mergeţi în Vim la meniul Edit. De acolo la Select Font... şi testaţi existenţa la rubrica Script a opţiunii Central European; dacă

această opţiune nu există, atunci n-aveţi litere româneşti în cazul tipului respectiv de literă.

Ideea de mai sus este foarte simplă. Este ca şi cum i-am spune unui tipograf că vrem literele româneşti din cutare tip de literă. Nu ajunge să-i spunem acest lucru. El trebuie să şi găsească literele respective prin cutiile sale.

Se mai poate întâmpla ceva neplăcut. Chiar dacă literele există, programul cu care lucrăm să nu interpreteze corect *input*-urile primite. Din păcate, acesta este cazul în mediul integrat TEXnicCenter când este vorba despre fereastra cu mesaje de eroare. Literele româneşti nu sunt afişate corect.[75]

2.3.1.1 Soluţii LaTeX pentru limba română

LaTeX poate pregăti pentru tipar în condiţii foarte bune texte cu litere româneşti. Voi prezenta mai întâi soluţia pe care o folosesc efectiv în această carte şi care mi se pare deosebit de comodă. Apoi voi descrie soluţia clasică, tipică pentru spiritul TeX, soluţie care poate fi folosită pentru a pregăti surse LaTeX pe computere care nu au posibilitatea de a instala tastaturi cu litere româneşti.

2.3.1.1.1 Pachetele LaTeX şi limba română Soluţia folosită la pregătirea pentru tipar a cărţii de faţă poate fi lesne înţeleasă examinând cele trei-patru rânduri din fişierul `sty` care-i spun sistemului LaTeX să folosească limba română:

```
\RequirePackage{type1ec}
\RequirePackage[T1]{fontenc}
\RequirePackage[cp1250]{inputenc}
\RequirePackage[english,romanian]{babel}
%=============================================
\newcommand{\texteng}[1]{\foreignlanguage{english}{#1}}
```

Primul rând cere sistemului să încarce pachetul cm-super. Acest pachet de peste 68MB conţine o versiune a complet scalabilă a tipurilor de litere computer modern. Computer modern sunt literele clasice ale TeX, proiectate de către însuşi Donald Knuth. Ele nu sunt disponibile decât pentru un număr limitat de dimensiuni şi rezistă mai prost la mărire. Pachetul cm-super este disponibil graţie lui Vladimir Volovich.

[75] N-am reuşit să localizez problema, dar bănuiesc că totul pleacă de la problemele generate de ideea de integrare ca atare. Programul trebuie să captureze fluxul mesajelor compilatorului, care este apelat însă în linia de comandă.

Nu este absolut necesar să încărcaţi cm-super. Puteţi folosi literele standard în LaTeX sau puteţi încărca pachetul `times` pentru tipul de litere Times New Roman.

Al doilea rând este însă necesar pentru a avea acces la literele româneşti.[76] Comanda pe care o conţine ar putea fi asemuită cu indicarea pe vremuri a cutiilor cu litere de plumb. Necesară pentru limba română este doar opţiunea `T1`.

Rândul al treilea conţine o comandă care-i spune sistemului în ce „limbă“ îi dăm ordinele privitoare la litere. În esenţă, opţiunea `cp1250` îi spune sistemului că noi folosim codificarea literelor din Windows.

Rândul al patrulea cere LaTeX să încarce pachetul `babel`. Acest pachet, creat de Johannes L. Braams, permite LaTeX să despartă corect în silabe în limba română, să scrie o serie de cuvinte în limba română. Căutaţi fişierul `romanian.ldf`; citiţi şi modificaţi (după ce v-aţi făcut o copie de siguranţă) unele elemente din acest fişier: eu unul prefer termenul „indice“ (pentru englezescul *index*); „tabelul“ şi nu „tabela“ şi „glosar“ pentru englezescul *glossary*.[77]

Dacă nu funcţionează ceva cum trebuie, citiţi documentaţia. Nu faceţi însă mai multe schimbări deodată. Faceţi o singură schimbare şi studiaţi efectele, pentru a detecta cauza fenomenului care vă deranjează.

Ultimul rând de cod listat mai sus din fişierul `sty` nu face decât să definească o comandă pentru delimitarea porţiunii de text care este în engleză. Atunci când încărcăm pachetul `babel` ultima opţiune este cea care se aplică automat.[78] Dacă vrem să folosim, într-o porţiune de text (un citat, de pildă), o altă limbă, trebuie să anunţăm în mod explicit acest lucru. Altfel, s-ar putea să avem diverse surprize neplăcute (la despărţirea în silabe, de exemplu).

2.3.1.1.2 Comenzi LaTeX pentru diacritice LaTeX are propriul său mod de a construi litere cu diacritice.[79] Există o serie de comenzi care ne permit să plasăm diacritice pe litere. Ca deobicei

[76] Programul de vizualizare a fişierului `dvi` n-ar afişa literele româneşti. Astfel se rupe ciclul editare-compilare-vizualizare.

[77] Nu uitaţi să spuneţi sistemului MikTeX că folosiţi limba română. A se vedea indicaţiile de instalare a MikTeX la pagina 70, rândul 5.

[78] Acelaşi lucru este valabil şi-n cazul pachetului `fontenc`!

[79] Pentru a folosi aceste comenzi n-aveţi nevoie de pachete speciale. `Babel` continuă totuşi să fie absolut recomandabil: fără acest pachet, ar trebui să indicaţi manual despărţirile corecte în silabe.

este mai simplu să recurgem la exemple. În cazul limbii române comenzile sunt:

```
\u{a} \u{A}\\
\^{a} \^{A}\\
\^{\i} \^{I}\\
\c{s} \c{S}\\
\c{t} \c{T}
```

ă Ă
â Â
î Î
ş Ş
ţ Ţ

Ca şi-n cazul altor comenzi din LaTeX, numele comenzii, chiar dacă se rezumă aici doar la un semn, este sugestiv.

2.3.2 Vim şi adaptarea tastaturii

Dacă vreţi să folosiţi comenzile specifice LaTeX, crearea unui meniu Vim pentru literele româneşti cu diacritice nu este o soluţie foarte practică. Există, din fericire, o alternativă. Schimbarea rolului unor taste.

Dacă fişierul KBDRO.KBD este criptic, fişierul de tip `vim` care face acelaşi lucru este relativ uşor de construit. În orice caz, soluţia pe care o prezentăm în continuare este lesne de realizat de către oricine:

```
:imap <F2>   \u{a}
:imap <S-F2> \u{A}
:imap <F3>   \^{a}
:imap <S-F3> \^{A}
:imap <F4>   \^{\i}
:imap <S-F4> \^{I}
:imap <F7>   \c{s}
:imap <S-F7> \c{S}
:imap <F8>   \c{t}
:imap <S-F8> \c{T}
```

Să zicem c-aţi scris un fişier `taste.vim` cu conţinutul de mai sus. Experimentaţi o vreme într-un dosar special creat pentru teste. Creaţi un fişier. În modul normal daţi comanda `:so taste.vim`.[80] Acum tastele funcţionale au un rol!

Nu este nevoie de prea multe explicaţii pentru ceea ce aţi obţinut. În scriptul vim, cele două puncte sunt desigur un semn că este vorba despre o comandă care poate fi dată şi-n linie. Puteţi să le şi eliminaţi. Cheia comenzilor este `imap`, care-i spune lui Vim să schimbe rolul unei taste în modul insert. `<F2>` se referă la cea de-a doua tastă funcţională.[81] După numele tastei, este indicat noul ei rol.

[80] Alternativ, puteţi folosi `Run a Vim Script` de pe bara cu instrumente.

[81] Nu schimbaţi rolul tastei F1; ea serveşte la apelarea `help`-ului din Vim.

Eu recomand schimbarea în felul arătat mai sus a rolului tastelor funcţionale. La urma urmei, pentru asta sunt disponibile pe tastatură: pentru a putea reprograma rolul lor.

Vă sâcâie însă inversarea rolului tastelor Y şi Z din tastatura românească a Windows? n-aţi găsit pe nimeni care ştie cum să modifice KBDRO.RO? În Vim, puteţi schimba rolul celor două taste. Studiaţi însă atentă sau atent următoarele două rânduri!

```
inoremap z  y
inoremap y  z
```

Dacă folosiţi `imap` simplu, vă învârtiţi în cerc! Vim va semnala o eroare. Comanda `inoremap` îi spune să nu *reschimbe* rolul tastei.

Dacă vă convine ideea, puteţi continua experimentul cu modificarea rolului combinaţiei SHIFT+Y, respectiv SHIFT+Z.

```
inoremap <S-z>  Y
inoremap <S-y>  Z
```

Dacă vreţi să permanentizaţi soluţii de genul celei de mai sus, nu uitaţi să plasaţi `taste.vim` în dosarul `plugin` din `vimfiles`. Verificaţi dacă nu se produc eventuale conflicte cu alte scripturi.

2.3.2.1 De la codurile Windows la comenzile LaTeX şi înapoi

Dacă studiaţi codurile unui fişier cu litere româneşti cu diacritice, veţi vedea că literele minuscule cu diacritice au codurile hex următoare: *e3 e2 ee ba fe*. Traduse în baza 10, aceste coduri sunt: *227 226 238 186 254*. Cum foloseşte Windows aceste coduri? Gândiţi-vă că este asemenea cuiva care ar avea un text criptat în faţă şi ar căuta într-o carte în care pe fiecare pagină sunt rubrici de la 0 la 255; la fiecare rubrică găseşte o dezlegare a codului care-l interesează.

2.3.2.1.1 Intermezzo pentru ucenicii vrăjitori Aţi văzut probabil tot felul de programe care vă ajută să exploraţi casetele cu litere. Puteţi descifra secretele care stau în spatele casetelor cu litere din Windows şi folosind Vim.

Folosind posibilitatea de a trece de la vizualizarea textului la cea a codurilor hex nu vă va fi greu să construiţi cu mâna un fişier care arată astfel în hex (în părţile sale semnificative):

```
00: 0102 0304 0506 0708 090d 0a0b 0c0d 0e0f
10: 1011 1213 1415 1617 1819 1a1b 1c1d 1e1f
20: 2021 2223 2425 2627 2829 2a2b 2c2d 2e2f
30: 3031 3233 3435 3637 3839 3a3b 3c3d 3e3f
40: 4041 4243 4445 4647 4849 4a4b 4c4d 4e4f
50: 5051 5253 5455 5657 5859 5a5b 5c5d 5e5f
60: 6061 6263 6465 6667 6869 6a6b 6c6d 6e6f
70: 7071 7273 7475 7677 7879 7a7b 7c7d 7e7f
80: 8081 8283 8485 8687 8889 8a8b 8c8d 8e8f
90: 9091 9293 9495 9697 9899 9a9b 9c9d 9e9f
a0: a0a1 a2a3 a4a5 a6a7 a8a9 aaab acad aeaf
b0: b0b1 b2b3 b4b5 b6b7 b8b9 babb bcbd bebf
c0: c0c1 c2c3 c4c5 c6c7 c8c9 cacb cccd cecf
d0: d0d1 d2d3 d4d5 d6d7 d8d9 dadb dcdd dedf
e0: e0e1 e2e3 e4e5 e6e7 e8e9 eaeb eced eeef
f0: f0f1 f2f3 f4f5 f6f7 f8f9 fafb fcfd feff
00: 0000 0d0a
```

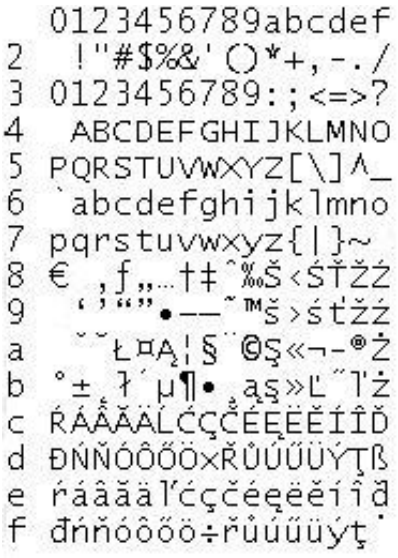
Reveniţi apoi la vizualizarea în mod text. Aranjaţi simbolurile pe care le folosiţi în texte în aşa fel încât să fie uşor de înţeles codul hex din spatele lor. Eliminaţi din fişier primele 32 de coduri (dacă vreţi să cercetaţi simbolurile ca atare).

```
În mediul verbatim semnele din
sursă pot fi reproduse ca atare.
Iată semnele de bază:

        \begin{verbatim}
  0 1 2 3 4 5 6 7 8 9 a b c d e f
2   ! " # $ % & ’ ( ) * + , - . /
3 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
4 @ A B C D E F G H I J K L M N O
5 P Q R S T U V W X Y Z [ \ ] ^ _
6 ‘ a b c d e f g h i j k l m n o
7 p q r s t u v w x y z { | } ~
        \end{verbatim}

Restul semnelor nu pot fi
reproduse în mediul verbatim.

Vedeţi în imaginea alăturată ce
se întâmplă pe ecranul Vim.
```



Dacă veţi combina cifra de pe prima coloană a rândului pe care se află litera cu cifra de primul rând al coloanei pe care se află simbolul care vă interesează, obţineţi codul hex corespunzător.[82]

Există însă o metodă mai simplă de a obţine codul literei. Treceţi editorul Vim în modul normal. Daţi apoi comanda `ga`; pe ultima line

[82] Atenţie! În hex, adică în baza 16, *a*, *b*, *c*, *d*, *e*, *f* sunt cifre, care se traduc în baza 10 prin *10*, *11*, *12*, *13*, *14*, respectiv prin *15*. Ceea ce vedeţi în figură pe prima coloană şi pe primul rând sunt cifre, nu litere!

a ecranului Vim vor apărea numele literei, codul zecimal al literei, codul în hex şi-n octal.

Cu fişierul astfel creat puteţi face investigaţii. Modificaţi versiunea tipului de literă şi veţi vedea alte semne pe ecran. Puneţi, de pildă, versiunea pentru greacă a tipului de literă şi observaţi unde apar literele greceşti. Schimbaţi tipul de literă şi studiaţi posibilităţile pe care vi le oferă.

2.3.2.1.2 Schimbări de care s-ar putea să aibă nevoie şi persoanele obişnuite Dacă sunteţi o utilizatoare sau un utilizator obişnuit, v-aţi mulţumit doar să citiţi în diagonală secţiunea 2.3.2.1.1. Dacă aţi sărit peste ea, aruncaţi totuşi o privire. Vă veţi întreba poate de ce am folosit o imagine şi nu un text pentru a ilustra felul în care Vim afişează literele limbilor Europei Centrale. Motivul rezidă în modul în care LaTeX prelucrează *input*-ul pe care l-am produs cu ajutorul Vim.

Semnele de bază de care se foloseşte TeX sunt între codurile 32 şi 126, după cum se vede din tabelul de la pagina 127, rândul 5. Codul hex 20 corespunde codului zecimal 32. Este codul pentru un spaţiu alb. Codul 127 generează <~>.

Pachetul LaTeX `inputenc` traduce, ca să spun aşa, limbajul Windows în cel al TeX. Aruncaţi o privire în fişierul `cp1250.def` şi veţi vedea definiţiile folosite în procesul de traducere. Observaţi modul în care toate comenzile TeX sunt scrise numai cu semnele de bază.

Ca şi-n Vim, drept nume pentru simboluri voi folosi simbolul pus în paranteze unghiulare. Cu alte cuvinte, <~> este acelaşi lucru cu „tildă“. Codurile 127, 128 şi 129 nu au o definiţie în `cp1250`. Dacă vreţi semnul pentru euro, folosiţi comanda `\EUR{}` din pachetul `marvosym`. Există şi variante ale acestei comenzi. Codul 130 este tradus prin `\quotesinglbase{}` şi generează <‚>.

Codul 131 n-are o definiţie în `cp1250`. Codul 132 este tradus prin `\quotedblbase{}` şi generează <„>.

Dăm în continuare restul listei codurilor care nu ridică probleme deosebite:

133<…>	134<†>	135<‡>	137<‰>	138<Š>
139<‹>	140<Ś>	141<Ť>	142<Ž>	143<Ź>
145<‘>	146<’>	147<“>	148<”>	149<•>
150<–>	151<—>	153<™>	154<š>	155<›>
156<ś>	157<ť>	158<ž>	159<ź>	160< >
161<ˇ>	162<˘>	163<Ł>	165<Ą>	167<§>
168<¨>	169<©>	170<Ş>	171<«>	174<®>

175<Ż>	176<°>	178<¸>	179<ł>	180<´>
182<¶>	183<·>	184<¸>	185<ą>	186<ş>
187<»>	188<Ľ>	189<˝>	190<ľ>	191<ż>
192<Ŕ>	193<Á>	194<Â>	195<Ă>	196<Ä>
197<Ĺ>	198<Ć>	199<Ç>	200<Č>	201<É>
202<Ę>	203<Ë>	204<Ě>	205<Í>	206<Î>
207<Ď>	208<Đ>	209<Ń>	210<Ň>	211<Ó>
212<Ô>	213<Ő>	214<Ö>	216<Ř>	217<Ů>
218<Ú>	219<Ű>	220<Ü>	221<Ý>	222<Ţ>
223<ß>	224<ŕ>	225<á>	226<â>	227<ă>
228<ä>	229<ĺ>	230<ć>	231<ç>	232<č>
233<é>	234<ę>	235<ë>	236<ě>	237<í>
238<î>	239<ď>	240<đ>	241<ń>	242<ň>
243<ó>	244<ô>	245<ő>	246<ö>	248<ř>
249<ů>	250<ú>	251<ű>	252<ü>	253<ý>
254<ţ>	255<˙>			

Codul 160 este tradus de `cp1250` prin `\nobreakspace{}`. Este normal ca-n tabel să apară doar un simplu spaţiu alb. Uzual, în sursa LaTeX, punem mai degrabă o tildă pentru a indica spaţiul insecabil.

Codul 173 este tradus prin `\-{}` care este comanda prin care i se spune sistemului LaTeX unde se *poate* face o despărţire în silabe. Nu l-am inclus în tabel pentru că efectul său este invizibil dacă nu se face efectiv o despărţire în silabe în punctul în care apare. De altfel, nici nu mi se pare o idee bună să folosim acest semn în sursa LaTeX; ar fi imposibil să-l deosebim de liniuţa de unire.

