GUST

Grupa Użytkowników Systemu TEX

Zeszyt 2 (1993)

Biuletyn Polskiej Grupy Użytkowników Systemu TEX
Zeszyt 2

Gdańsk
Czerwiec 1993

Redakcja:
Włodek Bzyl
Tomek Przechlewski

Adres:
Instytut Matematyki UG
Wita Stwosza 57, 80-952 Gdańsk
matwb@halina.univ.gda.pl

W zeszycie

Od Redakcji

Redakcja biuletynu dziękuje Rektorowi Uniwersytetu Mikołaja Kopernika za pokrycie kosztów druku biuletynu.

Autorem rysunku na okładce oraz znaczka GUST-u jest Toni Walter.

W następnym numerze

- po światowym zjeździe TEX-owców w Aston,
- o formacie w którym składamy biuletyn,
- i wszystkie stałe działy.

Zapraszamy wszystkich do wzięcia udziału w konkursie (szczegóły w numerze).

Zachęcamy do współpracy. Piszcie na adres:

◇ Włodzimierz Bzyl
Instytut Matematyki UG
Wita Stwosza 57, 80-952 Gdańsk
matwb@halina.univ.gda.pl

Bachotek'93

Jak hartowała się tradycja

Toni Walter

Tradycyjna Pierwsza Ogólnopolska Konferencja TEX-owa już za nami. W dniach od 30 kwietnia do 2 maja 1993 r. odbył się w Bachotku k. Brodnicy zlot polskich użytkowników TEX-a, tych, co rozwijają TEX-a, chcą się czegoś więcej o TEX-u dowiedzieć, a przede wszystkim tych, co lubią TEX-a. Cenny był niespodziewany udział w Konferencji Phila Taylora z Anglii, przewodniczącego projektu NTS (*New Typesetting System*). Oto kilka kronikarskich spostrzeżeń i refleksji.

◇

Pierwszy dzień obrad wypełniony był czterema referatami porządkującymi podstawy wiedzy TEX-owej.

Najpierw Włodek Bzyl spokojnie, nie poddając się nastrojowi rzutnika, pokazał TEX-a z punktu widzenia użytkownika (program do komputerowego składu tekstów) oraz komputera (TEX jako kompilator). To, że składając tekst piszemy de facto program komputerowy, jest główną cechą pozwalającą na uniwersalne stosowanie tego narzędzia.

Następnie Bogusław Jackowski wprowadził nas w świat METAFONT-a, świat pikseli dużych i małych, gdzie każda najeżona myśl zostaje wygładzona krzywą Béziera. Produkcja fontów dla TEX-a to tylko jedno z możliwych zastosowań tego programu. Inne, to np. tworzenie skomplikowanych znaków graficznych, rozwiązywanie zadań geometrycznych na płaszczyźnie, rozwiązywanie układów równań liniowych...

W dalszej części Wiesław Pawłowski omówił jeden z popularniejszych formatów TEX-a, jakim jest LATEX. Nie wdając się w dywagacje na temat tej lub innej wersji TEX-owej przedstawił po prostu możliwości, jakie daje LATEX zupełnie nawet początkującemu programiście przy opracowywaniu książek i publikacji naukowych.

Po tej serii wykładów ogólnych Marek Ryćko zajął się szczegółowo pewnym fragmentem TEX-a. Autor zadał sobie trud pokazania w skondensowanej formie wszystkich kolejnych kroków, według których następuje składanie akapitu. To przedstawienie swoistej logiki i myślenia TEX-owego było dla mnie, jako szarego użytkownika, szczególnie wartościowym uzupełnieniem poprzednich wystąpień. Sądzę, że jeżeli będę patrzył zdziwiony na to, co też nieoczekiwanego dzieje się z moim składem, to w tej broszurce (wszystkie omówione dotąd prace zostały wydane w materiałach konferencyjnych) mam duże szanse znaleźć coś, czego nie uwzględniłem, na przykład przy pisaniu makra.

◇

Drugi dzień rozpoczął się od informacji Staszka Wawrykiewicza o archiwum TEX-owym. Referent, który sam jest chodzącym archiwum, dał popis żelaznej kondycji, ponieważ przez całą noc instalował na komputerze co tylko udało mu się przywieźć ze środowiska TEX-owego i wokół TEX-owego. Były więc dystrybucyjne wersje emTEX-a i MEX-a, sbTEX, drivery, konwertery, źródła fontów, przeróżne makra aplikacyjne itd.

Odtąd komputer nie miał chwili wytchnienia i każdy, komu udało się zdobyć dyskietki, kopiował do woli. Bezpłatny dostęp do potężnego pakietu oprogramowania (*public domain*) jest, oprócz merytorycznych zalet TEX-a, jeszcze jedną zachętą do stosowania tego systemu.

Dalej serię wystąpień rozpoczął Phil Taylor. W pierwszym omówił nowe cechy TEX-a dla wersji $\geq$ 3.0. Ważniejsze uzupełnienia to możliwość uwzględnienia różnych języków, nowe komendy ułatwiające składanie trudniejszych akapitów, dodanie kleju, który nie znika na początku strony, wprowadzenie fontów wirtualnych.

Następnie Norbert Jankowski zasygnalizował „Help dla LATEX-a" swojego autorstwa, który może być użyteczną, podręczną ściągawką dla wszystkich LATEX-owców. Program można zamówić za symboliczną opłatą.

Ponownie powraca Ph. Taylor, dalej zwany już Philem. W tym momencie rzutnik odmówił zupełnie współpracy i na potrzeby referatu urządzeniem tym stał się Jerzy Ludwichowski z Zarządu, który rzucał teksty slajdów za pomocą kredy na tablicę. Phil zajął się przyszłością TEX-a*.

Kontynuując komunikaty Tomasz Przechlewski wspomniał o rodzinie fontów Pandora, które mają układ zgodny z fontami cm, a Marek Ryćko mówił o wykorzystaniu w TEX-u PostScripta. Plik

---

*: Tekst wystąpienia Phila prezentowany jest dalej w numerze.

makr PostScriptowych jest oczywiście dostępny u S. Wawrykiewicza.

Na zakończenie Stanisław Romański przedstawił swoje doświadczenia z połączenia TEX-a z bazą danych. Przy składzie wszelkiego rodzaju katalogów możliwość programowania w TEX-u jest bardzo wygodna.

Trzeci dzień rozpoczął B. Jackowski od omówienia możliwości graficznych TEX-a. Rysunki przygotowane w formatach PCX, TIF, GIF lub PS można przekształcać na pliki TFM i PK, a więc używane przez TEX-a. Odpowiednie programy znajdują się w dystrybucyjnych zestawach TEX-owych.

Następnie Elżbieta Kuczyńska zasygnalizowała umieszczanie preprintów TEX-owych w bazie danych oraz podała sposoby korzystania z nich — za pomocą sieci i poczty elektronicznej, a Zofia Walczak zachęcała do korzystania z $\mathcal{AMS}$-TEX-a przy pisaniu tekstów matematycznych.

Ostatnim mówcą, ale za to przez kilka godzin, był Phil. Pod hasłem „TEX dla zaawansowanych" zobowiązał się do odpowiadania na wszelkie pytania TEX-owe, które padną z sali. Czasu starczyło jedynie na trzy zagadnienia.

Najpierw wskazał na sposób składania długich tabelek ciągnących się przez kilka stron.

Następnie omówił problem umieszczania w tekście znaku komentarza % — gdzie może być, gdzie musi być oraz kiedy nie może się pojawić.

Na koniec zajął się sprawą emulowania LATEX-a, a więc zapisania w formacie `plain` tych makr, na których opiera się LATEX. Generalnie Phil nie jest zwolennikiem LATEX-a, ale ktoś może to uznać za kwestię gustu.

Końcówkę wystąpienia Phila zakłóciło wydarzenie, którego nie spodziewaliśmy się w — bądź co bądź — lesie. Otóż wykłady odbywały się w pomieszczeniu świetlicowym, gdzie stał telewizor. O godzinie 17.30 na salę wkroczyła mamusia z córkami i wszystkie one MUSIAŁY obejrzeć „Dynastię". Phil przez kilka minut współbrzmiał z Blakiem i Alexis, ale zdecydowaliśmy się jednak wynieść tablicę na zewnątrz i ostatnie pół godziny Phil wykładał pod drzewami.

Tak mniej więcej wyglądała oficjalna część robocza. Oficjalna część rozrywkowa trwała przez całą resztę pobytu. Dwie noce (może nie całe) spędzone zostały przy tradycyjnym ognisku z kiełbaskami i śpiewami. B. Jackowski akompaniował na gitarze i trudno się było zdecydować, czy lepiej gra, czy lepiej programuje. Można było także podziwiać wokalno-gitarowe umiejętności Phila i Staszka Wawrykiewicza. Cały czas trwały rozmowy, nawiązywano kontakty, a odjeżdżający uczestnicy żegnali się tradycyjnym „TEXuj zdrów!".

Ostatniego dnia Staszek Wawrykiewicz wręczył ufundowaną przez siebie nagrodę specjalną dla najbardziej zaangażowanego w konferencję uczestnika oraz za całokształt. Tradycyjną butelkę „Napoleona" otrzymał po raz kolejny Bogusław Jackowski.

Ogólnie przebieg konferencji, która była przecież debiutancka, zrobił na mnie, szarym użytkowniku, całkiem pozytywne wrażenie. Zespół z Uniwersytetu M. Kopernika w Toruniu w składzie: opiekuńcza Jolanta Szelatyńska, ofiarny Jerzy Ludwichowski oraz cichy i pracowity Roman Słupecki, sprawdził się całkowicie. Podobne opinie słyszałem również od innych uczestników, więc może tylko cieszyć, że BYŁO DOBRZE.

◇

Na koniec kilka uwag już tylko ode mnie — korzystając z tego, że jestem przy komputerze.

1. Cztery wykłady wstępne potraktowane jako tutoriale spełniły według mnie swoje zadanie. Z takiego porządkującego materiału słuchacz o każdym poziomie zaawansowania może wybrać coś interesującego dla siebie. Pomysł godny na tradycję. Marzą mi się już inne tutoriale, a ten szczegółowy to na temat *output routines*, czyli o przygotowaniu strony (i stron) do wydruku. Zresztą na ten temat można by zorganizować oddzielną konferencję.

2. Zagadnienia poruszane na Konferencji wydają mi się być kopalnią tematów, które można by przedstawiać w GUŚCIE. Może autorzy komunikatów przygotują coś? Może warto też napisać coś na temat spraw poruszanych przez Phila.

3. Cenna jako początek tradycji była sesja pytań i odpowiedzi. Wynikła ona jednak trochę znienacka i bez specjalnego przygotowania. Warto by wprowadzić ją jako żelazny punkt na dalszych konferencjach i dać jej więcej czasu. Duże i małe problemy, z którymi nie może poradzić sobie przeciętny użytkownik, oraz oczywiście odpowiedzi na nie, mogłyby też znaleźć miejsce w „Poradniku TEX-owca" jako stałym kąciku w biuletynie GUSTU.

DO ZOBACZENIA ZA ROK. TEXUJCIE ZDROWI!

PLAIN

The Future of TEX*

Philip Taylor

Abstract

TEX and the other members of Knuth's *Computers & Typesetting* family are arguably amongst the most successful examples of computer software in the world, having been ported to almost every conceivable operating system and attracting an allegiance that verges on the fanatical. Development work on this family has now ceased, and many members of the computer typesetting community are concerned that some action should be taken to ensure that the ideas and philosophy enshrined in TEX are not allowed simply to fade away. In this paper, we discuss some of the options available for perpetuating the TEX philosophy, and examine the strengths and weaknesses of the present TEX system. We conclude by postulating a development strategy for the future which will honour both the letter and the spirit of Knuth's wish that TEX, METAFONT and the Computer Modern typefaces remain his sole responsibility, and at the same time ensure that the philosophy and paradigms which are the strengths of TEX are not lost for ever by having artificial constraints placed on their evolution.