Codul 136 are efect în Vim, dar este nedefinit în `cp1250`. În schimb, codurile 127 şi 144 nu au efect în Vim şi sunt nedefinite şi-n `cp1250`. Folosirea lor generează o eroare.

Codurile următoare au efect în LaTeX, dar trebuie folosite în modul matematic, nu în mod text:

172<¬> 177<±> 181<μ> 215<×> 247<÷>

Cazurile cele mai interesante sunt cele ale codurilor 164 şi 166. Ele sunt definite în `cp1250`. LaTeX nu generează un mesaj de eroare din perspectiva tastaturii, a *input*-ului. Apare însă un mesaj de eroare care ne spune că nu există în `T1` simbolurile corespunzătoare. Este ca şi cum turnătoria de litere de plumb ne-ar spune că n-are mulajele necesare pentru a turna litere de forma cerută.

Nu vă bucuraţi prea mult de listele cu coduri de mai sus. Ele sunt bune în cazul Windows. În lumea largă, veţi întâlni o junglă de

astfel de coduri. Cu adevărat importantă este înţelegerea procesului în mai multe trepte prin care o apăsare pe o tastă se transformă într-un cod numeric, apoi într-o comandă LaTeX şi se termină prin găsirea unui simbol care este pus pe foaia de hârtie.

N-am putea opera şi noi asemenea transformări? Desigur că da. În interiorul LaTeX ele au o utilitate mai restrânsă, în măsura în care putem folosi pachete gata pregătite. Transformările sunt cu adevărat utile în afara LaTeX.

Putem folosi Vim pentru a opera o trecere de la codificarea Windows la cea a LaTeX. Voi ilustra acest lucru în cazul literelor româneşti. Generalizarea nu este dificil de realizat. Trebuie doar create nişte fişiere de tip `vim` care conţin scripturile pentru realizarea substituţiilor. Iată cum facem trecerea de la coduri Windows la comenzi LaTeX:

```
%s/ă/\\u{a}/ge | update
%s/Ă/\\u{A}/ge | update
%s/â/\\^{a}/ge | update
%s/Â/\\^{A}/ge | update
%s/î/\\^{\\i}/ge | update
%s/Î/\\^{I}/ge | update
%s/ş/\\c{s}/ge | update
%s/Ş/\\c{S}/ge | update
%s/ţ/\\c{t}/ge | update
%s/Ţ/\\c{T}/ge | update
```

Trecerea inversă este realizată în felul următor:

```
%s/\\u{a}/ă/ge | update
%s/\\u{A}/Ă/ge | update
%s/\\^{a}/â/ge | update
%s/\\^{A}/Â/ge | update
%s/\\^{\\i}/î/ge | update
%s/\\^{I}/Î/ge | update
%s/\\c{s}/ş/ge | update
%s/\\c{S}/Ş/ge | update
%s/\\c{t}/ţ/ge | update
%s/\\c{T}/Ţ/ge | update
```

Acestea sunt scripturi Vim care trebuie chemate cu comanda `:so` urmată de numele scriptului în fişierul pe care vrem să-l transformăm. Alternativ, putem folosi `Run Vim Script` de pe bara cu instrumente. Este inutil, de asemenea, să adăugăm că trebuie să „vă faceţi mâna“ pe fişiere de probă.

2.3.2.1.2.1 Vrăjitorii cu tastatura Tasta Q este, cel puţin din punctul meu de vedere, teribil de bine plasată pe tastatură şi mi se întâmplă, în română, să o folosesc extrem de rar. În schimb, trebuie uneori să tastez de multe ori &, care mi se pare incomod plasat. Ca să rezolv situaţia dau, în modul normal al Vim, comanda `:imap q &` şi tasta Q îşi schimbă ca prin minune rolul.

Cum scap însă de schimbarea de mai sus, fără să închid fişierul? Dau comanda `:iunmap q` şi tasta Q revine la vechiul rol.

Puteţi proiecta un script pentru a vrăji tastatura. Exemplul dat aici încearcă să arate cum pot fi corijate unele neajunsuri din tastatura românească standard în Windows. Schimbările sunt operabile numai în Vim!

Ideea de bază este cea de a ataşa o altă codificare tastei. Următoarele comenzi vă permit să tastaţi pe Ă în loc de { şi pe ă în loc de [.

```
:map! {  <char-195>
:map! [  <char-227>
```

Puteţi scrie şi ă în loc de <char-227> sau puteţi folosi notaţia în hex <char-0xe3>. Cifrele în hex trebuie precedate de 0x.

Dacă ştim codurile pe care vrem să le introducem de la tastatură, nu este greu să schimbăm rolul tastelor.

Putem modifica şi rolul unei combinaţii de taste:

```
:map! /a ă
```

Trebuie să bateţi însă suficient de repede tastele una după alta pentru a obţine efectul dorit.

Putem crea şi scripturi cu astfel de modificări ale tastelor. Atenţie însă la asigurarea unui gen sau altul de mecanism de scăpare. Puneţi, de exemplu, în script ceva de genul:

```
:map! <F5> <Esc>:unmap! [<CR>
```

tastatură românească pentru Vim

Vim are posibilitatea de a crea tastaturi cu ajutorul unor scripturi speciale. Doar cu titlu de exemplu, putem presupune c-am creat un fişier `romana.vim` şi l-am plasat în dosarul `keymap` din `vimfiles`. Conţinutul său este următorul:

```
let encoding='cp1250'
let b:keymap_name = "ro"
```

```
loadkeymap
" Taste modificate în jurul lui Enter:
[ ă
{ Ă
] î
} Î
\\ â
<Bar> Â
; ş
: Ş
’ ţ
\" Ţ
" Recuperarea tastelor pierdute (folosind combinaţii):
/. :
/, ;
/q {
/w }
/z \\
```

Încărcaţi tastatura cu comanda `:set keymap=romana` sau folosind Keymap din meniul Edit. Puteţi comuta între tastatura românească şi cea standard cu Ctrl+^.

Tastatura astfel definită este mult mai flexibilă decât cea din Windows. Poziţia literelor pe taste poate fi modificată uşor. Semnele sacrificate prin plasarea de litere româneşti pot fi regăsite prin combinaţii de taste. Aceste combinaţii pot fi adaptate la gusturile fiecărei persoane care utilizează Vim.

2.3.3 Limbile europene care folosesc alfabetul latin

Să vedem acum câteva exemple de nume, titluri şi expresii care pot fi întâlnite într-un text filosofic. Începem nume, titluri sau expresii de sorginte franceză sau germană:

```
Br\’{e}hier\\
L’\^{E}tre et le n\’{e}ant\\
La science et l’hypoth\‘{e}se\\
G\"{o}del\\
f\"{u}r sich
```

Bréhier
L’Être et le néant
La science et l’hypothèse
Gödel
für sich

Italiana, spaniola sau portugheza pun şi ele probleme dacă vrem să reproducem corect un titlu, un nume sau o expresie:

```
0,1 e 2 e cos\‘{\i} via\\
Espa\~{n}a inteligible\\
Inicia\c{c}\~{a}o ao filosofar
```

0,1 e 2 e così via
España inteligible
Iniciação ao filosofar

S-ar putea ca mai rar şi mai puţină lume să se ciocnească de problemele ridicate de limbile slave care folosesc alfabetul latin. Li-

teratura poloneză de logică şi de filosofie este totuşi foarte cunoscută şi voi alege de acolo câteva exemple:

```
semiotyka j\k{e}zyk\'{o}w
naturalnych\\
metaj\k{e}zyk\\
Tadeusz Cze\.{z}owski\\
Izydora D\k{a}mbska\\
mo\.{z}liwo\'{s}\'{c}
```

semiotyka języków naturalnych
metajęzyk
Tadeusz Czeżowski
Izydora Dąmbska
możliwość

Maghiara ridică şi ea probleme specifice atunci. Fonemele ö şi ü au şi variante lungi:

```
az igazh\'{\i}v\H{o}k\\
m\H{u}velet
```

az igazhívők
művelet

Unele limbi europene au simboluri speciale, pe lângă cele din alfabetul latin:

```
\L{}ukasiewicz\\
Bewu\ss{}tsein\\
S\o{}ren Kierkegaard\\
\AA{}kvist\\
filozof ta\c{s}\i{}\\
  \textquestiondown{}
  humanizaci\'{o}n
  o deshumanizaci\'{o}n?\\
\textexclamdown{}\ldots{}!
```

Łukasiewicz
Bewußtsein
Søren Kierkegaard
Åkvist
filozof taşı
¿humanización o deshumanización?
¡...!

Spaniola împarantezează întrebările şi propoziţiile exclamative. Dacă ne gândim bine, procedura este cât se poate de raţională; atâta doar că avem nevoie de două simboluri speciale.

Comenzile LaTeX îngropate, ca să spun aşa, în cuvinte sunt notate în versiunea pe care o prefer eu. Cred că sursa este mai lizibilă dacă punem sistematic argumentele unei comenzi în acolade şi, în cazul în care nu există un argument, punem doar acolade.[83] Consultaţi documentaţia pentru a vedea şi alte posibilităţi de a nota comenzile respective.[84]

Modul acesta de a cere prin comenzi speciale simboluri este util în special atunci când scriem nume străine, expresii din alte limbi sau texte foarte scurte. n-are rost atunci să punem în mişcare toată maşinăria descrisă în §2.3.1.1.1 pentru a pune un accent pe o literă.

[83] Atenţie la rolul acoladelor chiar şi atunci când nu conţin nimic! Fără ele, LaTeX nu ştie unde se termină comanda. Alternativa este un spaţiu alb, dar acest spaţiu mi se pare derutant pentru ochiul uman. Am mai putea pune în acolade întreaga comandă.

[84] Vedeţi în special Scott Pakin *The Comprehensive LaTeX Symbol List* în dosarul `texmf\doc\latex\comprehensive` din documentaţia MikTeX.

2.3.4 Tehnica alegerii tipului de literă

Până acum am pus un mare accent pe modul în care *vedem* ceea ce introducem de la tastatură. Am descoperit că literele scrise pe tastatura fizică sunt de fapt nişte nume pentru tastele fizice. Ceea ce introducem în realitate de la tastatură sunt nişte coduri numerice. Putem vizualiza în mod comod aceste coduri pe ecranul computerului.

Cum ideea fundamentală aici este aceea de a separa diversele procese pentru a le înţelege şi dirija mai bine, n-are rost să afişăm pe ecran rezultatul final (*output*)-ul, ca-n sistemul WYSIWYG. Să vedem însă cum putem controla aspectul final al literelor.

2.3.4.1 Modificarea aspectului *output*-ului

Pentru unii, cărţile în limba germană scrise cu „caractere gotice“ sunt ceva misterios, texte pe care nimeni nu pare a le putea citi. Pentru alţii, este vorba de un adevărat obiect de cult. LaTeX poate crea tipărituri pentru multe gusturi, după cum se vede din exemplele următoare:

```
{%start grup local
\fontfamily{yfrak}
\fontsize{19}{22}\selectfont
Eseul filosofic
\par}%încheiere de grup cu \par
{
\fontfamily{phv}\fontseries{b}
\fontsize{14}{17}
\selectfont\scshape
Eseul filosofic
\par}
{
\fontfamily{ptm}
\fontsize{17}{21}\selectfont
Eseul filosofic
\par}
{
\fontfamily{pzc}
\fontsize{17}{20}\selectfont
Eseul filosofic
\par}
{
\fontfamily{pcr}
\fontsize{16}{19}\selectfont
Eseul filosofic
\par}
```

Eseul filosofic

ESEUL FILOSOFIC

Eseul filosofic

Eseul filosofic

Eseul filosofic

```
{
\fontfamily{cmss}
\fontsize{17}{22}\selectfont
Eseul filosofic
\par}
{
\fontfamily{cmr}
\fontsize{17}{21}\selectfont
Eseul filosofic
\par}
```

Eseul filosofic

Eseul filosofic

Ar fi multe comentarii de făcut în legătură cu tipurile de litere exemplificate mai sus. Numele lor în TEX reprezintă o prescurtare. Ultimul exemplu este cel al `cmr`, tipul de literă creat de către Donald Knuth şi destinat în mod special sistemului TEX. Numele propriu-zis al acestui tip de literă este „computer modern“. Varianta sa fără serife este ilustrată de penultimul exemplu.

Puteţi compara tipul de litere `cmr` cu clasicul tip *times*, ilustrat de către cel de al treilea exemplu. Consultaţi documentaţia MikTEX pentru a afla mai multe despre diversele tipuri de litere.

Comanda `\fontsize{}{}` are două argumente. Primul argument este cel care indică dimensiunea literei. Al doilea reglează poziţia literei în cutia TEX.

2.3.4.2 Alfabetul chirilic

Experimentul este cea mai bună metodă de a învăţa. Cu condiţia să nu lucraţi cu fişiere pe care n-aţi vrea să le distrugeţi. Schimbaţi opţiunea deci opţiunea pachetului `fontenc` în `OT2`. Să vedem ce s-ar întâmpla în acest caz cu semnele folosite în mod clasic de către TEX (vezi pagina pagina 127, rândul 5):

```
  ! "     % ´ ’ ( ) *   , - . /
0 1 2 3 4 5 6 7 8 9 : ; «   » ?
  А Б Ц Д Е Ф Г Х И Й К Л М Н О
П Ч Р С Т У В Щ Ш Ы З [  ]
‘ а б ц д е ф г х и й к л м н о
п ч р с т у в щ ш ы з  |
```

S-ar părea că a avut loc un accident. Aveţi în sursă aceleaşi coduri, dar rezultatul final apare cu litere chirilice. Nu a avut loc nimic neobişnuit. Codificarea finală este cea care s-a schimbat.

Dacă în fişierul `sty` facem câteva modificări putem insera în textul cu litere latineşti text cu litere chirile:

```
1 \RequirePackage[OT2,T1]{fontenc}
```

```
%.............................
\newcommand{\textrus}[1]{\foreignlanguage{russian}{#1}}
\newcommand{\dur}{\cyrhrdsn}
\newcommand{\Dur}{\CYRHRDSN}
\newcommand{\moale}{\cyrsftsn}
\newcommand{\Moale}{\CYRSFTSN}
\newcommand{\erus}{\cyrerev}
\newcommand{\Erus}{\CYREREV}
```

Sistemul este util, de exemplu, pentru a scrie titluri cu chirilice. Iată codul pentru titlul cărţii lui Dubovik, Veize şi Golovko:

```
\textrus{Referirovanie i
annotirovanie
{\erus}konomicheskih tekstov
na angli\u{i}skom yazyke}
```

Реферирование и аннотирование экономических текстов на английском языке

Soluţia aceasta este ideală atunci când textul nu este preponderent în limba rusă sau nu dispunem de cunoştinţele necesare pentru a modifica tastatura. Secretul ei este că o comandă precum:

```
\textrus{
abvgde\"{e}zhzi\U{i}klmnoprstufkhtschshshch
\dur{}y\moale{}\erus{}yuya}
```

are drept rezultat la ieşire, pe foaia de hârtie (fizică sau electronică) şirul: абвгдеёжзийклмнопрстуфхцчшщъыьэюя de semne ale alfabetului rus.[85]

Dacă majoritatea textului este cu chirilice pare mai raţional să modificaţi codificarea *input*-lui. În acest caz, puteţi folosi opţiunea `cp1251` pentru `inputenc` şi o opţiune de genul `T2A` pentru `fontenc`. Puteţi avea în continuare acces la literele româneşti cu ajutorul comenzilor TEX pure. De asemenea, trebuie să modificaţi tastatura Vim în aşa fel încât să puteţi introduce lesne literele chirilice.[86]

2.3.5 Limba greacă veche

Ar fi inutil să explic aici importanţa limbii greceşti vechi pentru filosofie. Nu puţine sunt eseurile filosofice în care apar termeni în greaca veche sau citate în greaca veche.

Pot spune din proprie experienţă că transliterarea nu este o soluţie fericită. În dicţionarul termenilor filosofici greceşti,[87] un cuvânt

[85] Pentru detalii citiţi documentaţia din dosarul `\latex\cyrillic` din documentaţia MikTEX.

[86] Detalii şi alte exemple vor fi afişate treptat pe situl cărţii.

[87] Francis E. Peters, *Termenii filozofiei greceşti* (Bucureşti: Humanitas, 1993), p.92.

precum ἐπιστήμη apare transliterat ca **epistḗmē**. Spiritul textului grecesc vechi este pierdut la propriu şi la figurat. Mai rău, εἶδος devine **eídos**, ceea ce este de-a dreptul greşit.[88]

Există o transliterare precisă a textelor greceşti vechi. Ea se numeşte „betacod“. Transliterarea aceasta n-are un aspect estetic prea plăcut, dar este foarte eficientă în rolul de *input* într-o sursă LaTeX.

Pentru a folosi betacodul trebuie să încărcaţi pachetul `betacode` punând în locul cuvântului `babel` cuvântul `betababel`. Pachetul `betababel` este o extensie a pachetului `babel` creată de către Felix G. Berkemeier.

Voi explica principiile de bază ale betacodului cu ajutorul unor termeni filosofici foarte cunoscuţi:[89]

```
\bcode{A)DIA/FORON}\\
\bcode{AU)TA/RKEIA}\\
\bcode{AU)TARXEI/A}
```

ἀδιάφορον
αὐτάρκεια
αὐταρχεία

Betacodul nu deosebeşte între majuscule şi minuscule. Există obiceiul de a folosi masiv majusculele englezeşti. Transliterarea literelor ca atare este evidentă. De altfel, vom da exemple pentru fiecare literă din alfabetul clasic grecesc.

Spiritul lin este transliterat prin paranteza rotundă închisă. Spiritul aspru este transliterat prin paranteza rotundă deschisă. Accentul ascuţit este transliterat printr-o bară oblică. Accentul grav este transliterat printr-o bară oblică inversă. Comanda `\bcode{}` inserează textul în punctul în care este dată.