♢

"My work on developing TEX, METAFONT and Computer Modern has come to an end." [1] With these words, Professor Donald E. Knuth, creator of TEX, informed the world that the evolution of probably the most successful computer typesetting system yet developed had ceased, and that with the sole exception of essential bug fixes, no further changes would be made. TEX's version number will asymptotically approach π as bug fixes are made, and at the time of his death, it will be renamed 'TEX, Version π'; thereafter it will remain exactly as he last left it: a fitting and appropriate memorial to one of the most productive and inspired computer scientists (and mathematicians, and Bible scholars) that the world has ever known.

The future of TEX is therefore totally determined: why, then, is this paper entitled "*The Future of TEX*"? Because, primarily, TEX is already fifteen years old — four years as a child (TEX 78); eight years as an adult (TEX 82); and three years in maturity (TEX 3). Fifteen years is a long time in the lifespan of computer languages: TEX represents the pinnacle of Neanderthal evolution, building on the genetic heritage of Runoff, Nroff, Troff, Ditroff and Scribe, whilst Cro-Magnon man, in the guise of Ventura Publisher, Aldus Pagemaker and Quark Xpress, is already sweeping over the face of the planet. The halcyon days are long since gone (or so it would seem) when it was socially acceptable to: enter text; check it for spelling errors (by eye!); insert a series of formatting commands; pass the whole through an interpreter; identify the first error; correct the first error; pass the whole through the interpreter again; identify the second error; correct the second error; pass the whole through the interpreter for a third time; repeat for all subsequent errors...; pass the whole through the interpreter for the n^{th} time; then pass it through the interpreter again (to resolve forward- and cross-references); preview a facsimile of the final copy on the computer screen; notice a formatting error; and go right back to editing the file: our colleagues sit there clicking away on their mice[1] like demented death-watch beetles[2] and think us totally mad; and mad we surely must be, for we not only enjoy this mode of working, we seek to convert the demented mouse clickers into TEX users as well!

Why? What is it about TEX that is so totally addictive? Is it perhaps TEX's descriptive and character-oriented nature — the fact that, in direct opposition to current trends, TEX requires the user to think about what he or she wants to achieve, and then to express that thought as a series of words and symbols in a file, rather than as a series of ephemeral mouse movements on a screen? Is it, perhaps, its portability — the fact that implementations (almost entirely public domain) exist for every major operating system in the world? Is it the deterministic nature of TEX — the fact that a given sequence of TEX commands and text-to-be-typeset

* This article is based on a paper which was first presented in Prague, Czechoslovakia, and which appears in its original form in the proceedings thereof [4]; it has been updated to reflect changes made for the DANTE '93 meeting at Chemnitz, and the GUST '93 meeting at Bachotek.

[1] *Mus ordinatus microsoftiensis* or *Mus ordinatus applemacintoshii*

[2] *Xestobium rufovillosum*

will always produce *exactly* the same results, regardless of the machine on which it is processed? Is it the 'boxes and glue' paradigm, which provides a simple but somewhat naïve model of black and white space on the printed page? The ease with which form and content can be separated? The implementation as a macro, rather than a procedural, language? (would a procedural TEX still be recognisably TEX?) Is it, perhaps, the incredible contortions through which one occasionally has to go to achieve a desired result? (Or the incredible elation when such contortions finally achieve their intended effect?) How many of these elements could be eliminated and still leave something that is recognisably TEX? I propose to return to these questions, and to attempt to answer some of them, later in this paper.

It seems, then, that we have a choice: we can either allow natural selection to take its course, in which case TEX, having fulfilled its appointed rôle on this planet (which I assume is to teach us the merits of literate programming, whilst encouraging us to devote ever more time to the typesetting of beautiful papers, presumably at the expense of ever less time spent actually researching or writing them), will surely join XCHLF, JEAN & JOSS in the great bit-bin in the sky; or we can adopt a corporate responsibility for the future of TEX and intercede in the process of natural selection, taking steps to ensure that TEX evolves into a typesetting system which is so demonstrably superior to the miasma of mouse-based, menu-driven, manipulators of text and images which are currently snapping at its heels that no-one will be able to deny it its rightful place at the forefront of typesetting technology for the twenty-first century.

Let us consider the options which are available to us:

1. We can leave TEX exactly as it is: this is clearly a defensible position as it is exactly what Knuth himself intends to do; it would be extremely arrogant of us to suggest that we know better than Knuth in this respect.
2. We can enhance TEX by just enough that those who really understand its power, its limitations, and its inner workings agree that it no longer has demonstrable defects (i.e. there are some 'simple' typesetting tasks with which TEX$_\pi$ could not deal correctly, but with which an enhanced TEX could).
3. We can enhance TEX by incorporating the combined wish-lists of its major practitioners, thereby seeking to make TEX all things to all men (and all women), whilst retaining its present 'look and feel'.
4. We can enhance TEX as in option 3 above, whilst taking the opportunity to re-consider, and perhaps substantially change, its present look and feel.
5. We can take the opportunity to do what I believe Knuth himself might do, were he to consider today the problems of typesetting for the first time: look at the very best of today's typesetting systems (clearly including TEX among these), and then design a *new* typesetting system, far more than just a synthesis of all that is best today, which addresses the needs and potential not only of today's technology, but that of the foreseeable future as well. We would need to find some way to incorporate that spark of genius which characterizes Knuth's work!

No doubt each of us will have his or her own ideas on the desirability or otherwise of each of these options; it is not my intention in this paper to attempt to persuade you that any one of them is clearly preferable; but I would be shirking my responsibilities were I not to caution that, in my opinion, option 3 appears to represent the worst of all possible worlds, representing as it does a clear case of 'creeping featurism' at its worst while not possessing any redeeming qualities of originality.

Option 1 is, as I have suggested above, clearly defensible, in that it is Knuth's own preferred position; despite my fears that TEX will succumb to the pressures of natural selection if it is adopted, it may be that TEX represents both the pinnacle and the end of an evolutionary line, and that future typesetting systems will be based on an entirely different philosophy (e.g. mouse-based).

Option 2 represents the most conservative evolutionary position and has, I believe, much to commend it, certainly in the short term: it would retain the present look and feel of TEX; and compatibility with current TEX programs, whilst not intrinsically guaranteed, could be ensured by careful design; at the very worst, one could envisage a command-line qualifier which would disable the extensions, leaving a true TEX 3 underneath. Although option 2 is in opposition to Knuth's expressed wishes, he has made it plain that he has

no objection to such enhancements *provided that* the resulting system is not called TEX. I propose that we term the results of adopting option 2 'Extended TEX', both to indicate its nature, and, more importantly, to comply with the spirit as well as the letter of Knuth's wishes.

Option 3 is considerably less conservative, but does at least retain the present look and feel of TEX; it is completely open-ended in terms of the extensions made to TEX, and offers the opportunity to make sweeping enhancements (I hesitate to use the word 'improvements' for the reasons outlined above). Compatibility with current TEX programs need not prove problematic, provided that the design were adequately thought out, and again the possibility of a '/noextensions' qualifier provides a fallback position. The timescale for such an implementation would not be small if a new swarm of bugs is to be prevented, and it is not clear how future obsolescence is to be avoided: after all, if 'The Ultimate TEX' (as I will term it) includes all the proposed enhancements of TEX's major practitioners, what enhancements remain to be implemented in the future?

Option 4 represents the first attempt at a true re-design of TEX, allowing as it does the option to re-think TEX's look and feel, whilst continuing to incorporate many of its underlying algorithms. One could envisage, for example, an implementation of TEX in which text and markup were kept entirely separate, with a system of pointers from markup to text (and *vice versa*?). One advantage of such a scheme is that it would eliminate, at a stroke, the troublesome nature of the <space> character which currently complicates TEX; the escape character could become redundant, and the problems of category codes possibly eliminated. Of course, this is just one of many such possibilities: once one abandons the look and feel of TEX, the whole world becomes one's typesetting oyster. One might term such a version of TEX 'Future TEX'.

Option 5 is without doubt the most radical: not only does it reject (at least, initially), TEX's look and feel, it challenges the entire received wisdom of TEX and asks instead the fundamental question: "How should computer typesetting be carried out?" In so doing, I believe it best represents Knuth's own thoughts prior to his creation of TEX 78, and, by extrapolation, the thoughts which he might have today, were he faced for the first time with the problems of persuading a phototypesetter to produce results worthy of the texts which it is required to set. I think it important to note that there is nothing in option 5 which automatically implies the rejection of the TEX philosophy and paradigms: it may well be that, after adequate introspection, we will decide that TEX does, in fact, continue to represent the state of the typesetting art, and that we can do no better than either to leave it exactly as it is, or perhaps to extend it to a greater or lesser extent whilst retaining its basic model of the typesetting universe of discourse; on the other hand, neither does it imply that we *will* reach these conclusions. I will call such a system 'A New Typesetting System' (to differentiate it from 'The New Typesetting System' which is the remit of NTS, *q.v.*).

The options outlined above are not necessarily mutually exclusive: we might decide, for example, to adopt option 2 as an interim measure, whilst seeking the resources necessary to allow the adoption of option 5 as the preferred long-term position (indeed, I have considerable sympathy with this approach myself). But no matter which of the options we adopt, we also need to develop a plan of campaign, both to decide which of the options is the most preferable (or perhaps to adopt an option which I have not considered) and then to co-ordinate the implementation of the selected option or options.

So far, this paper has been concerned primarily with generalities; but I propose now to look at some of the specific issues to which I have earlier merely alluded, and to offer some personal opinions on possible ways forward. I propose to start by attempting to answer the question which I believe lies at the very heart of our quest: "What is the essence of TEX?"

It seems to me that there are some aspects of TEX which are truly fundamental, and some which are merely peripheral: among the fundamental I include its descriptive and character-oriented nature, its portability, and its deterministic behaviour; I also include some elements which I have not so far discussed: its programmability (for example, the way in which loops can be implemented, even though they are not intrinsic to its design), its generality (the fact that it can be used to typeset text, mathematics, and even music), its device independence, and its sheer æsthetic excellence (the fact that, in reasonably skilled hands, it can produce results which are virtually indistinguishable from material set professionally using traditional

techniques). Equally important, but from a different perspective, are the facts that it is totally documented in the ultimate exposition of literate programming (the *Computers & Typesetting* quintology), that it is virtually bug-free, that any bugs which do emerge from the woodwork are rapidly exterminated by its author, and finally that for higher-level problems (i.e. those which are at the programming/user-interface level rather than at the WEB level), there are literally thousands of skilled users to whom one can appeal for assistance. We should not forget, too, Knuth's altruism in making the entire source code[3] freely available with an absolute minimum of constraints. It is almost certainly true that this last fact, combined solely with the sheer excellence of TEX, is responsible for TEX's widespread adoption over so much of the face of our planet today.

Among its more peripheral attributes I include its implementation as a macro, rather than as a procedural or declarative, language, and perhaps more contentiously, its fundamental paradigm of 'boxes and glue'. I hesitate to claim that boxes and glue are not fundamental to TEX, since in many senses they clearly are: yet it seems to me that if a descendant of TEX were to have detailed knowledge of the *shape* of every glyph (rather than its bounding box, as at present), and if it were perhaps to be capable of typesetting things on a grid, rather than floating in space and separated by differentially stretchable and shrinkable white space, but were to retain all of the other attributes asserted above to be truly fundamental, then most would recognise it as a true descendant of TEX, rather than some mutated chimera.