Fiţi atente şi atenţi la diferenţa dintre ultimele două cuvinte de mai sus. Primul înseamnă *independenţă absolută* (autosuficienţă), iar al doilea *putere absolută*. Nu traduceţi al doilea termen de mai sus printr-o transliterare aproximativă a celui de-al treilea termen.[90]

```
\bcode{BOU/LHSIS}\\
\bcode{GE/NESIS}\\
\bcode{DIA/NOIA}\\
\bcode{EI)=DOS}
```

βούλησις
γένεσις
διάνοια
εἶδος

Pachetul `betababel` deosebeşte automat între sigma intern şi fi-

[88] În toată cartea se foloseşte doar un singur tip de accent, iar în glosarul român-elin de la sfârşit diacriticele greceşti dispar cu totul! Pentru forma corectă a lui εἶδος şi explicarea poziţiei spiritului a se vedea Ana Felicia Ştef, *Manual de greacă veche*(Bucureşti: Humanitas, 1996), p.23.

[89] Extraşi tot din Peters, *op.cit.*, dar transliteraţi în betacod şi prelucraţi în LaTeX.

[90] Din păcate, ediţia 1993 din Peters, *op.cit.* conţine exact această greşeală (v.p.52).

nal. Dacă avem nevoie, putem însă scrie `\bcode{s1}` pentru a obţine σ şi `\bcode{s2}` pentru a obţine ς.

```
\bcode{ZW=|ON}
```

ζῷον

Accentul circumflex este notat prin semnul egalităţii. Iota subscris este notat printr-o bară verticală.

```
\bcode{H(DONH/}\\
\bcode{QEWRI/A}\\
\bcode{I)SONOMI/A}\\
\bcode{KO/SMOS}\\
\bcode{LO/GOS}\\
\bcode{TA\ MAQHMATIKA/}
```

ἡδονή
θεωρία
ἰσονομία
κόσμος
λόγος
τὰ μαθηματικά

```
\bcode{NOU=S}\\
\bcode{CENO/S}\\
\bcode{O)/NOMA}\\
\bcode{PAQO/S}\\
\bcode{R(OH/}\\
\bcode{SWFROSU/NH}\\
\bcode{TE/LOS}\\
\bcode{U(PO/QESIS}\\
\bcode{FRO/NHSIS}\\
\bcode{XRO/NOS}\\
\bcode{YUXH/}\\
\bcode{W(/RA}
```

νοῦς
ξενός
ὄνομα
παθός
ῥοή
σωφροσύνη
τέλος
ὑπόθεσις
φρόνησις
χρόνος
ψυχή
ὥρα

Betacodul este, în general, transparent. Unele transliterări pot fi mai dificil de descifrat. Cred c-ar fi mai greu, la prima vedere, să vă daţi seama, văzând numai betacodul, că penultimul cuvânt este familiarul termen grecesc pentru minte sau suflet.

În cazul literelor majuscule, regula betacodului este ca semnele diacritice să fie puse înaintea literei. Pachetul `betababel` este flexibil şi admite punerea diacriticelor şi după literă, cu condiţia să fim consecvenţi. Voi testa posibilităţile pachetului cu câteva nume proprii.[91] Voi devia, de asemenea, de la practica folosirii exclusiv a majusculelor.

```
\bcode{*)Aristote/lhs}\\
\bcode{*Dioge/nhs *Lae/rtios}\\
\bcode{*E)mpedoklh=s}\\
\bcode{*Pla/twn}
```

Ἀριστοτέλης
Διογένης Λαέρτιος
Ἐμπεδοκλῆς
Πλάτων

Pentru porţiunile ample de text, pachetul `betababel` oferă un

[91] Numele sunt extrase din manualul Anei Ştef, *op.cit.*

mediu LaTeX special. Voi ilustra utilizarea acestui mediu cu un citat din Aristotel.[92]

συλλογισμὸς δέ ἐστι λόγος ἐν ᾧ τεθέντων τινῶν ἕτερόν τι τῶν κειμένων ἐξ ἀνάγκης συμβαίνει τῷ ταῦτα εἶναι.

Sursa în LaTeX a citatului este următoarea:

```
\begin{betacode}
sullogismo\s de/ e)sti lo/gos e)n w(=| teqe/ntwn tinw=n e(/tero/n
ti tw=n keime/nwn e)c a)na/gkhs sumbai/nei tw=| tau=ta ei)=nai.
\end{betacode}
```

2.3.5.1 Pachetul `babel` şi limba greacă

Betacodul este un mod comod şi precis de a introduce text în limba greacă veche. După cum spuneam, el nu este însă decât un etaj ridicat peste clădirea mai vastă a pachetului `babel`.

Vom coborî pentru o clipă sub etajul `betababel`-ului ca să vedem ce se petrece în cadrul `babel`.

Pentru început faceţi experimente de genul următor:

```
\documentclass{article}
\usepackage[cp1250]{inputenc}
\usepackage[LGR]{fontenc}
\begin{document}
Swkráthc
\end{document}
```

Σωκράτης

Observaţi cum, ca şi-n cazul literelor chirilice, am modificat codificarea la ieşire, nu pe cea de la intrare. Este normal să procedăm aşa atâta timp cât suntem interesate şi interesaţi de inserarea unor cuvinte sau mici fragmente de text în greacă într-un text care altfel este scris preponderent cu caractere latine.

Dac-aţi sesizat că *a* cu accent ascuţit are codul 225, atunci sunteţi pe drumul cel bun cu experimentele. Pasul următor constă în localizarea efectului obţinut mai sus. Scrieţi un alt proiect, în care preambulul arată astfel:

```
\documentclass{article}
\usepackage[cp1250]{inputenc}
\usepackage[LGR,T1]{fontenc}
\usepackage[polutonikogreek,romanian]{babel}
\newcommand{\textelin}[1]{\foreignlanguage{polutonikogreek}{#1}}
```

[92] Aristotel, An.pr.A **1.24**$^{\mathbf{b}}$ 18-20. Citat după textul grecesc editat de W.D.Ross, *Aristotelis analytica priora et posteriora*(Oxford: Oxford University Press, 1968).

Puneţi numele lui Socrate între acoladele comenzii `\textelin{}`. Acum doar acest cuvânt va apărea cu litere greceşti. Opţiunea `LGR` nu este acum neapărat necesară. De fapt, pachetul `babel` va face oricum apel la ea.

Acum ar trebui să fie şi mai limpede ce rol joacă `babel` şi ce rol joacă `betababel`. Când încărcăm `betababel` acest pachet cheamă pachetul `babel` şi introduce opţiunea `polutonikogreek`. De aceea, opţiunea aceasta este nu doar superfluă, ci generează chiar erori în cazul în care am încărcat pachetul `betababel`.

2.3.5.2 Ibycus

O codificare similară cu betacodul este cea folosită de către pachetul `ibycus`.[93] Trebuie să instalaţi însă separat acest pachet. El nu este inclus în distribuţia MikTEX. Este absolut recomandabil să-l puneţi în dosarul `localtexmf`, pentru a nu-l confunda cu pachetele standard MikTEX. Folosiţi pachetul `ibycus` separat de alte pachete, în documente destinate cu preponderenţă producerii de texte tipărite frumos în greaca veche.[94]

Iată un fragment, tot din Aristotel,[95] dar în codificarea ibycus:

```
le'gw de' oi(=on a)na'gkh me'n e)'sesqai naumaxi'an au)'rion h)'
mh' e)'sesqai, ou) me'ntoi gene'sqai au)'rion naumaxi'an
a)nagkai=on ou)de' mh' gene'sqai; gene'sqai me'ntoi h)' mh'
gene'sqai a)nagkai=on.S
```

Asemănările cu betacodul sunt limpezi. Există doar câteva deosebiri majore. Ibycus foloseşte distincţia dintre minuscule şi majuscule (nu steluţa pentru a marca majusculele). De asemenea, notarea accentelor ascuţit şi grav este diferită.

Folosirea semnului `;` este, de asemenea, diferită. În betacod, după obiceiul grecesc, acesta este semnul întrebării. Ibycus îl foloseşte pentru punctul ridicat. Afişat cu litere greceşti (folosind însă `betababel`) textul arată astfel:

[93] Pachetul a fost creat de către Pierre A. MacKay (University of Washington) pe baza unui tip de literă produs de către Silvio Levy. Levy a încercat să menţină tipul său de literă cât mai aproape de tipul de literă Didot, care-şi are originea la începutul secolului al XIX-lea.

[94] `Ibycus` poate fi folosit împreună cu opţiunea OT1 a pachetului `fontenc`, dar nu cu T1. Practic, `Ibycus` este de folosit când vreţi să produceţi text numai în greacă.

[95] Este vorba de un fragment din celebrul pasaj despre bătălia navală care va avea loc mâine. Vezi Aristotel, De Interpretatione **19a** 29-32.

λέγω δὲ οἷον ἀνάγκη μὲν ἔσεσθαι ναυμαχὶαν αὔριον ἢ μὴ ἔσεσθαι, οὐ μέντοι γενέσθαι αὔριον ναυμαχίαν ἀναγκαῖον οὐδὲ μὴ γενέσθαι· γενέσθαι μέντοι ἢ μὴ γενέσθαι ἀναγκαῖον.

S-ar putea să vreţi să folosiţi `ibycus` din pricina unor probleme ale pachetului `betababel`.[96]

2.3.6 LaTeX şi unicod

Dacă scrieţi într-un fişier de tip text, a apăsa pe o tastă la computer nu este totuna cu a pune o ştampilă sau cu a imprima o literă cu maşina de scris. Ceea ce pune, de fapt, în fişier computerul este un cod numeric. Până acum am văzut doar coduri formate din două cifre hex. În total există 256 de astfel de coduri distincte.

Dac-aţi studiat cumva tabelul de la pagina 127, rândul 5, aţi observat că nu toate cele 256 de coduri posibile sunt folosite pentru semnele diverselor alfabete. Motivul este foarte simplu: într-un fişier text este nevoie, de pildă, de coduri pentru sfârşitul de rând, precum şi de alte coduri speciale. Chiar şi aşa rămân destule coduri pentru limbile europene.

Toate limbile Europei folosesc un alfabet de un tip sau altul. Nu este posibil să fie puse toate semnele lor în corespondenţă un o singură pagină cu coduri de genul celor descrise mai sus. Faptul că folosim mai multe pagini cu coduri reprezintă motivul pentru care trebuie să specificăm codificarea *input*-ului. Ce se întâmplă însă în cazul unei scrieri ideografice? Ar fi greu să tot schimbăm pagina cu coduri la fiecare pas.

Ideea unicodului (a unei codificări universale) este să folosim patru cifre hex în loc de două. În acest caz, codul ultim din tabel ar fi *ffff*, ceea ce corespunde în sistemul zecimal cu *65535*. După cum se vede, diferenţa este uriaşă. Dar şi jungla creată este imensă; totul trebuie parcurs pas cu pas.

2.3.6.1 Pachetul `ucs`

După cum se vede mai sus, în cazul scrierilor care folosesc un alfabet, ne putem descurca excelent separând porţiunile în care folosim o anume codificare de celelalte. Cadrul oferit de codurile de la *0* la FF este totuşi strâmt. Pentru a avea acces la lumea codificării unicod,

[96] Sigma majusculă pune probleme în `betababel`. Urmăriţi situl cărţii pentru a detalii, exemple şi eventuale soluţii îmbunătăţite.

trebuie recurs la pachetul `ucs`. Încărcarea sa se face după modelul următor:

```
\usepackage{ucs}
\usepackage[utf8,cp1250]{inputenc}
\usepackage[LGR,OT2,T1]{fontenc}
\usepackage[romanian]{betababel}
\newcommand{\textelin}[1]{\foreignlanguage{polutonikogreek}{#1}}
```

Foarte important este faptul că pachetul `ucs` este încărcat înaintea pachetului `inputenc`. Observaţi, de asemenea, opţiunea `utf8` la pachetul `inputenc`. Această opţiune cere sistemului să preia şi coduri unicod. Ordinea opţiunilor este foarte importantă. Ultima opţiune este cea care are statut de `default` (este executată automat).

Puteţi folosi în continuare metodele descrise mai sus sau chiar le puteţi combina precum în acest fragment din Platon:[97]

```
\begin{quote}
\textelin{S}\bcode{WKRA/THS SOFO\S A)NH/R,
TA/ TE METE/WRA FRONTISTH\S KAI\}\ldots
\end{quote}
```

> Σωκράτης σοφὸς ἀνήρ, τά τε μετέωρα φροντιστὴς καὶ...

Codificarea amplă care este unicodul e imposibil de descris pe larg aici.[98] Principiul după care este construit pachetul `ucs` este acela al folosirii comenzilor LaTeX disponibile în diverse pachete LaTeX.[99] Practic, un cod unicod este tradus printr-o comandă LaTeX care, la rândul ei, generează comenzile necesare pentru a produce semnele (evident, dacă sunt disponibile în LaTeX).

Voi folosi tot exemple legate de limba greacă veche pentru a ilustra felul în care se aplică principiul de mai sus.

Să spunem c-aţi consultat tabelele unicod şi aţi văzut că litera alfa are codul hex 03b1. Faceţi calculele[100] pentru a trece în sistemul zecimal şi veţi afla că alfa minusculă are codul 945. Aceste numere de cod pot fi folosite ca-n exemplul care urmează.

[97] Platon, Apologia **18**b 7.

[98] Puteţi consulta listele de coduri utilizabile cu pachetul `ucs` la adresa de Internet `http://www.unruh.de/DniQ/latex/unicode/tables/index.html`. Listele sunt alcătuite chiar de către Dominique Unruh, care a realizat pachetul `ucs`.

[99] Atenţie la încărcarea lor atunci când este necesar!

[100] Puteţi folosi calculatorul din accesoriile Windows sau un program special de convertire de unităţi de măsură şi numere, cum este Versaverter de la `http://pawprint.net`, care este un program gratuit.

```
\textelin{\unichar{945}\\
          \unichar{946}\\
          \unichar{947}}
```

α
β
γ

Aşa cum spuneam, sistemul unicod este, de fapt, transpus în comenzi LaTeX. De unele dintre ele s-ar putea să aveţi nevoie, mai ales în cazuri speciale. Iată două exemple:

```
\textelin{\textdigamma}\\
\textelin{\textsanpi}
```

Ϝ
ϡ

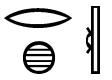
Ultimul simbol are codul hex 03e1 (zecimal 993). Alte coduri şi comenzile aferente lor pot fi studiate cu ajutorul documentaţiei pachetului `ucs` sau citind chiar programele din pachet ca atare.

2.3.7 Alte pachete cu simboluri în LaTeX

Unicod are ample posibilităţi, dar cred că nu trebuie să exagerăm cu recursul la unicod. Dacă scriem într-o limbă europeană, s-ar putea să folosim una-două pagini cu coduri şi n-are rost să încărcăm zeci de mii de simboluri pentru a folosi câteva zeci.

Persoanele pasionate de gândirea orientală, de culturile vechi vor fi desigur atrase de unicod şi posibilităţile acestuia. Dar chiar şi acestor persoane nu trebuie să le scape pachetele care s-ar putea să se potrivească mai bine cu obiectivele lor.

Favoritele mele sunt pachetele concepute de către Peter Wilson pentru unele scrieri arhaice.[101] Pachetele acestea sunt simple, se compilează uşor şi sunt bune pentru cineva care are cunoştinţe extrem de limitate despre scrierile respective. Exemplul care urmează provine din scrierea vechilor egipteni.


este un cuvânt care se transcrie fonetic *rḫ*. Înseamnă *a şti*. **Hieroglife**

Simbolul  este o gură stilizată. Simbolul  este o sită. Aceste simboluri sunt folosite aici cu valoarea lor fonetică, dar Christian Jacq arată că erau posibile asocieri cu transmiterea preponderent verbală a cunoaşterii şi cu procesul de selecţie, de cernere a ideilor.[102] Semnul  este un determinativ pentru conceptele abstrate. Christian Jacq sugerează că prin inversarea sitei şi a gurii obţinem un interesant joc de cuvinte. Termenul rezultat, care se transcrie

[101] A se vedea documentaţia creată pentru aceste pachete de către Peter Wilson <`peter.r.wilson@boeing.com`> în dosarul `\texmf\doc\doc\fonts\archaic`.

[102] Christian Jacq *Sag's mit Hieroglyphen: Lesen und Schreiben wie die alten Ägypter* (Hamburg: Rowohlt, 2003), p.169.

fonetic *ẖr*, înseamnă *a cădea*. Determinativul este acum un om aflat în cădere, un om care, ignorant fiind, este condamnat la decădere.[103]

Cu pachete LaTeX puteţi scrie şi-n armeană, georgiană, sanscrită, chineză şi multe alte limbi. Pachetele au sursele deschise şi sunt chiar mai uşor de folosit decât un program cu surse ascunse.

METAFONT

Nu găsiţi literele sau simbolurile de care aveţi nevoie? Donald Knuth a creat un limbaj special pentru generarea de tipuri de litere (*fonturi*). Limbajul se numeşte METAFONT.[104] Pentru hieroglifele egiptene din pachetul lui Peter Wilson vedeţi fişierul `pmhg.mf` din dosarul `\texmf\fonts\source\public\archaic`.

În principiu, dacă aţi dobândit competenţa necesară, în LaTeX puteţi include orice simbol. Desigur, la nevoie, trebuie să creaţi simboluri. Este însă foarte probabil că, în lumea largă a utilizatorilor de LaTeX, există deja cineva care a conceput o soluţie la problema care vă frământă.

2.4 Tabele şi formule

Multe dintre eseurile filosofice nu au deloc tabele sau formule. Persoanele care folosesc intens ananaliza logică au însă nevoie cel puţin de cunoştinţele de bază privitoare la scrierea de formule. De asemenea, multe eseuri de filosofie politică recurg la tabele statistice.

Voi începe cu tabelele pentru că înţelegerea modului în care se construiesc şi se folosesc tabelele va ajuta mult la formarea deprinderilor necesare pentru a scrie formule complexe.

2.4.1 Principiile de bază ale construirii tabelelor

Tipul cel mai simplu de tabel conţine doar text aranjat pe linii şi coloane. Voi descrie întâi modul de a construi un mic tabel, pe care-l voi folosi mai jos pentru a realiza un tabel ceva mai complicat.

Tabelul simplu are două coloane şi două rânduri. Nu serveşte decât la dispunerea a patru cuvinte în cele patru colţuri ale unei mici pagini.

```
\begin{tabular}{cc}
Liberal  & Libertarian \\
Populist & Conservator \\
\end{tabular}
```

Liberal	Libertarian
Populist	Conservator

Tabelul este construit într-un mediu denumit `tabular`. După

[103] *Ibidem.*

[104] A se vedea aici 3.3.

`\begin{tabular}` trebuie adăugată o pereche de acolade între care este precizată alinierea conţinutului coloanelor. Alinierea este indicată cu ajutorul literelor `l`, `c`, `r` (pentru aliniere stânga, centru, respectiv dreapta). În cazul nostru, ambele coloane au textul centrat. Pe fiecare rând, coloanele sunt separate prin semnul `&`. Capetele de rând sunt indicate prin \\ şi este important ca fiecare rând să aibă exact numărul de coloane specificat.

Pentru a crea tabele sau scheletul unor tabele ar fi bine însă, la început cel puţin, să folosiţi un program de creare a tabelelor în LaTeX. Puteţi folosi, de pildă, LaTable. Acesta este un program gratuit, creat de către Alex A. Denisov[105]. Interfaţa grafică a programului permite crearea tabelului fără a recurge direct la comenzile LaTeX.

Un tabel este în fapt rodul unui mic proiect. Ca la orice proiect, este necesar să precizaţi obiectivele. Căutaţi apoi mijloacele cele mai potrivite pentru a atinge obiectivele fixate. Desfaceţi totul pe module şi construiţi fiecare modul separat, pentru a nu fi victime ale complexităţii tabelului.

Proiectul meu aici este să reproduc un tabel cu tipurile de ideologii, creat de către William S. Maddox şi Stuart S. Lilie.[106] Miezul tabelului mai mare îl constituie micul tabel de mai sus. Maddox şi Lilie disting însă între două dimensiuni ale dezbaterii publice, între două seturi de probleme: chestiunile legate de extinderea libertăţilor personale (accesul la informaţii, dreptul la avort etc.) şi chestiunile legate de intervenţia statului în economie. Fiecare tip de om politic are o atitudine: pro sau contra libertăţilor personale extinse, pro sau contra intervenţiei statului în economie. Introducerea celor două seturi de probleme şi a atitudinilor în problemele respective complică desigur tabelul.

Procedez pas cu pas. Fac abstracţie de termenii „liberal“, „libertarian“, „populist“, „conservator“; pun doar nişte semne convenţionale în locul lor. Mai întâi introduc un rând şi o coloană suplimentare. Pun doar `+` şi `-`, după caz, în patru casete ale tabelului şi las colţul din stânga sus liber.

Mai complicat este cu dimensiunile ca atare. Textul este lung şi aş avea imediat probleme cu spaţiul în pagină. Trag concluzia că, într-o primă fază, este suficient să scriu un 1, respectiv 2. Dar unde? Trebuie ca 1 să ocupe două coloane, iar 2 două rânduri! Asta şi fac; pun 1 în două casete şi 2 în două casete.

[105] Pagina de web a programului este `http://g32.org`.

[106] William S. Maddox şi Stuart A. Lilie, *Beyond Liberal and Conservative: Reassessing the political spectrum* (Washington: Cato Institute, 1984), p.5.

```
\begin{tabular}{lccc}
  &   &  1  &  1 \\
  &   &  +  &  - \\
2 & + & L1  & L2 \\
2 & - &  P  &  C
\end{tabular}
```

		1	1
		+	-
2	+	L1	L2
2	-	P	C

Rezultatul este o schemă a tabelului final. Pe baza acestei scheme vom dezvolta tabelul final.

2.4.1.1 Tabelul complet şi trimiterile la tabele

De multe ori este nevoie să ne referim la un tabel care nu se află în imediata apropiere. În acest caz tabelului trebuie să-i dăm un nume şi trebuie să declarăm o etichetă care să ne permită să folosim mecanismul trimiterilor din LaTeX.

Până acum tabelele au fost moduri de a diviza convenabil spaţiul disponibil pe o pagină. Acum tabelul trebuie încastrat într-un mediu LaTeX care-i permite să *plutească* în pagină.[107]

Poziţia corpului plutitor o alege, în cele din urmă, LaTeX. Avantajul din punctul de vedere al autoarei sau autorului este că se poate referi, precum în cazul nostru, la „tipuri de ideologii“ adăugând şi „vezi tabelul 2.1“.

Cum am obţinut tot acest efect? Planul a fost foarte simplu. Am luat schema de tabel deja creată şi am introdus-o într-un mediu `table`. În mediul `table` se poate da o comandă `\caption{}` pentru titlul tabelului. Eticheta este ataşată tot mediului `table`. Trimiterea la tabel se face cu obişnuita comandă `\ref{}`.

```
\newcommand{\mc}[3]{\multicolumn{#1}{#2}{#3}}
\begin{table}[ht]
\centering
\begin{tabular}[c]{lccc}
              &                  & \mc{2}{c}{Intervenţia}\\
              &                  & \mc{2}{c}{statului}\\
              &                  & \mc{2}{c}{în economie}\\
              &                  &     pro     &     contra\\
\cline{3-4}
              & \mc{1}{r|}{}     &             &\mc{1}{c|}{}\\
Extinderea    & \mc{1}{r|}{pro}  & L1          &\mc{1}{c|}{L2}\\
libertăţilor  & \mc{1}{r|}{}     &             &\mc{1}{c|}{}\\
personale     & \mc{1}{r|}{contra}& P          &\mc{1}{c|}{C}\\
\cline{3-4}
\end{tabular}
\caption{Tipuri de ideologii după Maddox şi Lilie}
```

[107] Termenul tehnic în engleză este cel de *float*. Nu doar imaginile plutesc în pagini. Notele marginale sunt, de pildă, şi ele structuri plutitoare.

```
\label{TipuriDeIdeologii}
\end{table}
```

Comanda `\multicolumn{}{}{}` merită o atenţie specială. Pentru că numele ei este lung am redenumit-o `mc` (pe rândul 1 din fragmentul de sursă LaTeX de mai sus). Primul ei argument specifică numărul de coloane. Acest număr poate fi şi 1! Al doilea argument arată cum este aliniat textul. Al treilea argument este reprezentat de text ca atare. În rezumat, este o comandă care spune pe câte coloane este scris un text în tabel şi cum este aliniat.

Dacă după litera care specifică alinierea punem o bară verticală[108], atunci în tabel, la capătul coloanei(coloanelor) respective apare o linie verticală.

Liniile orizontale sunt trasate cu `\cline{}`; în argumentul acestei comenzi trebuie specificat de la ce coloană până la ce coloană se trage linia. Comanda `\hline` trage o linie orizontală de-a lungul întregului tabel.

		Intervenţia statului în economie	
		pro	contra
Extinderea libertăţilor personale	pro	liberalism	libertarianism
	contra	populism	conservatorism

Tabelul 2.1: Tipuri de ideologii după Maddox şi Lilie

Pentru a obţine tabelul 2.1 nu a mai fost nevoie decât să refac conţinutul miezului iniţial al tabelului.

Dacă numerotarea tabelelor nu începe de la 1, daţi comanda `\setcounter{table}{0}` înainte de primul tabel.

2.4.1.2 Tabelele şi aranjarea textului în pagină

Tabelele sunt folosite intens şi pentru a crea aranjamente (uneori destul de complicate) ale textului în pagină. De exemplu, pentru a aranja lista de coduri şi simboluri de la pagina 128 am folosit un tabel. Redăm aici doar comenzile pentru un rând, fără a include codurile sau simbolurile ca atare.

[108] În limbajul Vim este vorba despre `<char-124>`.

```
\begin{longtable}[c]{lllll}
---\textless{}...\textgreater{}&
---\textless{}...\textgreater{}&
---\textless{}...\textgreater{}&
---\textless{}...\textgreater{}&
---\textless{}...\textgreater{}\\
.............................
\end{longtable}
```

Scopul este dispunerea textului în aşa fel încât să fie cât mai lizibil. Am folosit mediul `longtable` pentru ca LaTeX să poată decupa tabelul în porţiuni care sunt afişate pe pagini diferite. Opţiunea `c` din primul rând de comenzi cere sistemului să centreze întregul tabel în cadrul paginii.

2.4.2 Câteva idei simple despre formule

LaTeX este îndeosebi faimos pentru capacitatea sa de a genera formule matematice. În secţiunea 2.1.3.4.3, referitoare la modul matematic, am arătat deja cum ne putem sluji de modul matematic în rândurile de text. Multe eseuri filosofice conţin formule în limbajul simbolic al logicii. Este, de aceea, util să trecem în revistă modul simbolistica de bază a logicii poate fi realizată în LaTeX.

```
Negaţia : $\neg{}p$\\
O notaţie pentru conjuncţie:\\
$p\&{}q$\\
Altă notaţie pentru
   conjuncţie:\\
   $p\wedge{}q$\\
Implicaţia: $p\to{}q$\\
Echivalenţa: $p\equiv{}q$
```

Negaţia : $\neg p$
O notaţie pentru conjuncţie:
$p\&q$
Altă notaţie pentru conjuncţie:
$p \wedge q$
Implicaţia: $p \to q$
Echivalenţa: $p \equiv q$

Acestea sunt doar câteva dintre posibilele soluţii în cazul logicii propoziţiilor. Trebuie consultate tabelele cu simboluri matematice din documentaţia MikTeX pentru a descoperi întregul evantai de simboluri.

În cazul logicii predicatelor putem folosi:

```
Cuantificare existenţială:\\
$(\exists{}x)(Px\wedge{}Qx)$\\
Cuantificare universală:\\
$(\forall{}x)(Px\to{}Qx)$
```

Cuantificare existenţială:
$(\exists x)(Px \wedge Qx)$
Cuantificare universală:
$(\forall x)(Px \to Qx)$

Operaţiile cu mulţimi şi relaţiile dintre mulţimi sunt şi ele uşor de reprezentat cu ajutorul LaTeX.

```
$x$ aparţine mulţimii $X$:
      $x\in{}X$\\
Negaţia apartenenţei:
      $x\notin{}X$\\
Incluziunea: $Y\subset{}X$\\
Intersecţia: $X\cap{}Z$\\
Reuniunea: $X\cup{}Z$\\
Complementara: $\sim{}X$
```

x aparţine mulţimii X: $x \in X$
Negaţia apartenenţei: $x \notin X$
Incluziunea: $Y \subset X$
Intersecţia: $X \cap Z$
Reuniunea: $X \cup Z$
Complementara: $\sim X$

2.4.2.1 Formulele din cărţile reale

Rezultatele matematice sunt exprimate ca formule în limbaj simbolic şi, de regulă, sunt scrise pe rânduri separate. Oricum, atunci când citǎm formule din cărţile altora, trebuie să le punem pe un rând separat.

De exemplu, Adrian Miroiu, într-un studiu în care dezvoltă ideea sa a lumilor în interiorul lumilor, introduce un prim rezultat general sub forma următorului enunţ:[109]

$$\vdash_{\mathrm{LK}} \varphi \text{ ddacă } \models_{\mathrm{LK}} \varphi$$

Iată cum am codificat în LaTeX enunţul din cartea lui Adrian Miroiu:

```
\begin{displaymath}
\mathrm{\vdash_{\scriptscriptstyle LK}\varphi\ ddac\breve{a}\
\models_{\scriptscriptstyle LK}\varphi}
\end{displaymath}
```

Codificarea LaTeX este destul de transparentă pentru cei care ştiu logică. S-ar putea spune chiar că oferă un plus de explicaţii. Trebuie totuşi făcute câteva comentarii.

Prima observaţie este aceea că trebuie să tratăm modul matematic ca un loc în care n-are sens să venim cu obiceiurile de a scrie în mod text. În primul rând, în mod automat, LaTeX scrie cursiv textul matematic. De aici necesitatea comenzii `\mathrm{}`, care-i spune sistemului să scrie cu tipul de literă roman.

litere greceşti în modul matematic

În al doilea rând, folosirea literelor în modul matematic are particularităţile sale.[110] Literele greceşti, de pildă, se obţin cu ajutorul unor comenzi speciale. Am folosit `\varphi` şi nu `\phi` pentru că aceasta era varianta lui φ folosită în textul lui Adrian Miroiu.

În al treilea rând, trebuie observat rolul spaţierii şi alegerii dimensiunii literelor în modul matematic. Barele oblice inverse urmate

[109] Adrian Miroiu, *Constructe formale* (Bucureşti: Editura Trei, 2000), p.16.
[110] Pentru literele româneşti în modul matematic vezi §2.4.2.2.

de un spaţiu sunt comenzi! Ele cer introducerea unui spaţiu, menit să elimine eventuale ambiguităţi. De asemenea, o pereche de acolade (fără nimica între ele) forţează mărirea spaţiului alb. De aici lipsa lor după `\vdash` sau `\models`. Cu acolade după comenzi, rezultatul ar fi $\vdash_{\text{LK}}\varphi$ sau $\models_{\text{LK}}\varphi$.

Declaraţia `\scriptscriptstyle` corectează dimensiunea indicelui, reducând-o la dimensiunea unui indice la indice. Observaţi, în acest sens, diferenţa dintre $\vdash_{\text{LK}}\varphi$ şi $\vdash_{\text{LK}}\varphi$.

Un alt exemplu, extras dintr-o carte a lui Mircea Dumitru, pune şi el probleme interesante de tehnoredactare.[111] Fragmentul citat mai jos este o parte din definiţia noţiunii de satisfacere în logica modală a propoziţiilor(LLMP). Iată fragmentul:

> pentru fiecare fbf Φ în LLMP,
>
> $(\mathcal{M}, w) \models \Diamond\Phi$ ddacă $(\exists u \in W_{\mathcal{M}})(R_{\mathcal{M}}wu$ şi $(\mathcal{M}, u) \models \Phi)$

Prima problemă a codificării este scrierea pe mai multe rânduri. Ar trebui să construim un tabel! A se vedea acest lucru, pentru modul matematic, în §2.4.2.4. Formula în sine este uşor de înţeles pentru cei care au citit un curs elementar de semantică a logicilor modale. Chiar dacă nu o înţelegeţi, problemele de tehnoredactare sunt interesante în sine.

Există o simetrie între cele două rânduri citate. Primul este scris în modul text, dar are o literă grecească realizată în mod matematic. Al doilea rând este scris în modul matematic, dar are două inserţii în mod text după cum se vede din rândurile de cod de mai jos.

```
pentru fiecare fbf $\Phi$ în LLMP,
\begin{displaymath}
(\mathcal{M},w)\models{}\ \diamondsuit{}\Phi{}\ \text{ddacă}\
(\exists{}u\in{}W_{\mathcal{M}}) (R_{\mathcal{M}}wu\ \text{şi}\
(\mathcal{M},u)\models{}\Phi{})
\end{displaymath}
```

text în modul matematic

Comanda `\text{}` cu ajutorul căreia am inserat text în modul matematic nu este accesibilă dacă aţi încărcat doar nucleul sistemului LaTeX. Trebuie să încărcaţi pachetul `amstext`.[112]

[111] Mircea Dumitru, *Modalitate şi incompletitudine* (Bucureşti: Paideia, 2001), pp.22-23.

[112] Pachetul `amstext` ar fi încărcat automat dacă aş folosi $\mathcal{AMS}$-LaTeX. n-are însă rost să încarc artileria grea de la American Mathematical Society în această carte. Pentru o explicaţie mai detaliată a comenzii `\text{}` vezi cartea lui Paul A. Blaga şi Horia F. Pop[1, p.156].

Oricât ar putea părea de curios problema cea mare este litera ş de pe rândul 4 din codul de mai sus. Obţinerea ei cu ajutorul comenzii `\text{}` este mai uşoară decât realizarea ei în modul matematic pur.

Deşi exemplele sunt simple în conţinutul lor, tehnoredactarea lor pune acelaşi tip de probleme pe care le-ar formulele cu un conţinut mai dificil din cărţile citate. Pentru listele de simboluri care pot fi utilizate trebuie consultată documentaţia.[113]

2.4.2.2 Modul matematic şi literele româneşti

Punerea de diacritice pe litere în modul matematic diferă de operaţia similară din modul text. Dacă scriu un ă în mod matematic, sistemul traduce acest lucru prin comanda `\u{a}`, care este inacceptabilă în mod matematic. Trebuie scris `\breve{a}` în mod matematic.

Iată literele româneşti în mod matematic pur (fără pachetul `amstext`):

```
$\breve{a}$  $\breve{A}$\\
$\hat{a}$    $\hat{A}$\\
$\hat{\imath}$    $\hat{I}$\\
$\stackrel{\textstyle{}s}{,}$
$\stackrel{\textstyle{}S}{,}$\\
$\stackrel{\textstyle{}t}{,}$
$\stackrel{\textstyle{}T}{,}$
```

$\breve{a}$ $\breve{A}$
$\hat{a}$ $\hat{A}$
$\hat{\imath}$ $\hat{I}$
$\stackrel{\textstyle{}s}{,}$ $\stackrel{\textstyle{}S}{,}$
$\stackrel{\textstyle{}t}{,}$ $\stackrel{\textstyle{}T}{,}$

Comenzile pentru ş şi ţ în mod matematic sunt meşterite cu ajutorul comenzii `\stackrel{}{}`. Ea este larg folosită pentru a produce simboluri precum $\stackrel{df}{=}$ (cu comanda `$\stackrel{df}{=}$`).

Exemplele de mai sus nu au probabil mare valoare practică, dar ele ilustrează posibilitatea de a crea în LaTeX simbolurile de care avem nevoie.

2.4.2.3 Numerotarea formulelor

Putem pune pe rânduri separate formule, fără să le numerotăm. Este păcat însă să nu ne folosim de posibilitatea de a numerota formulele. Putem apoi folosi mecanismul standard din LaTeX pentru a ne referi la o formulă.

[113] Eu am folosit în special Scott Pakin „The Comprehensive LaTeX Symbol List“ (v. fişierul `symbols.dvi` în dosarul `comprehensive` din documentaţia MikTeX).