Without consciously thinking about it, I have, of course, characterized TEX by its strengths rather than its weaknesses.[4] But if we are to intervene in the processes of natural selection, then it is essential that we are as familiar with TEX's weaknesses as with its strengths: if it had no weaknesses, then our intervention would be unnecessary, and the whole question of the future of TEX would never have arisen. But whilst it is (relatively) easy to identify a subset of its characteristics which the majority of its practitioners (I hesitate to say 'all') would agree represent its fundamental strengths, identifying a similar subset of its characteristics which represent its fundamental weaknesses is far more contentious. None the less, identify such a subset we must.

Perhaps the safest starting point is to consider the tacit design criteria which Knuth must have had in mind when he first conceived of TEX, and which remain an integral part of its functionality today. TEX, remember, was born in 1978 — a time when computer memories were measured in kilobytes rather than megabytes, when laser printers were almost unknown, when the CPU power of even a University mainframe was probably less than that available on the desktops of each of its academics today, and when real-time preview was just a pipe dream.[5] Each and every one of these limitations must have played a part in TEX's design, even though Knuth may not have been consciously aware of the limitations at the time. (After all, we are only aware of the scarcity of laser printers in 1978 because of their ubiquity today; we aren't aware of the limiting effects of the scarcity of ion-beam hyperdrives because they haven't yet been invented...). But by careful reading of *The TEXbook* (and even more careful reading of TEX.WEB), we can start to become aware of some of the design constraints which were placed on Knuth (and hence on TEX) because of the limits of the then-current technology. For example, on page 110 one reads: "TEX uses a special method to find the optimum breakpoints for the lines in an entire paragraph, but it doesn't attempt to find the optimum breakpoints for the pages in an entire document. *The computer doesn't have enough high-speed memory capacity to remember the contents of several pages* [my stress], so TEX simply chooses each page break as best it can, by a process of 'local' rather than 'global' optimisation." I think we can reasonably deduce from this that if memory had been as cheap and as readily available in 1978 as it is today, TEX's page-breaking algorithm may have been very different. Other possible limitations may be inferred from the list of numeric constants which appear on page 336, where, for example, the limit of 16 families for maths fonts is

[3] including source for the TEX and METAFONT books; this is frequently forgotten...

[4] OK, I admit it: TEX *might* have weaknesses...

[5] Although on page 387 (page numbers all refer to *The TEXbook* unless otherwise stated), we find "Some implementations of TEX display the output as you are running".

stated (a source of considerable difficulties for the designers of the New Font Selection Scheme);[6] 16 category codes, too, although seemingly just enough, force the caret character (^) to serve triple duty, introducing not only 64-byte offset characters and hexadecimal character specifiers, but also serving as the superscript operator.

So, we may reasonably infer that the combined restrictions of limited high-speed memory, inadequate CPU power, and very limited preview and proof facilities, combined to place limitations on the original design of TEX; limitations the effect of which which may still be felt today. It is perhaps unfortunate that in at least one of these areas, that of high-speed memory, there are still systems being sold today which have fundamental deficiencies in that area: I refer, of course, to the countless MS/DOS-based systems (without doubt the most popular computer system ever invented) which continue to carry within them the design constraints of the original 8088/8086 processors. Because of the ubiquity of such systems, there have been a fair number of submissions to the NTS list urging that any development of TEX bear the constraints of these systems in mind; despite the fact that I too am primarily an MS/DOS user, I have to say that I do not feel that the 64K-segment, 640K-overall limitations of MS/DOS should in any way influence the design of a new typesetting system. Whilst I feel little affinity for the GUI-based nature of Microsoft Windows, its elimination of the 640K-limit for native-mode programs is such a step forward that I am prepared to argue that any future typesetting system for MS/DOS-based systems should assume the existence of Windows (or OS/2), or otherwise avoid the 640K barrier by using techniques such as that adopted by Eberhard Mattes' *emTEX386*.[7] If we continue to observe the constraints imposed by primitive systems such as MS/DOS, what hope have we of creating a typesetting system for the future rather than for yesterday?

These might be termed the historical (or 'necessary') deficiencies of TEX: deficiencies over which Knuth essentially had no control. But in examining the deficiencies of TEX, we must also look to the needs of its users, and determine where TEX falls short of these, regardless of the reasons. The term 'users', in this context, is all-encompassing, applying equally to the totally naïve user of LATEX and to the format designers themselves (people such as Leslie Lamport, Michael Spivak, and Frank Mittelbach); for although it is possible for format designers to conceal certain deficiencies in TEX itself (e.g. the lack of a `\loop` primitive), the more fundamental deficiencies will affect both. (Although it is fair to say that a sure sign of the skill of a format designer is the ease with which he or she can conceal as many of the apparent deficiencies as possible). An excellent introduction to this subject is the article by Frank Mittelbach in *TUGboat*, 'E-TEX: Guidelines for future TEX' [2], and the subsequent article by Michael Vulis, 'Should TEX be extended?' [3]. Perhaps less accessible, and certainly more voluminous, are the combined submissions to NTS-L, which are archived at `Ftp.Th-Darmstadt.De` as {\tt /pub/tex/documentation/nts-l/*.

So, what are these so-called 'fundamental deficiencies'? No doubt each of us will have his or her own ideas, and the three references cited above will serve as an excellent starting point for those who have never considered the subject before. What follows is essentially a very personal view — one person's ideas of what he regards as being truly fundamental. It is not intended to be exhaustive, nor necessarily original: some of the ideas discussed will be found in the references given; but I hope and believe that it is truly representative of current thinking on the subject. Without more ado, let us proceed to actual instances.

1. *The lack of condition/exception handling:* It is not possible within TEX to trap errors; if an error occurs, it invariably results in a standard error message being issued, and if the severity exceeds that of 'warning'[8] (e.g. overfull or underfull boxes), user interaction is required. This makes it impossible for a format designer to ensure that all errors are handled by the format, and actually prevents the adoption of adequate defensive programming techniques. For example, it is not possible for the designer of a font-handling system to trap an attempt to load a font which does not exist on the target system.
2. *The inability to determine that an error has occurred:* The `\last...` family (`\lastbox`,

[6] Frank Mittelbach and Rainer Schöpf

[7] *emTEX386* uses a so-called 'DOS extender'.

[8] I use the VAX/VMS conventions of 'success', 'informational', 'warning', 'error' and 'severe error' as being reasonably intuitively meaningful here.

\lastkern, \lastpenalty, \lastskip) are unable to differentiate between the absence of a matching entity on the current list and the presence of a zero-valued entity; since there is all the difference in the world between a penalty of zero and no penalty at all, vital information is lost.

3. *The hierarchical nature of line-breaking and page-breaking:* Once a paragraph has been broken into lines, it is virtually impossible to cause TEX to reconsider its decisions. Thus, when a paragraph spans two pages, the material at the top of the second page will have line breaks within it which are conditioned by the line breaks at the bottom of the previous page; this is indefensible, as the two occur in different visual contexts. Furthermore, it prevents top-of-page from being afforded special typographic treatment: for example, a figure may occur at the top of the second page, around which it is desired to flow text; if the paragraph has already been broken, no such flowing is possible (the issue of flowing text in general is discussed below). The asynchronous nature of page breaking also makes it almost impossible to make paragraph shape dependent on position: for example, a particular house style may require paragraphs which start at top of page to be unindented; this is non-trivial to achieve.
4. *The local nature of page breaking:* For anything which approximates to the format of a Western book, the verso-recto spread represents one obvious visual context. Thus one might wish to ensure, for example, that verso-recto pairs always have the same depth, even if that depth varies from spread to spread by a line or so. With TEX's present page breaking mechanism, allied to its treatment of insertions and marks, that requirement is quite difficult to achieve. Furthermore, by localising page breaking to the context of a single page, the risk of generating truly 'bad' pages is significantly increased, since there is no look-ahead in the algorithm which could allow the badness of subsequent pages to affect the page-breaking point on the current page.
5. *The analogue nature of 'glue':* TEX's fundamental paradigm, that of boxes and glue, provides an elegant, albeit simplistic, model of the printed page. Unfortunately, the flexible nature of glue, combined with the lack of any underlying grid specification, makes grid-oriented page layup impossible to achieve, at least in the general case. The present boxes and glue model could still be applicable in a grid-oriented version of TEX, but in addition there would need to be what might be termed 'baseline attractors': during the glue-setting phase, baselines would be drawn towards one of the two nearest attractors, which would still honour the constraints of \lineskiplimit (i.e. if the effect of drawing a baseline upwards were to bring two lines too close together, then the baseline would be drawn downwards instead).
6. *The lack of any generalised ability to flow text:* TEX provides only very simple paragraph shaping tools at the moment, of which the most powerful is \parshape; but one could envisage a \pageshape primitive and even a \spreadshape primitive, which would allow the page or spread to be defined as a series of discrete areas into which text would be allowed to flow. There would need to be defined a mechanism (not necessarily within the primitives of the language, but certainly within a kernel format) which would allow floating objects to interact with these primitives, thereby providing much needed functionality which is already present in other (mouse-oriented) systems.
7. *An over-simplistic model of lines of text:* Once TEX has broken paragraphs into lines, it encapsulates each line in an \hbox the dimensions of which represent the overall bounding box for the line; when (as is usually the case) two such lines occur one above the other, the minimum separation between them is specified by \lineskiplimit. If any two such lines contain an anomalously deep character on the first line, and/or an anomalously tall character on the second, then the probability is quite great that those two lines will be forced apart, to honour the constraints of \lineskiplimit; however, the probability of the anomalously deep character coinciding with an ascender in the line below, or of the anomalously tall character coinciding with a descender in the line above, is typically rather small: if TEX were to adopt a 'skyline'[9]

[9] This most apposite and descriptive term was coined by Michael Barr.

model of each line, rather than the simplistic bounding-box model as at present, then such line pairs would not be forced apart unless it was absolutely necessary for legibility that they so be. Note that this does not require TEX to have any knowledge of the characters' *shape*; the present bounding-box model for characters is still satisfactory, at least for the purposes of the present discussion.

8. *Only partial orthogonality in the treatment of distinct entities:* TEX provides a reasonably orthogonal treatment for many of its entities (for example, the \new... family of generators), but fails to extend this to cover all entities. Thus there is no mechanism for generating new instances of \marks, for example. Similarly, whilst \the can be used to determine the current value of many entities, \the \parshape returns only the number of ordered pairs, and not their values (there is no way, so far as can be ascertained, of determining the current value of \parshape). It is possible to \vsplit a \vbox (or \vtop), but not to *\hsplit an \hbox. The decomposition of arbitrary lists is impossible, as only a subset of the necessary \last... or \un... operators is provided. The operatorless implicit multiplication of <number><dimen-or-skip register> (yielding <dimen>) is also a source of much confusion; it might be beneficial if the concept were generalised to <number><register> (yielding <register-type>). However, this raises many related questions concerning the arithmetic capabilities of TEX which are probably superficial to our present discussion. I would summarise the main point by suggesting that orthogonality could be much improved.
9. *Inadequate parameterisation:* TEX provides a very comprehensive set of parameters with which the typesetting process may be controlled, yet it still does not go far enough. For example, one has \doublehyphendemerits which provide a numeric measure of the undesirability of consecutive hyphens; it might reasonably be posited that if two consecutive hyphens are bad, three are worse, yet TEX provides no way of indicating the increased undesirability of three or more consecutive hyphens. Also concerned with hyphenation is \brokenpenalty, which places a numeric value on the undesirability of breaking a page at a hyphen; again it might be posited that the undesirability of such a break is increased on a recto page (or reduced on a verso page), yet only one penalty is provided. A simple, but potentially infinite, solution would be to increase the number of parameters; a more flexible solution might be to incorporate the concept of formula-valued parameters, where, for example, one might write something analogous to \brokenpenalty = {\ifrecto —500— \else —200— \fi}, with the implication of delayed evaluation.
10. *Inadequate awareness of æsthetics:* TEX is capable of producing results which æsthetically are the equal or better of any computer typesetting system available today, yet the results may still be poorer than that achieved by more traditional means. The reason for this lies in the increased detachment of the human 'operator', who now merely conveys information to the computer and sits back to await the results. When typesetting was accomplished by a human compositor, he or she was aware not only of the overall shape of the text which was being created, but of every subtle nuance which was perceivable by looking at the shapes and patterns created on the page. Thus, for example, rivers (more or less obvious patterns of white space within areas of text, where no such patterns are intended), repetition (the same word or phrase appearing in visually adjacent locations, typically on the immediately preceding or following line), and other æsthetic considerations leapt out at the traditional typesetter, whereas TEX is blissfully unaware of their very existence. Fairly complex pattern matching and even image processing enhancements might need to be added to TEX before it was truly capable of setting work to the standards established by hot-metal compositors.