Deşi pare complicată, axioma Nicod–Łukasiewicz pentru logica propoziţiilor(2.1) nu este greu de scris în LaTeX.

$$[p/(q/r)]/\Big([s/(s/s)]/\{(s/q)/[(p/s)/(p/s)]\}\Big) \tag{2.1}$$

Propoziţia compusă p/q este falsă dacă ambele variabile propoziţionale iau valoarea *adevărat*; altfel este adevărată. Codul LaTeX folosit la scrierea mediului în care este plasată axioma este următorul:

```
\begin{equation}
[p/(q/r)]/\bigg([s/(s/s)]/\{(s/q)/[(p/s)/(p/s)]\}\bigg)
\label{NicodLukasiewicz}
\end{equation}
```

Referirile la formulele numerotate se fac cu ajutorul comenzii `\ref{}` care ia drept argument eticheta declarată în cadrul mediului în care este scrisă formula.

2.4.2.4 Tabele care conţin formule

Tabelele care conţin text se construiesc în mediul `tabular`. Ar fi posibil să introducem simboluri sau formule matematice şi-n aceste tabele, prin trecerea în porţiunea respectivă în modul matematic. Recomandabil este însă să folosim pentru tabelele cu formule mediul `array`.

Voi începe cu un tip de tabel familiar oricui începe să înveţe logică: matricile conectorilor propoziţionali.

```
\begin{displaymath}
\begin{array}{llc}
p & q & p \to{}q \\
\hline
1 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 1 \\
0 & 0 & 1
\end{array}
\end{displaymath}
```

$$\begin{array}{llc}
p & q & p \to{}q \\
\hline
1 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 1 \\
0 & 0 & 1
\end{array}$$

Pentru un tabel asemenea celui de mai sus, scrieţi mai întâi comenzile pentru mediul care afişează formule nenumerotate. Procedaţi apoi ca şi-n cazul mediului `tabular`, dar folosiţi mediul `array`, în care puteţi introduce formule.

Tabelele cu formule sunt evident mult mai utile dacă ne putem referi la ele „de la distanţă“. De exemplu, vreau să vă dau un exemplu

de demonstrare a unei tautologii din logica propoziţiilor. Atunci voi trimite la tabelul 2.2.

p	q	$p \to q$	$p \wedge (p \to q)$	$(p \wedge (p \to q)) \to q$
1	1	1	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	1

Tabelul 2.2: Demonstrarea unei tautologii

Sursa LaTeX a tabelului 2.2 sper că este suficient de clară pentru a înţelege modul în care este construit tabelul. Am evitat în mod special utilizarea `&` drept conectiv pentru conjuncţie pentru a nu genera confuzii. Simbolul rezervat `&` este separatorul coloanelor, nu şi nume pentru o comandă de generare a simbolului conectorului conjuncţiei.

```
\begin{table}[htb]
\centering
\begin{displaymath}
\begin{array}{llccc}
p & q & p\to{}q & p\wedge{}(p\to{}q) & (p\wedge{}(p\to{}q))\to{}q\\
\hline
1 & 1 &    1    &          1         &              1            \\
1 & 0 &    0    &          0         &              1            \\
0 & 1 &    1    &          0         &              1            \\
0 & 0 &    1    &          0         &              1
\end{array}
\end{displaymath}
\caption{Demonstrarea unei tautologii}
\label{DemonstrareaUneiTautologii}
\end{table}
```

Puteţi localiza poziţia unui tabel în cadrul sursei LaTeX cu ajutorul TeXnicCenter. Puteţi, de asemenea, produce o listă a tabelelor în documentul final, plasând comanda `\listoftables` după comanda `\tableofcontents` din fişierul principal al proiectului.

Este necesar să comentăm este necesar puţin folosirea liniilor în tabele. Recomandarea noastră este de a folosi linii doar pentru a face tabelul clar. Liniile superflue sau cu caracter pur decorativ nu-şi au locul într-o lucrare academică.

O linie sub capul de tabel este, de multe ori, suficientă. Dacă tabelul conţine elemente de sinteză în partea de jos a coloanelor, atunci

o linie simplă sau dublă este necesară deasupra ultimului rând. Ea face tabelul mai clar. Abuzul de linii este total nerecomandabil.

Crearea de tabele poate fi un lucru extrem de pretenţios. Adesea ele sunt prea late sau prea lungi şi aşa mai departe. Prezentarea noastră a fost doar una sumară. Trebuie să citiţi o carte despre LaTeX care acordă mai mult spaţiu tabelelor.[114] Foarte utilă este şi documentaţia inclusă în distribuţia MikTeX.

2.4.2.5 Formulele ca tabele

Am pregătit recent un text în care voiam să citez o funcţie pe care o discută Kripke.[115] Funcţia, pe care Kripke o numeşte „quus“ (simbolic $\oplus$), generează două tipuri de valori: suma valorilor argumentelor sale şi o valoare constantă. Condiţia pentru a genera primul tip de rezultate este ca valoarea oricăruia dintre cele două argumente ale funcţiei să fie mai mică de 57. Altfel, este generată constanta.

Pentru a descrie simbolic funcţia *quus* Kripke foloseşte o notaţie cu două semne de egalitate şi nici o acoladă. Practica uzuală este însă diferită şi o voi respecta aici.[116]

Chiar dacă descrierea simbolică nu este complicată, trebuie să gândim tot din perspectiva proiectului. Obiectivele noastre sunt să prezentăm:

unu numele funcţiei şi argumentele sale;

doi valoarea generată când este adevărată prima condiţie;

trei valoarea generată când este adevărată a doua condiţie.

Pentru a atinge primul obiectiv este suficient să scriem un rând de text matematic. Al doilea şi al treilea obiectiv presupun folosirea unui tabel (cu două coloane şi două rânduri).

Construim separat elementele formulei 2.2 ca-n exemplele care urmează. Ca deobicei, este important să executăm totul pas cu pas. Scriem mai întâi rândul de text matematic.

```
\begin{displaymath}
x \oplus y =
\end{displaymath}
```

$$x \oplus y =$$

[114] A se vedea, în limba română, cartea lui Paul A. Blaga şi Horia F. Pop[1, pp.50-55, 212-219].

[115] În Saul A. Kripke, *Wittgenstein on Rules and Private Language* (Oxford: Blackwell, 1982), p.9.

[116] Pentru un model care respectă standardele uzuale v. Lamport[4, p.47].

Elaborăm apoi tabelul cu elementele generate de către funcţie, în cele două condiţii specificate mai sus.

```
\begin{displaymath}
\begin{array}{ll}
  x+y, & \text{dacă}\ x,y<57\\
    5  & \text{altfel}
\end{array}
\end{displaymath}
```

$$\begin{array}{ll} x+y, & \text{dacă } x,y<57 \\ 5 & \text{altfel} \end{array}$$

Testăm posibilitatea de a pune tabelul între acolade.

```
\begin{displaymath}
\left\{
 \begin{array}{ll}
   x+y, & \text{dacă}\ x,y<57\\
     5  & \text{altfel}
 \end{array}
\right\}
\end{displaymath}
```

$$\left\{ \begin{array}{ll} x+y, & \text{dacă } x,y<57 \\ 5 & \text{altfel} \end{array} \right\}$$

Acum nu este greu să obţinem forma finală, corectând eventuale inexactităţi, şi eliminând acolada din dreapta.

$$x \oplus y = \left\{ \begin{array}{ll} x+y, & \text{dacă } x,y<57 \\ 5 & \text{altfel} \end{array} \right. \tag{2.2}$$

Perechea de comenzi `\left \right` permite crearea de perechi de paranteze. Felul în care arată paranteza depinde de felul în care completăm numele comenzii. Dacă punem `{` apare acolada din stânga. Putem pune şi o paranteză rotundă, una pătrată sau o bară verticală (pentru matrici). Dacă vrem ca paranteza să fie invizibilă, atunci punem un punct după numele comenzii. Iată acum şi forma finală a codului LaTeX pentru descrierea funcţiei lui Kripke.

```
\begin{equation}
x \oplus y = \left\{
                \begin{array}{ll}
                   x+y,    &   \text{dacă}\ x,y<57 \\
                   5       &   \text{altfel}
                \end{array}
             \right.
\label{quus}
\end{equation}
```

În ciuda aparentei simplităţi a codului LaTeX, dacă încercaţi să-l construiţi fără nici un plan, s-ar putea să vă treziţi într-un noian de erori. Orice tabel, oricât de simplu, trebuie proiectat atent.

2.4.3 Tehnicile avansate de scriere matematică

Scrierea surselor LaTeX pentru modul matematic este un lucru mult mai pretenţios decât pregătirea părţii care este în mod text. Din fericire, cărţile de logică sau de filosofia ştiinţei, cele care folosesc deobiecei cel mai intens limbajele simbolice, nu conţin la tot pasul sisteme de ecuaţii complicate.

În cazul când vreţi să scrieţi sisteme de ecuaţii sau să grupaţi grafic rânduri cu formule, trebuie să folosiţi mediul `eqnarray`.

Dacă doriţi să obţineţi rezultate şi mai sofisticate, trebuie să recurgeţi la `amsmath` şi alte pachete dedicate special matematicii.

Marea majoritate a introducerilor în LaTeX sunt scrise din perspectiva utilizării sistemului pentru a tehnoredacta studii şi cărţi de matematică şi informatică. Nu veţi duce lipsă de surse de documentare dacă vă interesează tehnicile avansate de elaborare a textelor care conţin formule complicate.[117]

2.5 Indexarea electronică

LaTeX foloseşte un program special pentru a genera indici. Folosirea acestui compilator de indici este mai simplă decât în cazul BibTeX.

Ca şi-n cazul BibTeX, mediul integrat TeXnicCenter are,la proprietăţile proiectului, o opţiune pentru folosirea programului MakeIndex[118]. Această opţiune trebuie bifată, dacă vreţi să generaţi un indice.

Studiaţi, de asemenea, Define Output Profiles... din meniul Build al TeXnicCenter şi veţi vedea cum este apelat `makeindex.exe` în mediul integrat.

În preambulul documentului principal al proiectului trebuie să introduceţi două comenzi LaTeX:

```
1 \usepackage{makeidx}
2 \makeindex
```

La sfârşitul corpului documentului principal, după comenzile pentru bibliografie, introduceţi comanda `\printindex`.

[117] În limba română puteţi consulta cartea lui Paul A. Blaga şi Horia F. Pop[1].

[118] Autorul programul *MakeIndex* este Pehong Chen. Pentru detalii referitoare la program şi utilizarea lui, a se vedea Leslie Lamport, ***MakeIndex**: An Index Processor for LaTeX*, `\texmf\doc\makeindex\makeindex.dvi` în documentaţia MikTeX.

Marele avantaj al indexării în LaTeX este posibilitatea de a indexa din mers. N-aş recomanda însă acest fel de a indexa. Pentru a afla principiile indexării studiaţi un manual de indexare.

Comanda de bază este `\index{}`. În argumentul ei se plasează un descriptor. Comanda aceasta trebuie folosită precum comanda `\label{}`; ea trebuie pusă acolo unde vrem ca indicele de la sfârşit să ne trimită în text.

Creaţi un proiect test şi indexaţi în felul indicat mai sus. Vedeţi ce apare în fişierul `dvi`. Examinaţi, de asemenea, fişierul de tip `ind`, care este generat automat de *MakeIndex*. Nu este greu să vă daţi seama c-aţi putea crea acest fişier şi manual. Atâta timp cât utilizaţi *MakeIndex*, nu operaţi schimbări în acest fişier. Ele vor fi distruse la orice recompilare. Modificaţi comenzile din surse.

Puteţi folosi subdescriptori după modelul de mai jos:

```
\index{indici!generarea manuală}
\index{indici!generarea lor cu MakeIndex}
```

Descriptorul este separat de subdescriptor cu ajutorul semnului exclamării.

Pentru a realiza trimiteri de la un descriptor la altul în cadrul listei de indici folosiţi modelul următor:

```
\index{editor de texte|see{Vim}}
```

Bara verticală precede trimiterea. Pachetul `babel` va traduce pe `see`.

Folosiţi capacităţile de căutare cu ajutorul expresiilor regulate ale TeXnicCenter pentru a unifica intrările. De asemenea, trebuie să compilaţi cel puţin de două ori documentul pentru ca să apară modificările operate în indice.[119]

Pentru a imita notarea tradiţională a descriptorilor pe marginea paginii folosiţi pachetul `showidx`. Pachetul are un rol pur auxiliar. Nu eliminaţi *bad box*-urile generate de notele marginale introduse de `showidx`. Opriţi utilizarea acestui pachet după ce aţi terminat indexarea.

[119] Prima oară este modificat doar fişierul de tip `ind`. Abia a doua oară se modifică şi documentul final. Dacă aveţi probleme, citiţi mesajele compilatorului.

Capitolul 3

Pensula electronică

Cuprins

3.1 Imaginile **159**
3.1.1 Programul Gimp 161
3.1.2 Limbajul PostScript 162
3.2 Inserarea imaginilor în LaTeX **164**
3.2.1 Inserarea de imagini în fişiere `pdf` . . . 165
3.3 Inserarea literelor ca inserare de imagini **167**
3.3.1 Desenarea unei litere 167
3.3.2 Inserarea literei în text 170

LaTeX are propriile sale comenzi şi medii pentru realizarea de diagrame şi desene cu caracter tehnic. Prezentarea lor ar depăşi cu mult ambiţiile foarte reduse ale acestei anexe.

Eseurile filosofice nu abundă în diagrame şi nici în desene sau imagini. Câteva noţiuni elementare privitoare la imaginile computerizate şi integrarea lor în sursele LaTeX sunt suficiente pentru persoanele care scriu un eseu filosofic.

3.1 Imaginile

În anexa 2 ne-am referit adesea, în mod metaforic, la „literele de plumb“ ale meşterului tipograf. Computerul n-are litere de plumb. Literele de pe ecran sunt desenate din nişte puncte denumite „pixeli“[1]; punctele acestea pot fi albe, negre sau colorate.

Dacă imprimaţi pe un printer cu jet de cerneală, literele de pe foaia de hârtie sunt compuse din picături de cerneală extrem de mici.

[1] Denumirea vine de la *picture element* (element al unei imagini).

Atunci când tehnoredactăm un text în LaTeX nu ne interesează punctele de pe ecran sau de pe foaia de hârtie. Gândim din perspectiva cutiilor cu litere plasate pe ecran sau pe foaie, nu gândim la nivelul pixelilor. Dacă vrem să înţelegem însă secretele imaginilor generate de către computer, trebuie să ne îndreptăm atenţia către aceste puncte minuscule.

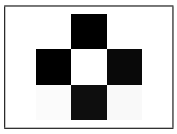
O primă strategie de realizare a imaginilor pe ecran este aceea de a nota într-un fişier informaţiile despre pixeli, în aşa fel încât să fim în stare să reconstituim imaginea pe ecran cu ajutorul unui program de vizualizare.

Puteţi face, folosind Vim ca pe un editor de coduri hex, precum şi un program de vizualizare ca IrfanView,[2] câteva experimente cu un fişier ce stochează imagini.

Deschideţi IrfanView şi mergeţi la rubrica Create New (empty) image din meniul Image. Creaţi o imagine de 3×3 pixeli şi cu 24BPP.[3] Salvaţi fişierul nou creat dându-i o denumire urmată de extensia `bmp`.

Deschideţi fişierul nou creat cu Vim şi treceţi la editarea în hex. Modificaţi fişierul după modelul de mai jos, schimbând doar ultimele trei rânduri, fără a altera ultimele patru cifre hex. Treceţi din nou în modul normal, salvaţi şi închideţi.

```
424d 5a00 0000 0000 0000 3600 0000 2800
0000 0300 0000 0300 0000 0100 1800 0000
0000 2400 0000 9c0e 0000 9c0e 0000 0000
0000 0000 0000 ffff ff00 0000 ffff ff00
0000 0000 00ff ffff 0000 0000 0000 ffff
ff00 0000 ffff ff00 0000 0d0a
```



Atenţie! Manipularea efectuată nu va scăpa neobservată cuiva competent (unele caracteristici ale imaginii, specificate pe rândul al treilea, nu mai corespund), dar fişierul poate fi vizualizat cu IrfanView. Imaginea, mărită suficient cu ajutorul IrfanView,[4] arată ca un fragment dintr-o tablă de şah, cu pătrate albe şi pătrate negre. Pătratele albe au culoarea specificată cu ajutorul şirurilor `ffffff`.

Schimbaţi un grup `ffffff` în `ff0000`. Nu uitaţi să treceţi din hex în modul normal Vim şi să salvaţi schimbarea. Pixelul a devenit

[2] Programul IrfanView poate fi descărcat gratuit de la adresa de Internet <`http://www.irfanview.com`> şi este un program de vizualizare de imagini foarte popular.

[3] Practic, 24BPP înseamnă că fiecărui pixel îi corespund patru perechi de două cifre hex. Primele trei perechi sunt folosite pentru informaţii despre culorile albastru, verde şi roşu.

[4] Apăsaţi tasta F.

albastru în programul de vizualizare, după ce aţi reîmprospătat[5] imaginea. Schimbaţi pe `ffffff` în `00ff00` şi pixelul devine verde. Schimbaţi pe `ffffff` în `0000ff` şi pixelul devine roşu.

Fişierele create în modul descris mai sus, conţin o hartă a pixelilor. Nu este de mirare că, atunci când mărim o astfel de imagine, lucrurile arată precum în figura 3.1.

Figura 3.1: O imagine care foloseşte o hartă a pixelilor mărită

Atunci când schimbăm dimensiunile unei imagini bazate pe o hartă a pixelilor, în mod inevitabil programul de vizualizare n-are informaţii suficiente decât pentru nişte pete de culoare mai mari. Imaginea capătă un aer de pictură impresionistă.

3.1.1 Programul Gimp

Vim este un excelent editor de fişiere de tip text, dar n-ar putea fi folosit pentru editarea de imagini. Într-un fel, exemplul cu care am început secţiunea §3.1 ne arată de ce acest lucru este adevărat.

Programul pe care l-aş recomanda în mod deosebit pentru prelucrarea hărţilor de pixeli este Gimp.[6] Gimp poate deschide inclusiv mica imagine creată prin manipularea codurilor hex (vezi pagina 160, rândul 20).

Posibilităţile Gimp sunt numeroase, dar descrierea lor nu constituie obiectivul anexei de faţă. Multe dintre acţiunile pe care le puteţi întreprinde cu Gimp sunt evidente din denumirea meniurilor contextuale ale ferestrelor programului.

O singură observaţie ar fi de făcut aici. Ar fi inutil să încercaţi să schimbaţi o imagine de 3×3 pixeli cu pensulele Gimp, oricât

[5] În cazul Vim şi IrfanView trebuie să închideţi şi să redeschideţi fişierele.

[6] Vizitaţi <`http://www.gimp.org`> pentru a afla cum puteţi descărca versiunea Gimp pentru Windows. Sub Linux, Gimp este programul standard de prelucrare a imaginilor şi îl conţine orice distribuţie de Linux. Gimp este o prescurtare de la „GNU image manipulation program“. A fost creat de Spencer Kimball şi Peter Mattis. Ca orice program GNU, cu sursă deschisă, include contribuţii a numeroşi programatori. Chiar şi versiunea sub Windows este uimitor de stabilă şi de eficientă.

de mici le-aţi face. Selectaţi zona (practic pixelii) pe care vreţi să o modificaţi şi umpleţi-o cu culoarea dorită.

3.1.2 Limbajul PostScript

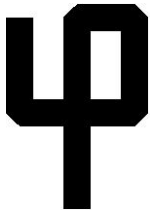
PostScript este un limbaj creat de către firma Adobe. A fost iniţial proiectat ca limbaj special pentru printere. Variantele ulterioare au adăugat elementele necesare pentru lucrul şi din perspectiva ecranului.

Sursele scrise în PostScript sunt fişiere de tip text. Vim este perfect capabil să creeze fişiere PostScript. Aceste fişiere au extensia `ps`. Pentru interpretarea fişierelor PostScript este nevoie de un program precum Ghostscript.

Pentru a vă da seama care este diferenţa dintre grafica PostScript şi cea bazată pe hărţi ale pixelilor, cel mai simplu lucru este să experimentaţi puţin cu un fişier PostScript. Creaţi un fişier `phi.ps` cu următorul conţinut:[7]

```
%!PS
%%BoundingBox: 18 28 124 180
/cm { 28.35 mul } def % definiţia centimetrului
%%EndProlog
1 cm 6 cm moveto      % punctul din care va fi desenată figura
1 cm 3.5 cm lineto    % marcarea unei căi
4 cm 3.5 cm lineto    % marcarea mijlocului lui phi
4 cm 6 cm lineto      % începem bucla lui phi
2.5 cm 6 cm lineto    % ne întoarcem către mijloc
2.5 cm 1 cm lineto    % ultima cale marcată
0.7 cm setlinewidth   % grosimea liniei pe calea trasată
2 setlinejoin         % rotunjirea colţurilor
stroke                % punerea de cerneală pe calea trasată
showpage              % afişarea paginii
```

Rezultatul, vizualizat cu Ghostscript este o versiune stilizată a literei greceşti phi.



[7] Drept ghid pentru limbajul PostScript am folosit McGilton şi Campione[5].

Persoanele familiarizate cu notaţia poloneză din logica simbolică pot înţelege mai uşor sintaxa PostScript. Limbajul creat de către firma Adobe foloseşte notaţia poloneză inversă.[8]

Primul rând din orice program PostScript începe cu `%!PS`. Sar peste cel de al doilea rând. Voi reveni la el mai târziu. Al treilea rând conţine o definiţie a centimetrului. Definiţia este necesară pentru că PostScript foloseşte ca unică unitate de măsură punctele PostScript. 267 de puncte PostScript sunt egale cu aproximativ 268 de puncte tipografice.

Rândurile 5-10 cuprind comenzile de trasare a unei căi. Ele nu desenează nimic. Dacă puneţi semnul comentariului `%` în faţa penultimului rând, veţi vedea că nu se desenează absolut nimic. Ideea de bază este că se trasează o cale şi apoi se aşterne cerneala pe calea respectivă. Restul se poate deduce din comentariile incluse în program.

Măriţi imaginea literei phi. Nu se produce nici un fel de spargere în puncte a literei. Ea este perfect scalabilă.

Explicaţia de principiu a fenomenului de mai sus este simplă. Dacă imaginea se măreşte, programul retrasează calea, în funcţie de procentul cu care cresc dimensiunile imaginii. Cerneala este adăugată abia ulterior. Interpretorul de PostScript nu depinde în acest caz de datele privitoare la pixeli. Grafica de acest tip se numeşte „grafică vectorială“ şi contrastează cu „bitmap“-purile (cu hărţile de pixeli).

Acum putem explica şi cel de al doilea rând din programul PostScript. Acest rând declară o ramă, o cutie în jurul imaginii. Este aproape inutil să spun că această cutie din jurul imaginii joacă un rol deosebit de important în integrarea imaginii PostScript în LaTeX.

Ghostview poate genera automat cutia. Ştergeţi rândul 2 din `phi.txt` şi deschideţi fişierul cu Ghostview. Creaţi un fişier PostScript încapsulat cu ajutorul dialogului PS to EPS din meniul File. Salvaţi-l cu numele `phi.eps` şi citiţi-l ca text cu ajutorul Vim. Veţi găsi cutia creată în mod automat. O puteţi vizualiza în Ghostviev cu ajutorul opţiunii Show Bounding Box.

3.1.2.1 Prelucrarea fişierelor PostScript în Gimp

Gimp este preponderent orientat către grafica hărţilor de pixeli. Este însă perfect capabil să deschidă un fişier PostScript. Trebuie însă să

[8] A se vedea şi limbajul `bst` în secţiunea 2.2.3.

aveţi instalat Ghostscript pe computer. Gimp foloseşte Ghostscript pentru a lucra cu fişiere PostScript.

Gimp poate, de pildă, deschide fără probleme fişierul `phi.ps`, cu sau fără cutia delimitatoare.

Fiţi atente şi atenţi la faptul că Gimp, dacă modificaţi fişierul PostScript, se va folosi de posibilitatea de a construi hărţi de pixeli în PostScript! Pe lângă faptul că fişierul va fi de needitat cu mâna, imaginea va avea probleme la scalare ca orice `bitmap`.

Dacă aveţi deja creat un fişier PostScript, lucrul cel mai bun pe care-l poate face Gimp este să-l transforme într-un fişier `jpg`.

Ciclul optim de lucru cu Gimp este în trei timpi. În primul timp, creaţi un fişier `xcf` (formatul nativ al Gimp) şi prelucraţi acest fişier. În al doilea timp, transformaţi fişierul `xcf` într-un fişier PostScript cu *BoundingBox*. În al treilea timp, transformaţi acest fişier într-un fişier `jpg`.[9]

În sfârşit, nu uitaţi că trecerea de la fişierul `jpg` la cel `xcf` nu este, de fapt, posibilă. Fişierul `jpg` comprimă imaginea creată. Faceţi toate prelucrările în formatul nativ Gimp.

3.2 Inserarea imaginilor în LaTeX

Cutiile imaginilor PostScript sunt inserate de către TeX printre cutiile proprii. Rezultatul este de o calitate deosebit de bună şi este foarte uşor de obţinut.

Voi descrie două cazuri: inserarea simplă a unei imagini PostScript şi transformarea ei într-un corp flotant, la care ne putem referi cu ajutorul mecanismului trimiterilor din LaTeX.

Litera phi stilizată este inserată în textul de faţă cu ajutorul următoarelor rânduri din sursa LaTeX:

```
1 ...
2 \usepackage{graphicx}
3 ...
4 \begin{center}
5 \includegraphics[scale=1]{./imagini/phi.ps}
6 \end{center}
```

Mediul `center` n-are decât funcţia pe care o sugerează numele său, aceea de a centra imaginea în raport cu textul. Dacă aţi creat fişierul `phi.ps` îl puteţi insera într-un document LaTeX şi puteţi experimenta schimbând valoarea opţiunii `scale`.

[9] Pentru a înţelege utilizarea fişierelor `jpg` vedeţi §3.2.1.

Cealaltă metodă este ilustrată cu ajutorul codului LaTeX folosit pentru a insera figura 3.1:

```
\begin{figure}[ht]
\includegraphics[width=.98\textwidth]{./imagini/bitmap.ps}
\caption{O imagine care foloseşte o hartă a pixelilor mărită}
\label{puncteBitmap}
\end{figure}
```

Mediul `figure` joacă acelaşi rol pentru imagini ca şi mediul `table` pentru tabele. Observaţi succesiunea comenzilor din rândurile 2-4. Am folosit opţiunea care indică direct lăţimea (ca procent din lăţimea textului).

Trimiterile la figuri se pot face folosind comanda `\ref{}` şi eticheta definită în mediul `figure`.

Mediul `figure` creează un obiect care pluteşte prin text. Opţiunile sale `h` şi `t` îi spun LaTeX să încerce fie o plasare chiar în punctul respectiv din text, fie în partea de sus a paginii.

În cazul eseurilor cu caracter academic aş pleda pentru acest mod sobru de inserare a figurilor în text. Mediul `wrapfig` permite îngroparea imaginilor în text, dar orice rescriere a textului dă peste cap toată munca de potrivire a textului în jurul figurii. Este o soluţie pentru tehnoredactarea unui text care nu mai suferă schimbări. Se ciocneşte de ideea de bază a elaborării textului academic: rescrierea repetată a textului.

Dacă vreţi să aveţi text alături de o imagine, fără prea multe complicaţii, puteţi folosi mediul `minipage`. Cu ajutorul lui secţionaţi pagina şi plasaţi într-o parte text, în alta imaginea.[10]

3.2.1 Inserarea de imagini în fişiere `pdf`

Dacă PostScript este un limbaj orientat în special către tipărirea pe hârtie, ruda sa PDF este un limbaj orientat către prodecerea de cărţi electronice. Un fişier `pdf` este aidoma unui text tipărit, dar este destinat lecturii pe un ecran de computer.

Limbajul PDF are propriile sale comenzi şi, folosind compilatorul `pdflatex.exe`, puteţi avea acces la ele. Există, de asemenea, o serie de pachete LaTeX şi opţiuni orientate către producerea de fişiere `pdf`. Pachetul `graphics` trebuie apelat astfel:

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

[10] A se vedea aici efectul acestei soluţii la pagina 127, rândul 5.

Un fişier `pdf` are două mari avantaje: este portabil (poate fi văzut atât sub Unix, cât şi sub Windows); este compact şi uşor de trimis sau de descărcat pe Internet. Cu alte cuvinte, este o superbă carte electronică.

Puteţi converti direct un fişier `ps` într-unul `pdf` folosind Ghostscript. Recomandabil ar fi să folosiţi opţiunea LaTeX => PS => PDF la compilare, disponibilă în versiunea 1 beta 6.20 a TEXnicCenter.

Dezavantajul convertirii descrise mai sus este acela că n-aveţi acces la compilatorul `pdflatex`. Limbajul PDF are specificul său, în raport cu PostScript, şi este firesc să fie aşa: o carte electronică se citeşte altfel decât una tipărită.

Există însă o problemă cu compilatorul `pdflatex`: nu cunoaşte limbajul PostScript. Dacă aveţi imagini PostScript, veţi primi la compilare un mesaj de eroare: *unknown graphics extension*.

Există două soluţii. Prima constă în convertirea separată a fişierelor PostScript în fişiere `pdf` şi includerea lor ca `pdf`-uri. A doua constă în folosirea unui fişier `jpg`.

fişiere `jpg`

Formatul `jpg` este de tipul *bitmap* (harta pixelilor).[11] Nu rezistă la scalare. Este, în schimb, un fişier comprimat. Fişierele `bitmap` pure tind să fie foarte mari. Fişierele `jpg` sunt de mici dimensiuni. Au şi marele avantaj de a putea fi integrate în fişierul `dvi`, dacă faceţi o manevră foarte simplă.

De exemplu, pentru a integra în `dvi` fişierul `phi.jpg`, trebuie creat un fişier de tip text `phi.bb` (acelaşi nume, dar extensia `bb`). Fişierul `bb` cuprinde un singur rând, o copie exactă a rândului din fişierul `ps` sau `eps` în care este declarată cutia, *BoundingBox*-ul.

Fişierul `jpg` poate fi integrat apoi şi-n fişierul `pdf` final prin compilare cu `pdflatex`. Sfatul meu este să folosiţi fişiere `jpg`. Oricât ar putea părea de curios, aceste fişiere sunt mai uşor de integrat în documentul final în limbaj PDF decât fişierele `pdf`. De asemenea, ele permit lucrul cu fişierul `dvi`. Este absolut nerecomandabil să lucraţi pe parcurs cu fişierul `pdf`.

Realizarea fişierului `pdf` este o operaţiune similară cu cea de tipărire pe un printer. Diferenţa constă doar în faptul că se foloseşte o „cerneală electronică“ aplicată pe o foaie de „hârtie electronică“. Nu confundaţi sub nici o formă un fişier `pdf` cu unul de tip text sau cu un fişier `dvi`, care vă permite să regăsiţi exact rîndul din sursa LaTeX care a generat un anumit efect pe ecran.

[11] Puteţi crea fişiere `jpg` cu Gimp. A se vedea §3.2.1.

3.3 Inserarea literelor ca inserare de imagini

TEX este specializat în plasarea de cutii pe foaia de hârtie. Pentru ca să poată pune însă litere şi simboluri în cutii trebuie să colaboreze cu METAFONT. Din această colaborare rezultă forţa nebănuită a sistemului TEX.

Ca şi TEX, METAFONT lucrează în fundal, ascuns în spatele perdelei de macro-uri LATEX.

3.3.1 Desenarea unei litere

Limbajul METAFONT este orientat către *descrierea* la nivel *meta* a tipurilor de litere. Probabil că prea puţine cititoare sau cititori se vor aventura prin hăţişurile METAFONT. Prezentarea noastră este, în orice caz, pentru cei care vor să ştie câteva chestiuni de principiu.

Ca deobicei, ideea noastră este că modul cel mai bun de a învăţa (chiar şi principiile în materie de programare) este meşterirea unui program şi experimentarea. Creaţi cu Vim un fişier `phi.mf` cu următorul conţinut:

```
u#:=1/2pt#;
define_pixels(u);
beginchar(70,12u#,12u#,6u#);"Litera phi";
   x1=x2=u; x3=x4=11u;
   y1=y4=y5=7u; y2=y3=u;
   x5=x6=6u;
   bot y6=-5u;
   pickup pencircle scaled 0.5pt;
   draw z1..z2;
   draw z2..z3;
   draw z3..z4;
   draw z4..z5;
   draw z5..z6;
   labels(range 1 thru 6);
endchar;
end
```

Ar fi destul de greu să ne dăm seama ce semnificaţie are programul `phi.mf` fără a genera o imagine. Deschideţi cu 2xExplorer o fereastră MS-DOS pentru dosarul în care se află `phi.mf` şi daţi două comenzi:

```
mf phi
gftodvi phi.2602gf
```

Prima comandă compilează fişierul. A doua comandă creează un fişier `dvi`.

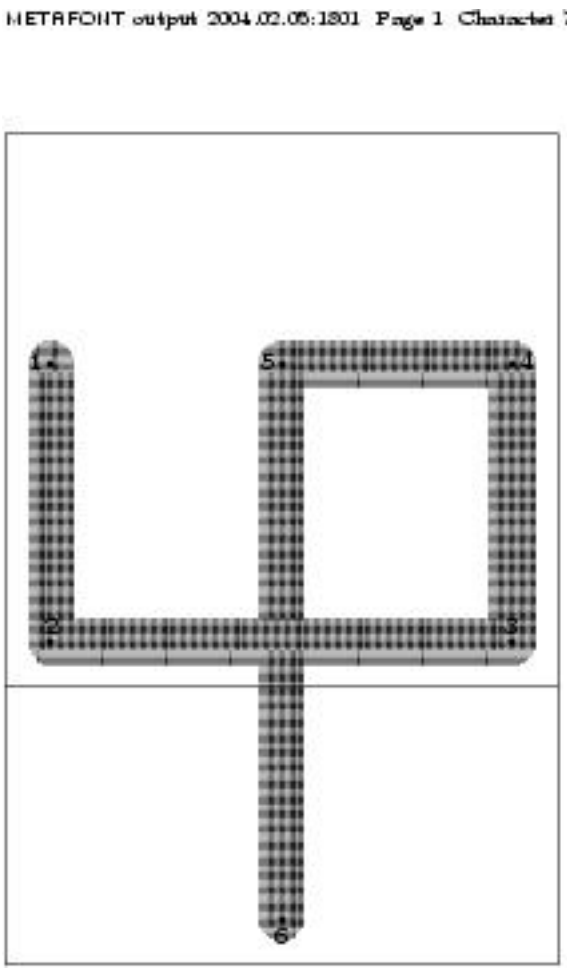


Figura 3.2: Imagine de control pentru programe scrise în METAFONT

Vizualizarea cu Yap a imaginii 3.2 nu este o chestiune de rutină. Dacă aveţi un computer cu suficient de multă memorie, căutaţi în dosarul `misc` din `\texmf\fonts\source\public` fişierul `gray.mf` şi aduceţi singurul său rând la forma următoare:

```
input graylj % the 'standard' gray font is for the CX
```

Deschideţi Yap şi daţi un clic pe Options... din meniul View şi-n panoul Display alegeţi în caseta Mode un printer de tip LaserJet cu 600dpi.[12] Acum ar trebui să vedeţi bine imaginea în Yap.

În orice caz, uitaţi-vă la Document Properties... din meniul File din Yap. Examinaţi tipurile de litere (*fonts*) încărcate şi vedeţi unde se găsesc pe computer. Citiţi eventualele mesaje de eroare şi soluţiile adoptate automat de către Yap.

Acum putem comenta, cu imaginea alături, conţinutul fişierului `phi.mf`. Primul rând permite mărirea sau micşorarea desenului. Rândurile 4–7 conţin descrierea unor puncte din desenul literei. Numele punctelor sunt de forma `z_`, unde în locul spaţiului subliniat este pusă o cifră. Rândurile 9–13 conţin descrierea liniilor. Rândul 8 inidică grosimea liniilor.