Clearly one could continue adding to this list almost indefinitely; every system, no matter how complex, is always capable of enhancement, and TEX is no exception to this rule. I have quite deliberately omitted any reference to areas such as rotated text and boxes, support for colour, or support for graphics, as I believe them to be inappropriate to the current discussion: they are truly *extensions* to TEX, rather than deficiencies

which might beneficially be eliminated. But I believe I have established that there *are* areas in which TEX is capable of being improved, and would prefer to leave it at that.

Considering first the conservative approach, we will need to identify what is feasible, as well as what is desirable. Clearly this will require advice from those who are truly familiar with TEX.WEB, as I see this approach purely as a change-file layered on the WEB rather than as a re-write in any sense.

For the radical approach, familiarity with WEB is probably unnecessary, and indeed may be a disadvantage: if we are seeking a truly NEW Typesetting System, then detailed familiarity with current systems may tend to obfuscate the issue, and certainly may tend to constrain what should otherwise be free-ranging thoughts and ideas. We will need to consult with those outside the TEX world, and the advice of practising typographers and (probably retired) compositors will almost certainly prove invaluable. But above all we will need people with vision, people who are unconstrained by the present limits of technology, and who are capable of letting their imagination and creativity run riot.

And what conclusions might such a group reach? Almost by definition, the prescience required to answer such rhetorical questions is denied to mere mortals; but I have my own vision of a typesetting system of the future, which I offer purely as an example of what a New Typesetting System might be. Firstly (and despite my quite ridiculous prejudices against windowing systems), I believe it will inherently require a multi-windowing environment, or will provide such an environment itself (that is, I require that it will make no assumptions about the underlying operating environment, but will instead make well-defined calls through a generic interface; if the host system supports a multi-windowing environment such as Microsoft Windows or the X Window System, the NTS will exploit this; if the host system does not provide such intrinsic support, then it will be the responsibility of the implementor to provide the multi-windowing facilities). I envisage that perhaps as many as eight concurrent displays might be required: linked graphic and textual I/O displays, through which the designer will be able to communicate the underlying graphic design in the medium of his or her choice (and observe in the other window the alternative representation of the design); an algorithmic (textual) display, through which the programmer will communicate how decisions are to be made; two source displays, one text, one graphic, through which the author will communicate the material to be typeset; and a preview display, through which an exact facsimile of the finished product may be observed at any desired level of detail. A further display will provide interaction (for example, the system might inform the user that some guidance is needed to place a particularly tricky figure), and the last will enable the user to watch the system making decisions, without cluttering up the main interactive window. Needless to say, I assume that the system will essentially operate in real time, such that changes to any of the input windows will result in an immediate change in the corresponding output windows. I assume, too, that the input windows will be able to slave other unrelated programs, so that the user will be able to use the text and graphics editors of his or her choice. Of course, not all windows will necessarily be required by all users: those using pre-defined designs will not need either the design-I/O or the algorithm-input windows, and will be unlikely to need the trace-output window; but the interaction window may still be needed, and of course the source-input windows unless the source, too, has been acquired from elsewhere. For just such reasons, the system will be capable of exporting any designs or documents created on it in plain text format for import by other systems.

And underneath all this? Perhaps no more than a highly refined version of the TEX processor; totally re-written, probably as a procedural language rather than a macro language (why procedural rather than, say, list processing or declarative? to ensure the maximum acceptability of the system: there are *still* more people in the world who feel comfortable with procedural languages than with any of the other major genres), and obviously embodying at least the same set of enhancements as the interim conservative design, together with support for colour, rotation, etc. The whole system will, of course, be a further brilliant exposition of literate programming; will be placed in the public domain; will be capable of generating DVI files as well as enhanced-DVI and POSTSCRIPT; and will be so free of bugs that its creators will be able to offer a reward, increasing in geometric progression, for each new bug found...

But we will need one final element, and I have deliberately left this point to the very end: we will need the advice of Don Knuth himself. Don

has now distanced himself from the TEX project, and is concentrating on *The Art of Computer Programming* once again. This detachment is very understandable — TEX has, after all, taken an enormous chunk out of his working (and, I suspect, private) life — and I hope that we all respect his wish to be allowed to return once again to 'mainstream' computer science, mathematics, and Bible study. But I think it inconceivable that we can afford to ignore his advice; and if I were to have one wish, it would be this: that I would be permitted to meet him, for whatever time he felt he could spare, and discuss with him the entire NTS project. I would like to know, above all, what changes *he* would make to TEX, were he to be designing it today, rather than fifteen years ago; I would like to know if he agrees that the deficiencies listed above (and those that appear elsewhere) are genuine deficiencies in TEX, or are (as I sometimes fear) simply the result of an inadequate understanding of the true power and capabilities of TEX; and I would like to know how he feels about the idea of an 'Extended TEX' and of a New Typesetting System (I suspect he would be far more enthusiastic about the latter than the former). And I suppose, if I am honest, I would just like to say 'Thank you, Don', for the countless hours, days, weeks, months and probably years of pleasure which TEX has given me.

♢

The preceding is essentially a summary of the paper which I first presented at Prague in 1992; what follows is intended to bring the reader up to date, and is reasonably accurate as of May 1993:

At the '92 Annual General Meeting of DANTE, Joachim Lammarsch announced the formation of a working group, provisionally entitled 'NTS' ('New Typesetting System'), to investigate ways by which the philosophy and paradigms of TEX might be perpetuated; the group was to be chaired by Rainer Schöpf, and had representatives from both DANTE and UK-TuG (the group was, and is, a truly international group, organised under the ægis of DANTE but not restricted to members thereof). During the year that followed, members of the group listened to, and contributed to, a wide-ranging discussion which took place on NTS-L, but the members of the group never actually met (on NTS business, that is; they probably met for social and other reasons), and no NTS-X list was ever formed (and therefore no discussion ever took place thereon).

During the period leading up to DANTE '93, Rainer realised that his other committments (particularly his committment to the LATEX-3 project, but also, of course, to his full-time employment...) prevented him from really getting the NTS project off the ground, and asked the present author if he would be willing to take over the project. I was very willing to accede to this request, and at the DANTE '93 meeting at which this paper was formally presented, Joachim Lammarsch announced the dissolution of the previous NTS group, and the formation of a new group of the same name, under the leadership of the present author; no other members of the group were nominated at the time, it being left up to the author to invite whomsoever he chose to participate in the group's activities.

Perhaps the major problem now facing the NTS team is a lack of public confidence; the group has now been in existence for over a year, and yet has apparently achieved nothing: it has listened, but apparently done no more. Because of this, I am convinced that if NTS is ever to be more than a pipe dream, it needs to accomplish something worthwhile within the next year; if two years go by, and the group has still achieved nothing, its reputation will undoubtedly suffer severely. This problem is sufficiently important that I am now prepared to compromise the ideals which I outlined in the Prague version of this paper, and concentrate on Option 2: enhance TEX by just enough that its major practitioners agree that it no longer had demonstrable defects *that could be rectified within the current implementation*; furthermore, I believe that in order to regain the confidence which we have perceivably lost, we will need to implement Option 2 in a series of phased stages, releasing the initial version within twelve months, and incremental enhancements at fairly regular ('though well-spaced: say six-monthly) intervals thereafter.

Let me then summarise my conclusions so far (which have been considerably influenced by Joachim Schrod):

1. TEX is demonstrably flawed, partly because of the era of its design, and partly because some excellent ideas were not seen through to completion (e.g. one cannot properly deconstruct a \vbox, because TEX lacks certain classes of register).

2. Despite its flaws, it none the less almost certainly represents the state of the art in Computer Typesetting, and attracts an allegiance which verges on the fanatical; its strengths far outweigh its weaknesses.
3. Given its enormous user base, and the fanaticism which it attracts, a successor to TEX which fails to capitalise on these is very unlikely to succeed unless it is so demonstrably superior (whilst remaining equally portable and free of commercial liens) that no-one could fail to recognise its superiority (it would also need to be able to process existing TEX documents and produce identical results).
4. The intellectual effort needed to create a superior system such as outlined in (3) is unlikely to be achievable in a finite timescale by a group of dedicated TEXxies working in their spare time, no matter how well motivated; such a project should be seen as a real research project, with a timescale of several years.
5. The previous incarnation of the NTS project lost 'street credibility' by being seen to do nothing — that is, it listened to NTS-L (and occasionally contributed to the discussion) whilst not actually producing anything.
6. If a new incarnation of the NTS project is to succeed, it has not only to do something useful, but has to be *seen* to be doing something useful (*facere quam vederi*).
7. If the NTS project is to be universally acceptable (which may be a pious hope, but we should aim for nothing less), then it needs to be totally compatible with TEX Vπ, not only in terms of existing TEX programs (i.e. things written in TEX), but also in terms of existing TEX implementations.
8. It therefore needs to possess two fundamental attributes:
   (a) To be written as a change file to the existing WEB, so that it builds upon, rather than replaces, existing TEX implementations; and
   (b) To be totally backwards-compatible, in that any existing TEX Vπ program will produce *identical* behaviour and results no matter whether run with TEX or with the extended TEX which NTS produces (I will refer to this extended TEX as e-TEX henceforth).
9. But of course, these two aren't enough: it must also add missing functionality to TEX, whilst remaining as close as possible to TEX in philosophy (thus it shouldn't seek to add entirely unrelated functionality [e.g. graphics], but rather to complete those elements of TEX that are already present but are in some sense incomplete).
10. One clearly missing feature of TEX Vπ is an interface to the operating system; the need for this has become so apparent to some that a discussion has recently raged as to how such functionality could be added by extending the semantics of `\openin|out`, `\read`, `\write` etc. Somewhat surprisingly, this extension of semantics has the blessing of Prof. Knuth.
11. To some (including myself), this extension of TEX's present semantics is nothing short of anathema; a bodge, where a proper solution is required.
12. I therefore propose that the newly reformed NTS group should regard as their primary rôle the identification of genuine deficiencies in TEX; the postulation and discussion of solutions to these deficiencies; the prioritisation of these solutions; and the generation of incremental reference implementations of these solutions, through the medium of change files to TEX, such that full alpha- and beta-testing of the proposed enhancements can be carried out on as many platforms as possible, thereby minimising the risk of introducing any bugs into the e-TEX code.