Rândul 3 are o importanţă aparte. În paranteza rotundă sunt patru valori. Prima este codul literei. Următoarele două valori indică poziţia colţului din dreapta sus al cutiei literei (*bounding box*-ului). Ultima valoare indică poziţionarea liniei de bază a literei. Cutiile cu litere sunt înşiruite de către TEX în aşa fel încât liniile de bază se continuă, pe orizontală, una pe alta.

Limbajul METAFONT este unul declarativ, descriptiv. Sistemului nu i se spune ce acţiuni să întreprindă, ci cum arată litera.

Structura programului este extrem de simplă. Pentru a vedea efectele schimbării descrierii literei, modificaţi ceva din program şi vedeţi ce se întâmplă. S-ar putea să nu puteţi compila deloc. Tastaţi `x` şi-ncercaţi să scăpaţi. Nu efectuaţi însă decât o singură modificare o dată, pentru a localiza efectele.

Litera noastră este în mod intenţionat construită din linii drepte. Rotunjirile sunt posibile însă. Scrieţi, în locul `draw z3..z4` şi `draw z4..z5`, o declaraţie `draw z3..z4..z5` şi veţi descoperi ce deştept este METAFONT.

[12] 600dpi este o măsură a rezoluţiei printerului (*dots per inch*). Cum minusculele pete (*dots*) de cerneală seamănă cu pixelii, măsura este bună şi pentru ecran.

3.3.2 Inserarea literei în text

Pentru a ne folosi efectiv de o literă desenată trebuie să pregătim fişierele de care se foloseşte TEX. Trebuie să ne întoarcem în fereastra MS-DOS şi să dăm comenzile:

```
mf \mode=ljfour; mode_setup; input phi.mf
gftopk phi.600gf phi.pk
```

Fişierele `phi.pk` şi `phi.tfm` sunt cele care reprezintă descrierea literei în formatul de care se foloseşte TEX. Acum pot scrie despre:

un ⊎ cu colţuri.

Cum am procedat? Am pus fişierele de tip `mf` `pk` şi `tfm` în dosarul proiectului cărţii[13] şi am spus sistemului să recunoască un nou tip de literă şi să-l folosească:

```
\newfont{\literaPhi}{phi}
\newcommand{\filos}{{\literaPhi F}}
un \filos{} cu colţuri.
```

Prezentarea de mai sus este una strict informativă. Litera construită n-are o valoare practică. În orice caz, valoarea ei este foarte limitată. Pot rescrie formula preluată din cartea lui Adrian Miroiu:[14]

$$\vdash_{\mathrm{LK}} ⊎ \text{ ddacă } \models_{\mathrm{LK}} ⊎$$

Am folosit însă o cale ocolită. Pentru a insera, în modul matematic, simbolul ⊎ l-am apelat în argumentul unei comenzi `\text{}`.

Dac-aţi meşterit însă variante ale fişierului `phi.mf` vă daţi mult mai bine seama care este sensul mesajelor pe care le primiţi la compilarea în TEXnicCenter. Acum este posibil să identificaţi mai uşor partea care vine de la METAFONT şi cea care provine de la TEX.

Un lucru important pe care trebuie să-l ştiţi este că METAFONT creează *bitmap*-uri. Micile desene care sunt literele sunt inserate în document ca imagini de tip hartă de pixeli. Hărţile acestea pot fi însă modificate în funcţie de dimensiunea literelor. Puteţi primi însă mesaje de genul „nu găsesc litera la dimensiunea cerută“ sau să aveţi dificultăţi cu documentele `pdf` (litere neclare pe ecran) dacă folosiţi doar nucleul LATEX.[15]

[13] Yap compilează automat sursele literelor; lipsa fişierului de tip `mf` poate duce la erori.

[14] Vezi aici pagina 149.

[15] Mai multe informaţii şi exemple pe tema literelor şi simbolurilor folosite în LATEX ne propunem să le plasăm, cu timpul, pe situl cărţii.

Capitolul 4

Translatorii

Cuprins

4.1 Drumul către HTML **172**
4.1.1 De la LaTeX la HTML cu ajutorul HeVeA 174
4.1.2 De la LaTeX la HTML cu ajutorul TeX4ht 176
4.1.3 De la LaTeX la HTML cu LaTeX2HTML . . 177
4.2 Drumul către RTF **178**
4.3 Înapoi către LaTeX **179**

LaTeX este un limbaj pentru cei care au nostalgia cărţilor frumos tipărite. Este, de asemenea, limbajul ideal pentru elaborarea scrierilor cu caracter academic. În lumea de azi există însă multe alte limbaje pentru documentele electronice.

La începutul celui de al treilea mileniu, Internetul găzduieşte deja o bibliotecă planetară. Limbajul său de bază este HTML. Editoarele de birou au propriile lor limbaje, cel mai adesea ascunse de privirea persoanelor care le utilizează, dar fără de care n-ar fi posibilă afişarea pe ecran şi tipărirea textelor produse cu ajutorul lor.

Există deci o problemă a integrării documentelor LaTeX în acest peisaj variat şi, în funcţie de necesităţi, a traducerii lor în alte limbaje.

Trebuie distins însă între două situaţii: traducerea părţii care este în mod text şi traducerea părţii care este în mod matematic. În ambele cazuri, traducerea este dificilă. Deosebirea rezidă însă în faptul că, în primul caz, traducerea este – în linii mari – posibilă. În al doilea caz, traducerea este foarte greu de făcut.

Recomandarea noastră, pentru seminarii şi examene, este aceea de a tipări documentele LaTeX. La urma urmei, pentru asta sunt făcute.

Tipărirea poate fi pe hârtie sau electronică. În cazul tipăririi electronice, formatul `pdf` este cel la care trebuie ajuns, dacă este posibil. Utilizarea pachetului `hyperref`, creat de către Sebastian Rahtz, este, de asemenea, absolut recomandabilă în cazul tipăririi electronice.[1]

Date fiind scopurile cărţii de faţă, prezentarea în detalii a traducerii din LaTeX în alte limbaje n-are sens. Obiectivul nostru principal îl reprezintă producerea unor texte tipărite. Această anexă oferă doar câteva informaţii cu privire la posibilităţile şi tehnicile de traducere, fără nici un fel de indicaţii cu caracter practic. Pentru instalarea, utilizarea şi modificarea programelor la care ne vom referi trebuie să consultaţi documentaţia lor.

Nu uitaţi niciodată să efectuaţi fiecare traducere într-un dosar separat. Pentru a obţine rezultate mai bune sursele LaTeX trebuie pregătite în mod special în vederea traducerii, conform indicaţiilor din documentaţia relevantă.

4.1 Drumul către HTML

Documentele LaTeX, tipărite electronic în fişiere `ps` sau `pdf` sunt uşor de vehiculat pe Internet. Aşa cum am arătat mai sus, formatul `pdf` este preferabil.

Limbajul HTML este însă limbajul de bază al Internet-ului. Dacă vrem să prezentăm eseurile noastre pe Internet, trebuie să ştim câte ceva despre HTML.

SGML

Puţină lume ştie că HTML este o aplicaţie, cu posibilităţi limitate, a SGML. Acronimul SGML vine de la *Standard Generalized Markup Language*.[2] Raţiunea existenţei SGML poate fi mai lesne înţeleasă dacă vom reaminti modul de lucru tradiţional într-o editură.

Redactorii şi tehnoredactorii unei edituri scriau pe dactilograma textului care urma să fie tipărit o serie întreagă de indicaţii. Unele dintre ele priveau tipul de literă ce urma să fie folosit în tipografie. Altele priveau poziţia unei porţiuni din text în raport cu restul textului şi aşa mai departe. În limba engleză, toate aceste operaţii sunt desemnate cu termenul *mark up*.

[1] A se vedea explicaţiile lui Sebastian Rahtz, în *Hypertext marks in LaTeX*, fişierul `hyperref.dvi` din `\texmf\doc\latex\hyperref`, precum şi restul documentaţiei din dosarul respectiv. Nu încercaţi să înţelegeţi de la bun început subtilităţile tehnice ale pachetului; simpla lui includere în document şi câteva minime adaptări ale documentului vor produce efecte spectaculoase.

[2] Bradley[2, p.4] arată că SGML a devenit un standard în 1986.

Ideea de bază a SGML este simplă. Porţiunile de text în care urmează să se aplice o anumită indicaţie sunt puse într-o paranteză construită în felul următor:

<tag>porţiunea de text aflată între etichete</tag>

Fiecare tag are un nume. Se observă mai sus cum începutul porţiunii etichetate este marcat de numele tag-ului pus între paranteze ascuţite, iar finalul ei este semnalat prin bara oblică pusă înaintea tag-ului.

Numele tag-urilor, precum şi indicaţiile de prelucrare a porţiunilor de text plasate sub cupola unui tag sunt date separat în *Document Type Definition*, pe scurt DTD. **DTD**

Pentru ca perechea formată dintr-un text în care sunt incluse indicaţii în SGML şi un DTD să conducă şi la efecte concrete pe ecranul unui computer este nevoie şi de un analizor (*parser*). Analizorul detectează construcţiile SGML din text şi le prelucrează conform instrucţiunilor din DTD. Analizorul poate să fie, de pildă, un procesor de cuvinte sau un program de vizualizare a textelor structurate conform SGML.

Spre deosebire de TEX, SGML nu este un limbaj de programare. Este un limbaj destinat identificării *obiectelor* dintr-un text. Printre aceste obiecte se poate afla însă şi o porţiune de cod TEX[3] sau o secvenţă scrisă în alt limbaj de programare.

Mecanismul trimiterilor electronice care pot fi definite în SGML este cunoscut sub numele de „hipertext“. Hipertextul nu este altceva decât o dezvoltare a ideii tradiţionale de trimitere. În mod tradiţional, dacă o notă conţinea o referire la o carte, ca să vezi efectiv pasajul din carte la care se făcea trimitere te deplasai la bibliotecă, cereai cartea şi citeai fragmentul cu pricina. Prin contrast, trimiterea electronică nu indică doar locul unde se poate citi ceva, ci chiar îţi aduce pe ecran textul la care se face trimitere. **hipertextul**

HTML este o aplicaţie a limbajului SGML. Deschideţi cu *Vim* un fişier de tip html şi veţi remarca imediat tag-urile, scoase în evidenţă cu ajutorul culorilor. Studiaţi, de asemenea, primul rând al fişierului şi ar trebui să vedeţi ceva de genul:

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

Observaţi precizarea privitoare la DTD. Chiar dacă ea lipseşte, vizualizatorul de HTML se descurcă. Analizoarele de HTML nu sunt

[3] Bradley[2, p.72] explică felul în care poate fi identificată porţiunea scrisă în TEX.

nici pe departe aşa de pretenţioase cu textul pe care-l au în faţă ca TEX.

Capacităţile HTML sunt destul de limitate, dar în ultima vreme există un efort considerabil de a dezvolta XML, în încercarea de a redescoperi întreaga putere a SGML.

Din descrierea sumară de mai sus se vede că LATEX şi HTML au şi o serie de puncte comune. Sunt, în orice caz, perfect compatibile.

Soluţia mea pentru un sit academic ar fi o serie de pagini HTML interconectate cu ajutorul mecanismului hipertextului. În ele se găsesc trimiteri electronice la fişiere `pdf` sau `ps`.[4] O parte dintre documentele LATEX sunt traduse în HTML; unele bibliografii sunt oferite şi împreună cu fişierul `bib` aferent.

De ce să nu scriem direct în HTML? În cazul eseurilor academice, LATEX creează automat bibliografia, notele de subsol, indicii. HTML n-are resursele unui limbaj de programare şi tot aparatul auxiliar academic ar trebui creat manual. Evident, este posibil să recurgem la alt limbaj de programare care să completeze cumva capacităţile HTML (să ţină, de pildă, socoteala numerelor notelor de subsol); LATEX este însă soluţia superioară la acest capitol.

4.1.1 De la LATEX la HTML cu ajutorul HEVEA

HEVEA este un program creat de către Luc Maranget. Philip A. Viton a realizat o versiune a HEVEA sub Windows.[5]

Instalarea HEVEA sub Windows se face fără nici o dificultate, dacă urmaţi instrucţiunile lui Philip A. Viton.

Integrarea în TEXnicCenter nu este nici ea dificilă. Imitaţi profilul folosit la crearea unui fişier `ps`, dar cu `hevea.exe` ca postprocesor şi cu Internet Explorer ca vizualizator.

Rezultatele obţinute cu HEVEA sunt bune şi, la o sursă LATEX simplă, aproape că n-aveţi ce schimba în fişierul `html`. La textele în limba română aveţi grijă ca următorul `tag` din fişierul `html` să aibă

charset atributele corecte:

```
<META http-equiv="Content-Type" content="text/html;
                                          charset=windows-1250">
```

Valoarea lui `charset` poate să fie şi `ISO-8859-2`; `windows-1250` pare

[4] Pentru strategia integrării documentelor `pdf` în reţeaua de informaţii de pe web v. McKinley[6].

[5] Pentru a porni în căutarea HEVEA mergeţi la pagina de Internet a lui Philip A.Viton la <`http://www.arch.ohio-state.edu/crp/faculty/pviton/`>.

mai naturală aici, dacă documentul LaTeX foloseşte `cp1250`; fără aceste valori ale lui `charset` nu se văd literele româneşti. Nu este recomandabil să puneţi `utf-8` ca valoare a `charset`. Pentru limbile europene n-aveţi nevoie de întreaga putere a unicod. Pagina dumneavoastră se va încărca lent din pricina dimensiunilor fişierului cu simboluri unicod.

Dacă doriţi să folosiţi entităţi HTML, puteţi încerca transformarea fişierului de tip `html` cu ajutorul unui script Vim ca acesta:

```
:%sno/ă/&atilde;/ge | update
:%sno/Ă/&Atilde;/ge | update
:%sno/â/&acirc;/ge | update
:%sno/Â/&Acirc;/ge | update
:%sno/î/&icirc;/ge | update
:%sno/Î/&Icirc;/ge | update
:%sno/ş/&scedil;/ge | update
:%sno/Ş/&Scedil;/ge | update
:%sno/ţ/&tcedil;/ge | update
:%sno/Ţ/&Tcedil;/ge | update
```

Puteţi pune şi coduri numerice, după moda XML, prin substituiri ca acelea de mai jos:

```
:%sno/&atilde;/&#259;/ge | update
:%sno/&Atilde;/&#258;/ge | update
:%sno/&acirc;/&#226;/ge | update
:%sno/&Acirc;/&#194;/ge | update
:%sno/&icirc;/&#238;/ge | update
:%sno/&Icirc;/&#206;/ge | update
:%sno/&scedil;/&#351;/ge | update
:%sno/&Scedil;/&#350;/ge | update
:%sno/&tcedil;/&#355;/ge | update
:%sno/&Tcedil;/&#354;/ge | update
```

4.1.1.1 Pregătirea documentului pentru Internet folosind HaCha

Documentele HTML, oricât de lungi ar fi, sunt derulate sub forma unei pagini unice pe ecranul computerului. Acest lucru are două dezavantaje: când conexiunea Internet este slabă, încărcarea paginii se face anevoios; de asemenea, este dificil să urmăreşti o pagină care se derulează foarte mult.

Ideal ar fi ca pagina să ocupe aproximativ un ecran. În orice caz, plasarea secţiunilor unui document în fişiere `html` diferite este dezirabilă.

Programul care vă ajută să faceţi secţionările descrise mai sus este HACHA. Acest program trebuie folosit după ce aţi prelucrat documentul cu ajutorul HEVEA.

4.1.1.2 HEVEA şi imaginile din documentele HTML

Programul HEVEA a fost creat de către Luc Maranget pentru a fi folosit sub Unix. Sub Unix el prelucrează automat imaginile la care face apel sursa LaTeX.

Sub Windows, fără uneltele Unix, trebuie să pregătiţi manual imaginile pentru Internet. Veţi găsi totuşi, în fişierul `html`, create de către HEVEA, locurile în care se pot integra imaginile.

Dacă aţi instalat unelte Unix şi aveţi la dispoziţie *shell*-ul Unix, atunci trebuie să rescrieţi scriptul `imagen` în aşa fel încât să poată rula sub Windows. În aceste condiţii, puteţi include imagini în documente HTML şi sub Windows.[6]

4.1.2 De la LaTeX la HTML cu ajutorul TeX4ht

Distribuţia MikTeX include un translator de `html`. Dacă nu i-aţi sesizat deja prezenţa, mergeţi în dosarul `\texmf\tex4ht`. Acolo veţi găsi o parte din elementele pachetului creat de către Eitan Gurari pentru a traduce din LaTeX în HTML.

Dacă citiţi descrierea pachetului TeX4ht, veţi vedea că este *configurabil*. Luaţi în serios diferenţa dintre *configurat* şi *configurabil*. Ca peste tot în lumea TeX, şi aici se aplică principiul „Dumnezeu dă omului, dar în traistă nu-i bagă“; aşa că trebuie să configuraţi TeX4ht. Philip A. Viton are o pagină de sprijin pentru persoanele care instalează TeX4ht în cadrul MikTeX.[7]

Dac-ar fi să dau câteva indicaţii bazate pe experienţa proprie, aş spune, în primul rând, c-am mutat fişierele din dosarul `base`, aflat pe calea `\texmf\tex4ht` în dosarul `home`. Trebuie modificat cu grijă fişierul `tex4ht.env`. Atenţie trebuie acordată şi fişierului `tex4ht.fls`; dacă nu există, trebuie creat unul de tip text cu acest nume în dosarul unde sunt plasate `texmf` şi `localtexmf`.

Pentru a vedea dac-aţi reuşit să configuraţi TeX4ht faceţi un test cu un fişier `tex`. Eu am folosit sursa manualului lui Stéphane

[6] Pentru descrierea modificărilor operate de către noi în scriptul `imagen` vizitaţi situl cărţii. În momentul de faţă, versiunea noastră converteşte, sub Windows, imaginile PostScript şi le integrează în documente HTML.

[7] Adresa paginii cu explicaţiile lui Philip A. Viton este <`http://facweb.arch.ohio-state.edu/pviton/support/tex4ht.html`>.