In summary, what I propose is that rather than regarding themselves as an esoteric research group, which is conducting research on typesetting technology suitable for the twenty-first century, the group should first concentrate on the very real problems which are encountered when TEX is pushed to its limits. Having identified real deficiencies in TEX, it should decide how those deficiencies should properly be rectified, and through the medium of a master TEX change file, implement each of the solutions in an incremental manner, starting with those that lead to the greatest rewards for the least implementation effort. By publicising the existence of tools such as TIE, WEB-Merge & Patch-WEB, the group should encourage existing TEX implementors to produce platform-specific versions of e-TEX, and should participate in the alpha- and beta-testing of

these versions. When satisified that, modulo human error, no new bugs have been introduced into the code, and that it performs *identically* to TEX when given pure TEX input (whilst accepting extensions through a mechanism to be discussed in a forthcoming paper), the group should offer the resulting e-TEX implementations to the existing TEX world. It will be necessary to monitor the take-up rate; if there is marked reluctance to adopt e-TEX, despite its total backwards compatibility, then the group may choose to abandon the whole project; this would be a shame, but better than investing vital intellectual effort in a project which no-one is going to use; if, on the other hand, the project is a success, and end-users are happy to migrate from TEX to e-TEX, then the group should continue with its work, seeking advice from the TEX world as a whole as to what genuine deficiencies remain in e-TEX, and which of those it would be most valuable to eliminate next. In this way, I hope that NTS can both do something useful for the TEX world, and to be *seen* to be doing something useful at the same time.

Philip Taylor, Bachotek 1993
Co-ordinator, NTS project

References

[1] Donald E. KNUTH: "The Future of TEX and METAFONT", in *TUGboat*, Vol. 11, No. 4, p. 489, November 1990.

[2] Frank MITTELBACH: "E-TEX: Guidelines for future TEX", in *TUGboat*, Vol. 11, No. 3, pp. 337 – 345, September 1990.

[3] Michael VULIS: "Should TEX be extended?", in *TUGboat*, Vol. 12, No. 3, pp. 442 – 447, September 1991.

[4] Zlatuška, Jiří(ed): *EuroTEX '92 Proceedings*, pp. 235 – 254, September 1992. Published by CSTUG, Czechoslovak TEX Users Group, ISBN 80-210-0480-0.

Wariacje na temat przyszłości TEX-a

Poniższy tekst jest dość swobodnym tłumaczeniem wybranych fragmentów 10-stronicowego referatu, wygłoszonego na Pierwszej Ogólnopolskiej Konferencji TEX-owej w Bachotku. Philip Taylor jest koordynatorem drugiego wcielenia grupy roboczej NTS (New Typesetting System, *Nowy System Składu*). Działalność pierwszego jej wcielenia, pod kierunkiem Rainera Schöpfa, nie zaowocowała żadnymi konkretnymi wynikami — grupa tylko „prowadziła nasłuch", rejestrowała życzenia i uwagi.

Wcześniejsza wersja referatu była po raz pierwszy zaprezentowana na konferencji *EuroTEX'92* w Pradze i jest zamieszczona w jej materiałach [4].

Wybór (nie uzgadniany z Autorem) i tłumaczenie: Włodzimierz J. Martin (wjm@sunrise.pg.gda.pl). Tytuł pochodzi ode mnie (wjm).

♢

Można argumentować, że zarówno sam TEX jak i pozostali członkowie Knuthowej rodziny *Computers & Typesetting* stanowią przykład najbardziej udanego oprogramowania na świecie. Zostało ono przeniesione pod prawie wszystkie możliwe systemy operacyjne i przyciąga użytkowników, którzy są mu oddani w stopniu graniczącym z fanatyzmem. Rozwój tego oprogramowania został już zakończony, jednakże wielu członków rodziny składaczy tekstów sądzi, że należy podjąć działania, które nie pozwolą, by filozofia i idee uwiecznione w TEX-owej świątyni po prostu rozpłynęły się i zaginęły. W artykule rozważone są niektóre opcje umożliwiające kontynuację TEX-owej filozofii oraz zbadane silne i słabe miejsca TEX-a. Na koniec postuluje się przyszłościową strategię rozwoju, uwzględniającą zarówno pisane jak i niepisane życzenia Knutha co do jego osobistej odpowiedzialności za TEX-a, METAFONT i czcionki Computer Modern, jak i zapewniającą, że filozofia i paradygmaty stanowiące siłę TEX-a nie zostaną na zawsze utracone wskutek nałożenia na ich ewolucję kagańca sztucznych ograniczeń.

♢

Co jest takiego w tym TEX-u, że przyciąga jak narkotyk jaki? Może to znakowy i opisowy charakter TEX-a, może i to, iż wprost odwrotnie do panujących obecnie trendów, TEX wymaga od użytkownika, by pomyślał trochę nad tym, co chce osiągnąć poczem wyraził tę myśl w postaci ciągów słów i symboli umieszczonych w pliku, a nie za pomocą sekwencji efemerycznych ruchów myszy[1] po ekranie. Może jest to przenośność — fakt, że jego implementacje

[1] *Mus ordinatus microsoftiensis* albo *Mus ordinatus applemacintoshii* (przyp. Autora).

(będące w przeważającej większości dobrem wspólnym — *public domain*) istnieją dla wszystkich istotniejszych systemów operacyjnych na świecie? Czy to nie czasem deterministyczny character TEX-a — fakt, iż dana sekwencja TEXowych poleceń przemieszanych z tekstem do złożenia da zawsze *dokładnie* ten sam wynik, niezależnie od maszyny, na której je przetworzymy?

♢

Mamy do wyboru kilka opcji — rozpatrzmy je pokrótce.

1. Możemy pozostawić TEX-a dokładnie takim, jaki jest. Jest to podejście w oczywisty sposób defensywne, zgodne z tym, co chce zrobić sam Knuth. Byłoby arogancją z naszej strony sugerować w tej mierze, że wiemy lepiej od Knutha.
2. Możemy ulepszyć TEX-a na tyle tylko, by ci, którzy naprawdę rozumieją gdzie leży jego siła, gdzie jego ograniczenia, rozumieją też co się dzieje w jego trzewiach zgodzili się, że nie ma on już widocznych usterek (tzn. że istnieją pewne *proste* zadania z dziedziny składu tekstów, z którymi TEXπ nie może sobie poradzić poprawnie, natomiast ulepszony TEX może).
3. Możemy ulepszyć TEX-a wcielając do niego połączone spisy życzeń najpoważniejszych użytkowników, starając się w ten sposób stworzyć TEX-a potrafiącego zrobić wszystko dla wszystkich, pozostawiając jednak jego obecny „wygląd i odczucie".
4. Możemy ulepszyć TEX-a tak jak w opcji 3 powyżej, korzystając jednak ze sposobności ponownego rozpatrzenia i ewentualnie radykalnej zmiany obecnie istniejących „wyglądu i odczucia".
5. Możemy również skorzystać z okazji i zrobić to, co sam Knuth zrobiłby bez wątpienia, gdyby miał po raz pierwszy w życiu teraz właśnie zabierać się do składu tekstów: przyjrzeć się najlepszym w tej chwili systemom składu (uwzględniając naturalnie samego TEX-a), a następnie zaprojektować *nowy* system, będący czymś więcej niż tylko syntezą tego, co dziś najlepsze, lecz spełniającym wymogi i wyzyskującym potencjalne możliwości nie tylko dzisiejszej techniki, ale również i tej, którą potrafimy przewidzieć w przyszłości. Będziemy też musieli znaleźć jakiś sposób, by znalazła się tam ta iskra geniuszu, tak znamienna dla pracy Knutha!

♢

Można z pewnością powiedzieć, że istniejące w czasie powstawania TEX-a połączone niedostatki w ilości dostępnej szybkiej pamięci, niewystarczającej mocy obliczeniowej procesora oraz bardzo ograniczonych możliwości wstępnego przeglądania i korekty złożyły się na istnienie ograniczeń w rozwiązaniach TEXa, ograniczeń, których skutki odczuwamy do dziś. Tak się nieszczęśliwie składa, że w co najmniej jednej z tych dziedzin, a mianowicie w ilości dostępnej szybkiej pamięci, nadal istnieją i są dziś sprzedawane systemy mające podstawowe niedostatki. Mam oczywiście na myśli niezliczone systemy oparte na MS-DOS (mimo, iż jest to niewątpliwie najbardziej popularny ze wszystkich systemów komputerowych, jakie kiedykolwiek wymyślono), które nadal noszą piętno ograniczeń konstrukcyjnych pierwotnych procesorów 8088/8086. Ogromne rozpowszechnienie sytemów MS-DOS spowodowało, że na liście NTS znalazło się wiele zgłoszeń, wyrażających życzenie, by przy dalszym rozwijaniu TEX-a miano na uwadze i uwzględniano ułomności tych systemów. Muszę powiedzieć, że mimo, iż sam jestem użytkownikiem przede wszystkim systemu MS-DOS nie wydaje mi się, aby narzucane przez MS-DOS ograniczenia rozmiaru segmentu do 64K i wielkości całej pamięci do 640K miało w jakikolwiek sposób mieć wpływ na projekt nowego systemu składu tekstów. (...) Będę się zawsze spierał, że każdy przyszłościowy system składu dla maszyn pracujących pod MS-DOSem musi zakładać istnienie *Windows* (albo OS/2), bądź też w jakiś inny sposób obchodzić barierę 640K za pomocą, przykładowo, techniki podobnej do użytej prze Eberhardta Mattesa w jego *emTEX386*.[2] Jeśli nadal damy się zmuszać do stosowania ograniczeń narzucanych przez prymitywne systemy w rodzaju systemu MS-DOS, jaką możemy mieć nadzieję na stworzenie systemu wybiegającego w przyszłość, a nie systemu na wczoraj?

♢

TEX ma i inne, oprócz historycznych, niedostatki. Wiele z nich projektanci formatów potrawią zgrabnie ukryć przed oczami *zwykłych* użytkowników. Dobre wprowadzenie do tych zagadnień stanowią

[2] *emTEX386* stosuje tzw. *DOS extender* (przyp. Autora).

artykuły [2] i [3], jak również znacznie obszerniejsze, acz trudniej dostępne zgłoszenia listy NTS-L, znajdujące się w archiwum `Ftp.Th-Darmstadt.De` jako:

`/pub/tex/documentation/nts-l/*`.

Jakie zatem są te tak zwane „fundamentalne niedostatki". Każdy z nas ma na to niewątpliwie swój własny pogląd. Ci, co się nigdy nad tym nie zastanawiali, mogą zacząć od wymienionych powyżej trzech źródeł. Poniżej przedstawiam swój bardzo osobisty na to pogląd — wymienione problemy uważam za rzeczywiście fundamentalne. Lista nie jest ani szczególnie oryginalna ani wyczerpująca.

1. *Brak obsługi wyjątków i warunków.* Nie daje się w TEX-u „złapać" i obsłużyć błędu, co uniemożliwia programowanie defensywne (również twórcom formatów).
2. *Niemożność stwierdzenia, że wystąpił błąd.* Jest, na przykład, ogromna, zasadnicza różnica między karą o wartości zero a brakiem kary w ogóle.
3. *Hierarchiczny charakter łamania wierszy i łamania stron.* Gdy TEX raz złamał stronę, nie można go zmusić do ponownego rozpatrzenia swej decyzji.
4. *Lokalny charakter łamania stron.*
5. *Analogowy charakter „kleju" (glue).* Nie jest możliwe uzyskanie w TEX-u rozkładu strony opartego o siatkę.
6. *Brak uogólnionego mechanizmu przemieszczania (pływania) tekstu.* Obecnie TEX ma jedynie bardzo proste mechanizmy kształtowania akapitów, z których najmocniejszym jest \parshape; można by wyobrazić sobie mechanizmy, pozwalające zdefiniować stronę (kolumnę) czy rozkładówkę jako szereg dyskretnych obszarów, po których mogły by „pływać" teksty. Takie rzeczy są bardzo potrzebne, a potrafią je już robić niektóre inne systemy *myszoidalne*.
7. *Nadmiernie uproszczony model wierszy tekstu.* Po złamaniu wierszy w akapicie, każdy z nich jest zamykany w pudełku, które może ulec rozepchnięciu w pionie przez szczególnie wystające litery, co z kolei może dać w wyniku za duże światło między wierszami. Znacznie stosowniejsze byłoby tu posługiwanie się obrysem.
8. *Częściowa jedynie ortogonalność w traktowaniu odróżnialnych obiektów.* Mimo, iż TEX traktuje ortogonalnie wiele swoich obiektów (np. rodzina generatorów \new...), nie rozciąga tego traktowania na wszystkie obiekty. Nie ma, na przykład, mechanizmu do generowania nowych istnień \mark. Podobnie, mimo, że \the pozwala uzyskać bieżace wartości wielu obiektów, \the\parshape daje jedynie liczbę uporządkowanych par, a nie ich wartości.
9. *Niedostateczna parametryzacja.* Na przykład istnieje jedna tylko wartość \brokenpenalty, wyrażająca numerycznie niechęć do łamania strony na przeniesieniu wyrazu. Istnieje jednak istotna różnica między przeniesieniem wyrazu ze strony parzystej na nieparzystą, a z nieparzystej na parzystą.
10. *Niewystarczające uwrażliwienie na estetykę.* TEX zdolny jest dać nam materiał porównywalny albo lepszy od materiału uzyskiwanego za pomocą wszystkich innych dziś dostępnych komputerowych systemów składu. Mimo tego materiał ten może być gorszy od wyników uzyskanych środkami bardziej tradycyjnymi. Weźmy chociaż, na przykład, niepożądane białe „kanały" na stronie czy pionowe sąsiedztwo tych samych wyrazów. Człowiek potrafi to zauważyć natychmiast, a TEX nawet sobie z tego nie zdaje sprawy. Po to, by TEX był w stanie dorównać osiągnięciom fachowców epoki odlewania czcionek, może być potrzebne zastosowanie złożonych technik dopasowywania wzorców czy nawet rozpoznawania obrazów.

Nie ulega wątpliwości, że listę powyższą można by powiększać nieomalże bez końca. Każdy system, nie wiedzieć jak złożony, można zawsze ulepszyć.

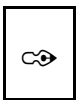
Cytowana literatura

[1] Donald E. KNUTH: "The Future of TEX and METAFONT", *TUGboat*, **11**, Nr 4, s. 489, listopad 1990.

[2] Frank MITTELBACH: "E-TEX: Guidelines for future TEX", *TUGboat*, **11**, Nr 3, ss. 337 – 345, wrzesień 1990.

[3] Michael VULIS: "Should TEX be extended?", *TUGboat*, **12**, Nr. 3, ss. 442 – 447, wrzesień 1991.

[4] Jiří Zlatuška (red): *EuroTEX '92 Proceedings*, ss. 235 – 254, wrzesień 1992. Wydane przez CSTUG (Czechosłowacką Grupę Użytkowników TEX-a), ISBN 80-210-0480-0.



LaTeX

The LaTeX3 Project*

Frank Mittelbach
Chris Rowley

Abstract

This is a brief sketch of the LaTeX3 Project: background, history, principles, aims and functionality.

The new version of LaTeX is, like the current version, a freely available system for automated processing of structured documents, formatting them to the highest typographic standards by use of the TeX typesetting software.

Although its uses include a very large range of published documents, the importance of its unsurpassed ability to format mathematical formulas will not be forgotten in producing the new version.

It is being produced by an international group of volunteers under the technical direction of Frank Mittelbach.

1 Why a new version?

With TeX, Knuth designed a formatting system [7] that is able to produce a large range of documents typeset to extremely high quality standards. For various reasons, including its quality, portability, stability and availability, TeX spread very rapidly and can nowadays be best described as a world-wide de facto standard for high quality typesetting. Although it is most famous for its ability to typeset mathematics, it is being used for many other types of document, particularly those with multi-lingual requirements.

The TeX system is fully programmable. This allows the development of high-level user interfaces whose input is processed by TeX's interpreter to produce low-level typesetting instructions; these are input to TeX's typesetting engine which outputs the format of each page in a device-independent page-description language.

Many people have made use of this powerful feature of TeX and developed their own front-ends; the most commonly used such packages are $\mathcal{AMS}$TeX and L$\mathcal{AMS}$TeX by Michael Spivak [30, 29] and LaTeX by Leslie Lamport [8]. The development of $\mathcal{AMS}$TeX was sponsored by the American Mathematical Society, a publishing house which now processes and typesets the majority of its output using TeX.

The principal aim of $\mathcal{AMS}$TeX was to simplify the user interface to the sophisticated built-in math-typesetting capabilities of TeX. It therefore does not provide support for certain more general document processing requirements (such as symbolic cross-references, automatic numbering, etc.); it is therefore most appropriate for short articles which contain lots of formulas.

The LaTeX system, on the other hand, supports the needs of long documents such as textbooks and manuals. It was designed to separate content and form as much as possible by providing the user with a generic (i.e. logical rather than visual) markup interface; this is combined with style files in which the formatting is specified. Nevertheless, LaTeX also provides a complete set of direct formatting instructions; these allow the user to access the full power of TeX's typesetting expertise when preparing the final version of the document.

Recent years have shown that the concepts and approach of LaTeX have become widely accepted. Indeed, LaTeX has become the standard method of communicating and publishing documents in many academic disciplines. This has led to many publishers accepting LaTeX source for articles and books. The American Mathematical Society, for example, now provides a LaTeX style option [1] which makes the math-typesetting features of $\mathcal{AMS}$TeX available to users of LaTeX. But the use of LaTeX in the publishing industry goes further: Elsevier Science Publishers, for example, are developing a system [24] which links SGML (for electronic storage of documents in databases) and LaTeX (for formatting these documents).

The use of LaTeX, together with SGML, is now also spreading into industrial environments, where the technical qualities of TeX together with the concepts of LaTeX are considered a powerful combination of great potential importance to such areas as corporate documentation [33] and database publishing.

* This is a reprint of an article originally submitted to the Euromath Bulletin.

With the spreading use of SGML-compliant systems, such as the Grif editor, LaTeX again is a common choice as the formatter for high quality typeset output [9]. A typical SGML Document Type Definition (DTD) uses concepts similar to those of LaTeX. Therefore, as in the Euromath system, the formatting is often implemented by simply mapping document elements to LaTeX constructs rather than directly to 'raw TeX' [31]. This enables the sophisticated analytical and processing techniques built in to much of the LaTeX software to be exploited; and it avoids the need to program in TeX.

Such developments ensured that the uses of LaTeX have become widespread, both geographically and typographically. However, they have also pushed the system way beyond its original intended purpose. Moreover, as most people who have been involved in the production of style files will agree, they have demonstrated that the current version, whilst making the author's job easier, provides little assistance to developers of new applications. Thus there can be no doubts about the need for the continued development of LaTeX into a new, improved, front-end to TeX—one that will serve the typesetting needs of the nineties and beyond—so that is what we intend the LaTeX3 project to provide.

2 History

The original goals for the development of a new version of LaTeX are described in a paper [21] presented in 1989 at the 10th annual meeting of the TeX Users Group at Stanford—it was also at this time that Leslie Lamport expressed his full support for such a project. However, we have since discovered that those original goals did not touch many of the deficiencies of the current system.

New applications of LaTeX have highlighted many limitations of its interface (both for authors of documents and for designers of styles); and further research on such problems led us to the conclusion that one gains very little by just providing more and more specialized style files to solve this or that special problem. This is because many of these deficiencies and limitations have their source in LaTeX's internal concepts and design [21, 22, 23].

The most important of those original goals, and one that is still a core part of the LaTeX3 system and a central concern of the project team, is the provision of a good style design interface—one that allows easy implementation of various layouts. (Easy, of course, is relative: we mean 'as easy as possible, given the complexity of the task'.) In order to make LaTeX3 a fully flexible and extensible system, a major effort is needed in the near future in order to get this interface 'right'.

3 Aims

The principle aims guiding our work on the project's development are as follows.

- The LaTeX3 system will provide high quality typesetting for a wide variety of document types and typographic requirements.
- For authors, it will be easy to use since it will be highly automated, but controllable.
- For editors and designers, it will support the direct formatting commands which are essential to the fine-tuning of document layout.
- It will process complex structured documents and support a document syntax that allows automatic translation of documents conforming to commonly used SGML document type definitions into LaTeX documents—this syntax will therefore, for example, support the SGML concepts of 'attribute' (or 'named argument') and 'short reference', in such a way that these can be easily linked to the corresponding SGML features.
- LaTeX3 will be designed as an open system and, like the present version, it will be usable with any standard TeX system and will thus be available on a very wide range of platforms.
- Its highly modular design will provide a system that is flexible and extensible, with well-defined and fully documented interfaces.
- The code itself will also be thoroughly documented and the modular design will help to make the system easy to maintain and enhance.
- We shall also provide extensive catalogues containing many examples that are carefully designed to make the learning time for new users (including designers, editors and programmers) as short as possible.

4 Available now!

In some important areas, the extra facilities of LaTeX3 are already available to current LaTeX users.

- The New Font Selection Scheme (NFSS) is now in widespread use, providing very general and powerful tools for setting up and accessing

all available fonts. A new version (NFSS2) has recently been released, which extends the flexibility of the system in the following areas of font management.

  - Scalable fonts, e.g., support for PostScript fonts.
  - Encodings, i.e., support for multilingual documents allowing change of font encoding (code page) within a document.
  - Math symbols, facilitating access to a wide range of symbols.
  - Math typesetting using any suitable family of fonts, eg Lucida or Adobe Times.

- With $\mathcal{AMS}$LaTeX [1] the capabilities of LaTeX for mathematical typesetting have reached at least the standard of $\mathcal{AMS}$TeX.
- Work by Frank Mittelbach and David Carlisle, together with valuable suggestions by several others, has made available more sophisticated tabular processing [15].
- Various forms of multiple-column formatting are also now available [14].
- Johannes Braams [2] has produced a new version of the 'Babel system' to support a wide range of languages.

5 New features

LaTeX3 will provide much new and enhanced functionality in addition to the above facilities; and its 'LaTeX3 programming language' will enable it to be further extended in a controlled way.

Here are further details of some of the many planned improvements.

- A robust author interface: providing interactive error recovery linked to an on-line help system.
- Languages: the typesetting will be customisable for use with different languages (as will the whole system, e.g. the error/help components). In particular, there will be support for mixed language documents and for multiple languages within one paragraph to be correctly hyphenated.
- Table formatting: a large number of extensions in this area will be available, including:
  - multi-page tables;
  - automated column width calculation;
  - a variety of designs for ruled tables;
- Float handling: the major problem in processing floats (e.g. tables and figures) is the precise specification of the designer's rules concerning how to position them. The new system will make it possible to implement a large range of such positioning algorithms, including the requirements of multi-column formatting, whilst also supporting explicit positioning commands.
- Many of the non-typesetting aspects of document processing will be automated, providing a flexible interface to cover a wide range of requirements and styles in the following areas:
  - citations and bibliographies;
  - cross-references;
  - indexing, etc.;
  - tables of contents, etc.;
  - multiple marks.
- The enhanced functionality in the area of mathematical typesetting will include:
  - alignments in displayed formulas;
  - alphabets, symbols, embellishments;
  - commutative (arrow) diagrams.

  These will extend many of the features which are already available in the current version of $\mathcal{AMS}$LaTeX [1].

  The system will also support the elements defined in the 'standard SGML DTD for mathematical expressions' which is currently under development [32].
- The typesetting requirements of many other areas will also be addressed—some examples:
  - technical documentation (e.g. offset layout, change bars);
  - academic publishing in the humanities (e.g. critical text editions);
  - structural formulas in chemistry;
  - integration of graphical structures, including shading and colour;

6 Interfaces

As we said earlier, the design specification interface is probably the most significant single new feature of LaTeX3. It will have two clearly separated parts:

- the creation of the generic mark-up (e.g. environments) used by the author in creating the LaTeX form of the document;

– the specification of how (in SGML language) the document elements will be formatted.