Bressan şi Eitan Gurari pentru pachetul TEXProject.[8] Am instalat elementele din pachetul respectiv. Am pus sursa manualului într-un dosar în care nu era alt fişier. Am adăugat în sursă comanda `\usepackage{tex4ht}` şi am dat într-o fereastră MS-DOS comanda `ht latex manual`. Fără a exagera, se poate spune că rezultatul a fost de o calitate deosebit de bună.

TEX4ht poate produce şi fişiere XML în formatul definit de către DTD-uri ale OpenOffice.

4.1.3 De la LATEX la HTML cu LATEX2HTML

LATEX2HTML a fost conceput iniţial de către Nikos Drakos. Opera sa a fost continuată de către Ross More şi alţii. Ideea lor de bază este aceea de a realiza un translator care traduce în întregime documentul LATEX: unele părţi sunt traduse direct în limbajul HTML, iar ceea ce nu se poate traduce este transformat în imagine şi inserat ca imagine în documentul HTML.

Recursul masiv la imagini are şi unele dezavantaje. Dacă descărcaţi un text convertit din LATEX în HTML cu LATEX2HTML trebuie să recurgeţi la `wget`.[9] Altfel, aproape sigur veţi rămâne cu fără unele imagini sau hipertextul va fi stricat. De asemenea, din pricina imaginilor căutarea unora dintre termeni este imposibilă sau trebuie făcută cumva indirect (folosind elemente din fişierul HTML care, în mod normal, nu se văd).

Instalarea LATEX2HTML sub Linux nu cred că mi-a luat mai mult de cinci minute. Iar programul a funcţionat fără cusur. În schimb, sub Windows, ar trebui să fiţi fericite sau fericiţi dacă duceţi la bun sfârşit instalarea.

În principiu, pentru instalare, aveţi nevoie de Perl şi de NetPBM.[10] Perl este un limbaj de programare. LATEX2HTML este un script realizat în Perl. Din pricina aceasta nu intru aici în detalii.[11]

În rezumat, programul este dificil de instalat sub Windows, poate să vă lase fără resurse de memorie şi nu este tocmai uşor de folosit. Este totuşi cel mai vestit dintre programele de traducere din LATEX în HTML şi simplul fapt de a-l putea folosi sub Windows va oferi

[8] Pachetul respectiv poate fi descărcat de pe pagina de web a lui Eitan Gurari <`http://www.cis.ohio-state.edu/~gurari/index5.html`>.

[9] A se vedea aici §1.4.2.1.

[10] A se vedea Luis Seidel Gómez de Quero, *Installing LATEX2HTML with MikTEX* <`http://www.mayer.dial.pipex.com/l2h.htm`>, 4 februarie 2000, document accesat pe data de 20/08/2003.

[11] A se vedea situl cărţii pentru detalii.

oricui o mare satisfacţie. Este recomandabilă folosirea lui în cazul transformărilor mai dificile. Este singurul dintre translatori care s-a descurcat cu anexele cărţii de faţă chiar şi fără să fi operat vreo schimbare în sursa LaTeX.

4.2 Drumul către RTF

Formatul RTF este proprietatea firmei Microsoft. A fost creat pentru a facilita transferul de texte formatate între diverse aplicaţii, sub diverse sisteme de operare. Între altele, el deschide drumul către popularul format Word al firmei Microsoft.

Dacă o editură sau o revistă vă solicită textul în format Word, iar sursa dumneavoastră este în LaTeX, aveţi la dispoziţie două soluţii. Puteţi folosi HTML ca pe o platformă de pe care să treceţi la formatul Word. Puteţi folosi în acelaşi scop RTF. Pentru aceasta vă trebuie însă un translator din LaTeX în RTF.

Cea mai bună alegere este programul LaTeX2RTF disponibil pe marele sit al programării cu sursă deschisă, SourceForge.net.[12] Începând cu 20 ianuarie 2004 există şi un program de instalare sub Windows. Dacă nu instalaţi în dosarul care vi se propune în mod automat, trebuie să efectuaţi manual unele corecturi în fişierele de tip `bat`.

Pentru a beneficia de toată puterea LaTeX2RTF trebuie să aveţi instalat Ghostscript şi ImageMagick[13].

Sintaxa RTF seamănă, sub anumite aspecte, cu sintaxa TeX.[14] Iată un fragment de cod RTF:

```
1 Aceasta este {\b o carte} foarte grea. \par
```

Porţiunea prinsă între acolade reprezintă un grup. Ca şi o instrucţiune TeX, secvenţa `\b` îi spune programului care interpretează fişierul `rtf` că grupul trebuie scris cu aldine. Alternativ, am putea folosi comenzile `\b` şi `\b0` pentru a marca începutul, respectiv sfârşitul porţiunii de text care este scrisă îngroşat.

Puteţi experimenta pe un fişier `rtf` creat cu ajutorul translatorului folosind editorul Vim. Fişierele de tip `rtf` sunt fişiere text.

[12] Adresa exactă este <`http://latex2rtf.sourceforge.net`>.

[13] ImageMagick este un program de convertire de imagini. Pentru mai multe informaţii vizitaţi situl <`http://www.ImageMagick.org`>. Identificaţi cu atenţie în programul instalat locul în care se află programul `convert.exe`.

[14] Pentru detalii consultaţi pagina <`http://latex2rtf.sourceforge.net/rtfspec.html`>.

RTF are un mod de a trata căciula de pe ă şi sedilele lui ş şi ţ care n-arată prea estetic în programul de vizualizare.[15] Eu ocolesc traducerea automată în stil rtf substituind o parte dintre literele româneşti în felul următor:

```
:%s/\\u{a}/a227/ge | update
:%s/\\u{A}/A195/ge | update
:%s/\\^{a}/â/ge | update
:%s/\\^{A}/Â/ge | update
:%s/\\^{\\i}/î/ge | update
:%s/\\^{I}/Î/ge | update
:%s/\\c{s}/s186/ge | update
:%s/\\c{S}/S170/ge | update
:%s/\\c{t}/t254/ge | update
:%s/\\c{T}/T222/ge | update
```

În documentul final, aceste substituiri sunt eliminate. Atenţie însă la acest subterfugiu dacă aveţi într-un tabel pe A195 şi nu vreţi să devină Ă.

Fişierul rtf n-ar trebui predat ca atare redacţiei care l-a solicitat. El necesită aproape întotdeauna unele prelucrări suplimentare. El este un fişier destinat transferului între aplicaţii. Redacţiei ar trebui să-i fie predat un fişier în formatul propriu aplicaţiei pe care o foloseşte.

4.3 Înapoi către LaTeX

Ar trebui spuse câteva cuvinte şi despre posibilitatea de a traduce în LaTeX formatările din alte limbaje. Problema s-ar putea să se pună practic cel mai frecvent în cazul documentelor Word.

Formatul Word al firmei Microsoft este foarte popular. Imensa majoritate a persoanelor care utilizează computerele nu au motive să investească timp şi bani în învăţarea programării (chiar într-o măsură limitată). Editorul Word al firmei Microsoft a reprezentat o reuşită deosebită în efortul de a ascunde aspectele legate, direct sau indirect, de programare. Au dispărut comenzile cu punct din editoarele mai vechi. Utilizatorii sunt scutiţi de orice contact cu marcajele

[15] Pentru a vedea un fişier rtf puteţi folosi programul de vizualizare a documentelor Word pus la dispoziţie gratuit de către Microsoft. Acest program poate citi şi fişiere rtf. Puteţi construi propriul lector de rtf folosind codul inclus de către Microsoft în specificaţia limbajului RTF. Ca şi formatul RTF, programul de vizualizare şi codul pentru lector sunt proprietatea firmei Microsoft şi pot fi folosite numai în condiţiile respectării drepturilor firmei. Citiţi desigur licenţele aferente.

introduse în text. Aceasta este raţiunea succesului produsului respectiv pe piaţă.

Trecerea de la Word la LaTeX este departe de a fi chiar atât de simplă. Există o serie de produse comerciale care încearcă să facă – fără prea mult succes – acest lucru. Eu mă voi referi aici la un program cu sursă deschisă, MSWordView.[16] MSWordView a fost conceput iniţial ca un program de traducere, sub Unix, a documentelor Word în HTML. În stadiul actual este capabil să facă o traducere, inclusiv sub Windows, în LaTeX.

Partea bună a MSWordView atunci când traduce în LaTeX este capacitatea sa de a produce o sursă LaTeX curată. Redarea literelor specifice limbii române este, de asemenea, impecabilă.

Partea proastă a lucrurilor începe atunci când este vorba despre notele de subsol. MSWordView marchează locul lor în text şi mută tot textul notelor la sfârşit. Este perfect posibil să înlocuieşti marcajul `$^{}$` introdus de translator cu comanda `\footnote{}` şi să muţi apoi textul notelor, dar munca este anevoioasă.

În rezumat, MSWordView este util dacă aveţi un text masiv, cu puţine note de subsol, pe care vreţi să-l tehnoredactaţi în LaTeX.

Cocluzia este limpede. LaTeX este un limbaj pentru elaborarea unor surse organizate logic. Este un punct de plecare, nu o ţintă de atins plecând de la alte moduri de a structura texte.

Faptul că sursa LaTeX este practic extrem de greu de reconstituit are o consecinţă de care se pot folosi persoanele care se tem că lucrările lor ar putea fi plagiate cu uşurinţă dacă sunt plasate pe Internet. Ele pot pot afişa doar fişierele care pot fi citite comod. Sursele nu sunt făcute publice. Ar fi foarte greu pentru plagiatori să pretindă că sunt adevăraţii autori. Reconstituirea absolut exactă a surselor originare este practic imposibilă. Dacă ţinem cont de faptul că sursele originare conţin în mod normal şi comentarii, reconstituirea este absolut imposibilă.

[16] Versiunea cea mai comod de instalat sub Windows este cea disponibilă în cadrul proiectului GnuWin32. A se vedea <`http://gnuwin32.sourceforge.net`>.

Bibliografie

[1] Paul A. Blaga şi Horia F. Pop. *LaTeX 2ε*. Bucureşti: Editura Tehnică, 1999. Citată la pp. 65, 150, 154 şi 156.

[2] Neil Bradley. *The concise <SGML> companion*. Harlow, Anglia: Addison-Wesley, 1996. Citată la pp. 172 şi 173.

[3] I. Funeriu. *Principii şi norme de tehnoredactare computerizată*. Timişoara: AMARCORD, 1998. Citată la p. 86.

[4] Leslie Lamport. *LaTeX: a document preparation system*. Reading, Massachusetts: Addison Wesley Longman,Inc., 1994. Citată la pp. 65, 92, 93, 108 şi 154.

[5] Henry McGilton şi Mary Campione. *PostScript by Example*. Reading, Massachusetts: Addison-Wesley, 1992. Citată la p. 162.

[6] Tony McKinley. *From Paper to Web*. San Jose, California: Adobe Press, 1997. Citată la p. 174.

[7] Bram Moolenaar. *Vim User Manual*. <`http://vimdoc.sf.net`>, 2002, data accesării: 2003/12/03. Citată la pp. 35, 36, 37, 38, 43, 45 şi 46.

[8] Costică Moroşanu. *Elemente de bază ale sistemului* TeX *şi* plain TeX. Iaşi: Editura Universităţii „Alexandru Ioan Cuza“, 2001. Citată la pp. 64 şi 65.

[9] Steve Oualline. *Vi IMproved—Vim*. <`http://www.truth.sk/vim/vimbook-OPL.pdf`>, 2001, data accesării: 2003/12/30. Citată la p. 35.

[10] Artur Pusztai şi Gheorghe Ardelean. *LaTeX: ghid de utilizare*. Bucureşti: Editura Tehnică, 1994. Citată la p. 65.

[11] Raymond Seroul. *Le petit Livre de TeX*. Paris: InterEditions, 1989. Citată la pp. 83, 84, 109 şi 110.

[12] S. Tóth. *Litera de tipar*. Bucureşti: Editura Ştiinţifică, 1966. Citată la p. 64.

Indice

§, 89

AUTOEXEC.BAT, 19, 20

betacod, 137–140
BibDB, 118
bibliografie
  cu BibTEX, 119–121
  stil, 119, 120
  stil simplu, 120
bitmap, *vezi* imagini(hărţi de pixeli)
Bozinis, Nikos, 9
Braams, Johannes L., 124

căutare, *vezi şi* expresii regulate
  pe Internet, 54–60
cale, 5
computerul şi munca intelectuală, 71
conţinut vs. formă, 113
concordanţă, 53, 54
corectura
  computerizată, 60–62
`ctags`, 102–104
cuprins
  cu LaTeX, 80, 81, 153

Denisov, Alex, 145
descriptor, 157
Doron, Eyal, 118
Drakos, Nikos, 177

expresii regulate, 46–52

fişe, 7
  bibliografice, 114, 118
fişiere
  analogia cu fişele tradiţionale, 7
  bb, 166
  bib, 116
  bmp, 160
  bst, 119
  căutarea, 11
  comprimate, 12
  crearea, 10
  csv, 114, 115
  dvi, 75, 79
  extensiile, 8
  filtre, 10
  html, 173
  jpg, 166
  `log`, 85
  management cu 2xExplorer, 9
  mf, 167
  numărarea cuvintelor, 20
  operaţii, 10
  pdf, 14, 165
  proprietăţi, 10
  ps, 13, 14, 162–164
  redenumire, 17
  rtf, 178
  `sty`, 109
  ştergere, 17
  tex, 78
  tiff, 18
  xcf, 164
formule, 99, 100, 148–155

gcc, 21
Ghostscript, 13
Gimp, 161–163
GNU, 22
GnuWin32, 21
Gonzato, Guido, 112
grep, 51
Gurari, Eitan, 176

HACHA, 175
HEVEA, 174
hieroglife, 143
HOME, 20

Ibycus, 140, 141
imagini
    hărţi de pixeli, 160, 161
    vectoriale, 162, 163
indici
    generarea lor cu MakeIndex, 157

Jacq, Christian, 143
Joy, William, 24

Knuth, Donald, 64, 66–68, 83, 123, 135

Lamport, Leslie, 65
LaTable, 145
LATEX, 65–67
    #, 105
    alineat, 85
    antet, 76
    bad box(es), 96
    \bcode{}, 137
    {\bfseries }, 93
    blocurile, 92
    \cite{}, 121
    comentarii, 86
    compilare, 78
    contoare, 107
    corpul programului, 79
    cp1250, 127–130
    declaraţiile, 92
    diacritice, 124
    documentaţie, 112
    {\em }, 93
    \fontfamily{}, 134
    \fontsize{}, 134
    \footnote{}, 94
    ghilimele, 91
    \hspace{}, 96
    hyperref, 172
    ifthen, 108
    \index{}, 157
    {\itshape }, 93
    \label{}, 88
    \ldots{}, 91
    letrine, 109
    \marginpar{}, 94
    \mbox{}, 92
    {\mdseries }, 93
    mediile, 97
    modul matematic, 99
    modurile, 83
    \newcommand{}{}, 104
    \newenvironment, 106
    pachete, 107
    pachete proprii, 109
    \pageref{}, 101
    \par{}, 85
    \ref{}, 101
    \relax, 110
    revizia, 93
    {\rmfamily }, 93
    româna, 123
    {\scshape }, 93
    secţiuni, 87
    \selectfont{}, 134
    semne rezervate, 76, 90
    {\sffamily }, 93
    {\slshape }, 93
    structura paginii, 111
    \textbf{}, 90
    \textit{}, 90
    \textrm{}, 90

\textsc{}, 90
\textsf{}, 90
\textsl{}, 90
\today{}, 92
{\ttfamily }, 93
\underline{}, 90
unicod, 141
{\upshape }, 93
\vspace{}, 96
LaTeX2HTML, 177, 178
LaTeX2RTF, 178
Levy, Silvio, 140
litere
chirilice, 135, 136
greceşti, 136–141
limbile europene, 132, 133
româneşti, 29, 122–126, 151

MacKay, Pierre, 140
Maranget, Luc, 174
matchit, 82
meniu contextual, 4
METAFONT, 144, 167–169
MikTeX, 69, 70
More, Ross, 177
MS-DOS, 15–17
MSWordView, 180

note
de subsol, 94, 95
marginale, 95

octet, 34
Oetiker, Tobias, 112, 113
OpenOffice, 60, 177
folosirea bibliografiei, 24
Ostrum, Piet van, 113

Pakin, Scott, 151
paragraf, 87
partiţiile, 5
Patashnik, Oren, 119, 120
Pehong Chen, 156
pixel, 159
plagiatul, 180
proiect
LaTeX, 80, 82

rând
logic, 28
sfârşitul de rând, 8, 35
vizual, 28
Rahtz, Sebastian, 172

secţiune, 87
SGML, 172
sit, 57
splitfile, 6
stil bibliografic
Chicago, 120
simplu, 24, 120, 121
subdescriptor, 157
substituţie, 50
surse deschise, 22, 23

tabele, 100, 144–148, 152, 154, 155
TeX4ht, 176, 177
TeXnicCenter, 51, 52, 57, 72, 73
TeXnicCenter
şi Vim, 74
trimiteri
în comentarii, 103
în pagini web, 58, 173
de la un indice la altul, 157
la altă pagină, 101
la altă secţiune, 101
la bibliografie, 121
la figuri, 165
la tabele, 146

unicod, 141, 175
Unix, 6, 19, *vezi şi* GnuWin32
unelte, 20
Unruh, Dominique, 142
utilizare vs. programare, 94

Vim
    căutare, 52, 53
    comenzi în linie, 27
    configurare, 29
    corectura sintactică, 31
    cursorul, 40, 41
    dialog, 98
    instalare, 25
    meniuri, 36, 37, 39
    modificarea textului, 42–45
    modul insert, 25, 26
    modul normal, 25, 26
    modul vizual, 27
    scripturi, 30
    semne de carte, 32
    tastaturi, 131
Viton, Philip, 174
Volovich, Vladimir, 123

`wget`, 58, 59
Wilson, Peter, 143
Windows
    ascuns, 18, 19
    comenzi, 15
    ghidul, 3, 5
windows-1250, 174
WindowsXP
    variabile de mediu, 20
Word, 179

Yap, 75, 79, 96, 97, 169, 170
    şi Vim, 75