This will simplify the production of different layouts for the same document type. It will also enable LaTeX documents to be created and modified by structured document editors, such as Arbortext Publisher and Grif. Indeed, these two parts are direct analogues of the following components of the Grif system: the 'generic structure definition (or S) language' and the 'presentation description (or P) language' [4].

However, in order to support the requirements of high quality typesetting, the LaTeX3 equivalent of the P language will need to be even richer than the Grif language. Despite the complexity of the task, this part of the 'designer interface' will help to make the following straightforward:

- specification of a wide variety of typographic design rules;
- linking of the elements in a document type to the desired formatting;
- specifying and modifying all of the parameters that influence the layout.

Another important interface will be the LaTeX3 programming language, used for producing enhancements and extensions: it will be an entirely new language based on data structures and operations suited to the kind of programming required by document processing applications and to the expression of visual components of the layout process. Built on this language there will be high-level generic functions that allow the straightforward expression of common layout components.

7 Resources

The majority of the work, including conceptualisation, modelling, prototyping, implementation and testing, is being undertaken by a dozen individuals under the technical direction of Frank Mittelbach. This work is, in most cases including that of the technical director, done entirely in their spare time and so involves, as you can imagine, a lot of enthusiasm to keep the project alive. A large number of other individuals and organisations have contributed in one way or another to the effort, and we are confident that this number will continue to rise.

There are, nevertheless, many other tasks still to be done in support of the LaTeX3 project. These can be worked on concurrently with the development of the LaTeX3 kernel system. Furthermore, some of these tasks require special expertise not found among the core programming team. Initial research, analysis, and other work on these tasks by volunteers will, we are sure, greatly speed up the process of integrating a number of desirable features into LaTeX3.

For this reason a 'volunteer task list' has been set up: this describes briefly the individual tasks, which require a wide variety of expertise and time involvement. It will be updated at regular intervals and a copy is available, via anonymous `ftp` from the following sites:

- `niord.shsu.edu`,
  directory `[fileserv.vol-task]`;
- `ftp.uni-stuttgart.de`,
  directory `soft/tex/vol-task`.

For access via mail server, send mail to `fileserv@shsu.bitnet`, with no subject line and in the body write: `sendme vol-task`, or to `mail-server@rus.uni-stuttgart.de`, with no subject line and in the body write: `send soft/tex/vol-task/vol-task.tex`. If you are unable to retrieve a copy via electronic networks, please contact Chris Rowley.

To provide a means of communication with a large number of LaTeX users an electronic mailing list has been installed at Heidelberg. To subscribe to this list send a mail message to `listserv@vm.urz.uni-heidelberg.de`, with one line as the body of the message (substituting your own names):

```
subscribe LaTeX-L First-name surname
```

One of the major, and growing, problems is how to bring people from all over the world together to discuss the open questions and find new solutions. It is important that these meetings involve people from outside the project since we very much need the views and experience of typesetters, designers, publishers, etc. to help eliminate the flaws in the system and to find new and better solutions.

We have so far held two 'open workshops', in London, UK and Boston, USA; these were hugely successful and showed that further workshops of this kind are essential if we are to provide LaTeX3 with a good designer interface.

It is now clear that our ability to maintain the current progress will depend on adequate financial support being available for the following purposes:

- enhancement of computer equipment and software for the core development team;
- purchase of books on typography and other related subjects;
- essential expenses (travel, accommodation, etc.) for meetings of the project's core development team; and also for meetings with others, for example: testers of early versions; publishers; typographic designers; suppliers of related software, etc.

References and Bibliography

This bibliography contains some items which are not specifically referenced in this article: these contain further information about the LaTeX3 project. It also contains entries concerning BibTeX, which will be reimplemented and enhanced by Oren Patashnik for use with LaTeX3.

[1] American Mathematical Society, Providence, Rhode Island. *$\mathcal{A}\mathcal{M}\mathcal{S}$LaTeX Version 1.1 User's Guide*, December 1990.

[2] Johannes Braams An update on the babel system *TUGboat*, 14, to appear 1993.

[3] Malcolm Clark. What is the LaTeX3 project? *TeX and TUG NEWS*, 1(1):4, February 1992.

[4] Grif S.A, St Quentin en Yvelines. *Grif Languages: Grif 2.2*, 1992.

[5] Mary Guenther, editor. *TeX 90 Conference Proceedings*, March 1991. Published as TUGboat 12#1.

[6] Alan Honig. LaTeX3, TUG and You. *TeX and TUG NEWS*, 2(1):2–4, July 1992.

[7] Donald E. Knuth. Computers & Typesetting. Addison-Wesley, Reading, Massachusetts, 1986.

[8] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, Massachusetts, 1986.

[9] Helmut Lenzing. The euromath project. *Euromath Bulletin*, 1(1):13–20, 1992.

[10] Frank Mittelbach. LaTeX 2.10. In Lincoln K. Durst, editor, *1990 Conference Proceedings*, page 444, September 1990. Published as TUGboat 11#3.

[11] Frank Mittelbach. LaTeX2.09 → LaTeX3. *TeXline*, (14):15–18, February 1992.

[12] Frank Mittelbach. LaTeX3. *Die TeXnische Komödie*, 4(2):15–22, August 1992.

[13] Frank Mittelbach. LaTeX3 project. *TeX Gebruikers Group*, 92(1):87–90, May 1992.

[14] Frank Mittelbach. The `multicol` package. Distribution of style options that allow multi-column layout with LaTeX, May 1992.

[15] Frank Mittelbach and David Carlisle. The `array` package. Distribution of style options that extend LaTeX's array and tabular facilities, May 1992.

[16] Frank Mittelbach and Chris Rowley. The LaTeX3 project fund. *Die TeXnische Komödie*, 3(4):13–15, December 1991.

[17] Frank Mittelbach and Chris Rowley. LaTeX209 → LaTeX3. *TUGboat*, 13(1):96–101, April 1992.

[18] Frank Mittelbach and Chris Rowley. The LaTeX3 project fund. *TeX and TUG NEWS*, 1(1):5–6, February 1992.

[19] Frank Mittelbach and Chris A. Rowley. The pursuit of quality—how can automated typesetting achieve the highest standards of craft typography? In C. Vanoirbeek and G. Coray, editors, *Electronic Publishing '92*, pages 260–273, Cambridge, April 1992. Cambridge University Press.

[20] Frank Mittelbach, Chris Rowley and Michael Downes. Volunteer work for the LaTeX3 project. LaTeX3 project paper, September 1992.

[21] Frank Mittelbach and Rainer Schöpf. With LaTeX into the nineties. In Christina Thiele, editor, *1989 Conference Proceedings*, volume 10#4 of TUGboat, pages 681–690. TeX Users Group, December 1989.

[22] Frank Mittelbach and Rainer Schöpf. LaTeX dans les années 90. *Cahiers GUTenberg*, (6):2–14, July 1990.

[23] Frank Mittelbach and Rainer Schöpf. Towards LaTeX 3.0. In Guenther [5], pages 74–79. Published as TUGboat 12#1.

[24] N.A.F.M. Poppelier. SGML and TeX in scientific publishing. In Guenther [5], pages 105–109. Published as TUGboat 12#1.

[25] David Rhead. Could LaTeX do more for chemists? *TeXline*, (12):2–4, December 1990. Suggestions for LaTeX3.

[26] David Rhead. Towards BibTeX style-files that implement principal standards. *TeXline*, (10):2–8, May 1990.

[27] David Rhead. How might LaTeX3 deal with citations and reference lists? *TEXline*, (13):13–20, September 1991. Suggestions for LaTeX3.

[28] Chris Rowley. LaTeX209 → LaTeX3: an update. In Mimi Burbank, editor, *1992 Annual Meeting Proceedings*, pages 390–391, October 1992. Published as TUGboat 13#3.

[29] Michael D. Spivak. *LAMSTEX The Synthesis*. The TEXplorators Corporation, Houston, 1989.

[30] M. D. Spivak, Ph.D. *The Joy of TEX, A Gourmet Guide to Typesetting with the AMSTEX macro package*. American Mathematical Society, Providence, Rode Island, second edition, 1990.

[31] Eric van Herwijnen. *Practical SGML*. Wolters Kluwer Academic Publishers, Dordrecht, 1990.

[32] Björn von Sydow. The design of the euromath system. *Euromath Bulletin*, 1(1):39–48, 1992.

[33] Reinhard Wonneberger. Structured document processing: the LaTeX approach. *Computer Physics Communications*, (61):177–189, 1990.

[34] Reinhard Wonneberger and Frank Mittelbach. BibTeX reconsidered. In Guenther [5], pages 111–124. Published as TUGboat 12#1.

Projekt LaTeX3

Wybór (nie uzgodniony z autorami) i tłumaczenie Bogusław Lichoński.

Wstęp

Pragnę przybliżyć wszystkim projekt powstania nowej wersji LaTeX-a, która dzięki nam wszystkim może wkrótce się narodzić.

LaTeX3 jest – podobnie jak LaTeX – programem dostępnym za darmo, służącym do komputerowego formatowania publikacji na najwyższym typograficznym poziomie. LaTeX3 będzie kolejnym superrozwinięciem TeX-a, wytworzonym przez grupę ochotników pod moim technicznym nadzorem.

3. Główne cele

Oto cele jakie postawiliśmy sobie do realizacji:

- LaTeX3 będzie spełniał wymagania każdego typografa pod względem stylu i wyglądu publikacji
- Autorzy uzyskają łatwy w obsłudze lecz wykonujący skomplikowane operacje system. Przy czym wykonanie nawet bardzo złożonych operacji będzie w pełni kontrolowalne przez użytkownika
- Publikacja – podobnie jak w LaTeX-u – będzie miała określoną strukturę, wzbogaconą o mechanizm pozwalający zautomatyzować tłumaczenie dokumentów dostosowanych do często używanego obecnie standardu SGML
- Nowy LaTeX3 będzie systemem otwartym, podobnie jak wersja obecna działał będzie z dowolną implementacją TeX-a
- Każdy użytkownik będzie miał możliwość dowolnego rozszerzania systemu LaTeX3 dzięki możliwości tworzenia własnych makr
- Zamierzamy przygotować cały zbiór przykładowych publikacji, które ułatwią poznanie systemu LaTeX3

4. Dostępne już teraz

Ponieważ prace trwają, więc mamy już pewne osiągnięcia. Oto one:

- Nowy System Wyboru Fontów (*New Font Selection Scheme*) jest już gotowy i używany. Dostarcza on bardzo ogólnych i „mocnych" narzędzi, które pozwalają na wykorzystanie:
  - fontów skalowalnych
  - wielu różnych alfabetów dzięki możliwości zmiany stron kodowych (*code page*) wewnątrz publikacji
  - całego zbioru nowych symboli matematycznych
  - dowolnej rodziny np. Lucida czy Adobe Times fontów do składu formuł matematycznych
- Nowy mechanizm tabulacyjny [15]
- Nowa forma składu wieloszpaltowego [14]
- Włączenie nowej wersji systemu Johannesa Braamsa [2] "Babel" wspomagającego używanie wielu różnych języków narodowych

5. Zamiary

Skoro nowy LaTeX3 rozwija się i nie jest jeszcze ukończony, mamy wiele do zrobienia, choćby:

- Nowoczesny i wydajny interfejs użytkownika

- Możliwość konfigurowania składu ze względu na różne języki w zakresie jednego akapitu z uwzględnieniem poprawności dzielenia wyrazów
- Nowoczesny system składania tabel, bez konieczności obliczania szerokości kolumn itp.
- Dowolne umieszczanie tabel i rysunków w składzie wieloszpaltowym
- Zautomatyzowane w pełni zostanie zastosowanie odwołań do cytatów i numerów stron, tworzenie bibliografii, skorowidza, spisu treści itp.
- Zwiększenie funkcjonalności w zakresie składu matematyki, czyli dowolne justowanie wyeksponowanych formuł matematycznych ($$), nowe alfabety, symbole i akcenty oraz diagramy komutacyjne. System będzie także zawierał elementy zdefiniowane w standardowym SGML DTD dla wyrażeń matematycznych [32]
- Bogata dokumentacja

6. Interfejsy

Jak wspomniałem wcześniej nowy interfejs użytkownika może być jedną z największych atrakcji systemu LaTeX3. Interfejs ten będzie miał dwie zasadnicze części:

- tworzenie ogólnych odwołań będących elementami formatowanego dokumentu
- wskazówki w jaki sposób elementy publikacji powinny być formatowane (w języku SGML)

Innym ważnym interfejsem będzie język programowania LaTeX3 używany do tworzenia rozszerzeń. Będzie to język bazujący na strukturach danych i operacjach dostosowanych do sposobu programowania danego typu publikacji.

Poszukuję wolontariuszy

Jak wspominałem system LaTeX3 tworzony jest i będzie przez grupę ochotników, których aktualną listę uzyskać można przez Internet. Dokładny adres znaleźć można w angielskim oryginale artykułu.

Uciecha z TeX-a

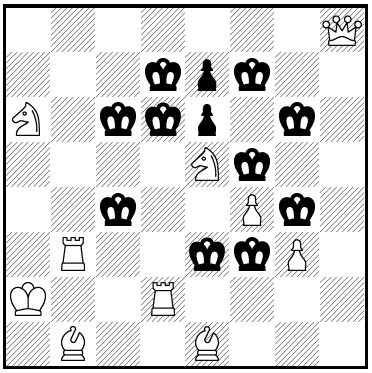
Krzyżówka

Rozwiązanie z poprzedniego numeru (jedno z wielu możliwych):

\	H	G	L	U	E
\	O	U	T	E	R
\	H	S	I	Z	E
\	S	T	R	U	T

Szachy

Przedstawiona w poprzednim numerze żartobliwa kompozycja autorstwa G. R. Reichhelma z 1861 r. (Jenoő Bán, *The tactics of end-games*, Corvina Press 1963) wydaje się łatwa, ale dla porządku...
po ♘ d3–e5, wszystkie czarne króle są zamatowane.



Jak nazywa się ten font?

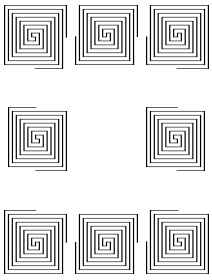
Są to litery XYZ fontu yinit zMETAFONTowanego przez Yannisa Halarambousa, dostępnego w wielu archiwach.

METAFONT

Ramki spiralne

Bogusław Jackowski*
i Tomasz Przechlewski**

Projektowanie ramek to jedno z prostszych zadań, które można zrealizować wykorzystując METAFONT-a. Przedstawione niżej projekty ramkek, z uwagi na zwartość kodu, są prezentowane w całości. Z tego też powodu dodano w kilku miejscach komentarze. Do wykreślenia spiralnej ramki należy zaprojektować osiem następujących znaków:



Program METAFONTowy rozpoczyna się (pomijając sakramentalne `mode_setup`) od nadania wartości czterem parametrom:

- `px` określa szerokość linii poziomych,
- `py` określa szerokość linii pionowych,
- `leap` odległość między zwojami,
- `spir` liczba elementów projektowanego znaku.

```
mode_setup;
px=round(.4pt); % pen x-dimen
py=round(1.2pt); % pen y-dimen
leap=round(2.5pt); spir=15;
```

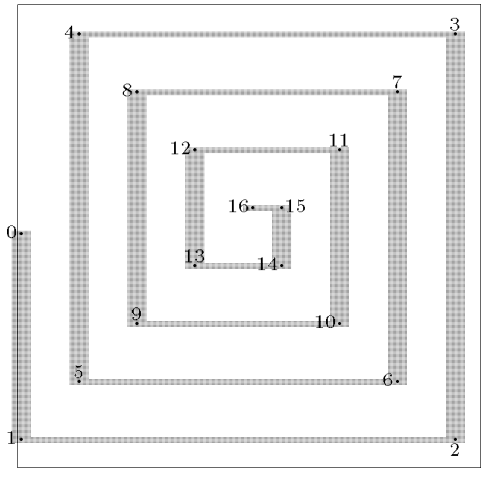
Na podstawie tych informacji obliczana jest wielkość znaku. Jest ona zależna od rozdzielczości urządzenia, na które ramka jest generowana. Przykładowo dla urządzenia o rozdzielczości 300dpi `designsize`$\approx$ 38,54 pt, dla 600dpi `designsize`$\approx$ 40,47 pt, a dla rozdzielczości 72dpi nawet $\approx$ 32,12 pt. Dzięki tej procedurze `designsize` (określający w tym przypadku szerokość i wysokość znaków) jest krotnością piksela. Chodziło w tym wszystkim o uniknięcie zjawiska przesuwania znaków przez sterowniki przy ich pozycjonowaniu. Gdy szerokość znaku nie stanowi całkowitej wielokrotności rozmiaru piksela występuje zjawisko określane potocznie jako „pływanie" znaków. Eliminując pływanie gwarantujemy, że sąsiadujące ze sobą znaki będą względem siebie ustawiane przez sterownik identycznie, niezależnie od ich położenia na stronie.

```
font_size:=((spir+1)*leap)/hppp;
message "designsize =
        " & decimal designsize & "pt";
message "";
```

Wszystkie znaki są generowane poprzez wywołanie makrodefinicji `fix_spir`. Począwszy od brzegu kwadratu linia labiryntu jest zakręcana do środka.

```
vardef fix_spir@# text t =
% |t| is an appropriate rotation
 n:=0; pair uv;
% wrapping:
 z0=(.5px,.5h) t;
 z[incr n]=(.5px,leap-.5px) t;
 uv:=(right t)-(origin t);
 for i:=spir*leap step -leap until leap:
  z[incr n]=z[n-1]+i*uv;
  uv:=uv rotated 90;
 endfor
```

Na tym etapie wygenerowany znak wyglądałby następująco:



Ale to nie koniec. Teraz linia jest rozwijana od środka z powrotem do brzegu, za pomocą następnej instrukcji iteracyjnej.

*: METAFONTologia
**: Reszta

```
20. % unwrapping:
21.  z[incr n]=z[n-1]+leap*uv;
22.  uv:=uv rotated -90;
23.  for i:=leap step leap until
24.       (spir if (str @#)="'": +1 fi)*leap:
25.   z[incr n]=z[n-1]+i*uv;
26.   uv:=uv rotated -90;
27.  endfor
28.  z[incr n]=if (str @#)="'":
29.     (.5w,-.5px) else: (w+.5px,.5h) fi t;
30.  labels(range 0 thru n);
```

Na końcu makrodefinicji `fix_spir` definiowana jest ścieżka `skeleton` zawierająca wszystkie, niezbędne do wykreślenia znaku punkty.

```
31.  path skeleton;
32.  skeleton=z0 for i:=1 upto n:
33.                  --z[i] endfor;
34. enddef;
```

Druga makrodefinicja rysuje zwoje znaku „przygotowane" w poprzedniej:

```
35. def draw_spir =
36.  pickup pensquare xscaled py;
37.                    yscaled px;
38.  draw skeleton;
39. enddef;
```

a trzecia, służąca wyłącznie do skrócenia kodu, określa transformację, przekazywaną jako drugi argument w makrodefinicji `fix_spir`.

```
40. def trs(expr a)(expr c,d) =
41.     rotatedaround ((.5w, .5h), a)
42.     shifted (c,d);
43. enddef;
```

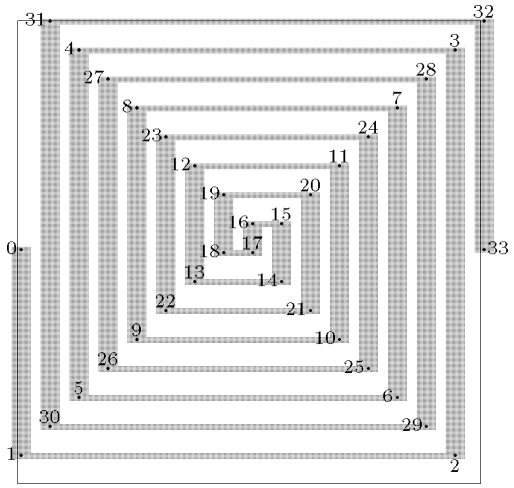
Górny środkowy znak jest zdefiniowany następująco:

```
44. beginchar("0",((spir+1)*leap)/hppp,
45.           ((spir+1)*leap)/hppp,0);
46.  fix_spir;
47.  draw_spir;
48. endchar;
```

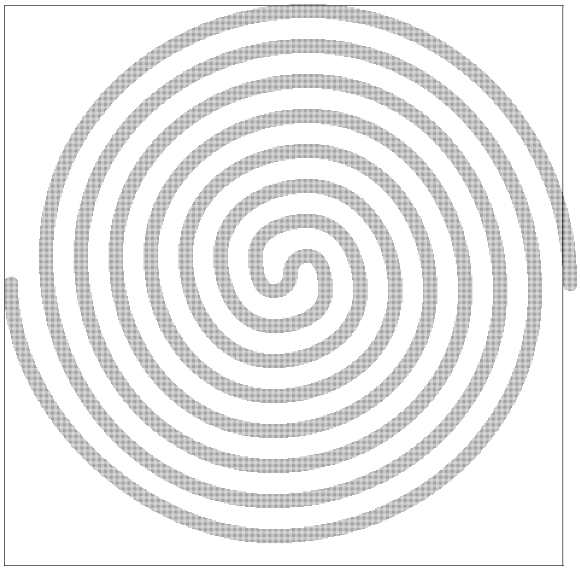
Pozostałe znaki są definiowane identycznie za wyjątkiem linii 46 zawierającej makro `fix_spir` (oczywiście zmienia się także pierwszy argument makra `beginchar` — pozycje poszczególnych znaków są jednak zupełnie nieistotne i mogą być dowolnie ustalone, w zależności od *gustu* programującego). Makro `fix_spir` ma dwa parametry, z których pierwszy jest typu `suffix` a drugi `text`. Suffiksem jest albo `prim '` albo jest on pusty (pełni on tutaj rolę zmiennej logicznej — por. wiersze 24 i 28). Argument tekstowy w dwóch przypadkach także jest pusty, a w pozostałych jest nim odpowiedni obrót wokół środka znaku, oraz przesunięcie.

- górny prawy `fix_spir'`.
- boczny lewy `fix_spir  trs(-90,px,0);`
- dolny prawy `fix_spir' trs(-90,px,0);`
- dolny środkowy `fix_spir trs(-180,px,-px);`
- dolny lewy `fix_spir' trs(-180,px,px);`
- boczny prawy `fix_spir trs(-270,0,-px);`
- górny lewy `fix_spir' trs(-270,0,-px);`

Efekt końcowy dla górnego środkowego znaku prezentuje się (w dużym powiększeniu) następująco:



Wystarczy niewielka modyfikacja (jaka?) końcowej części definicji `skeleton` i makrodefinicji `draw_spir` aby otrzymać następujący ciekawy rezultat:



Teraz kilka słów na temat wykorzystania fontu w TEXu. Niech ilustracją tego będzie ogłoszenie o konkursie. Oto kod TEXowy, nieco trickowy:

```
\font \S spirama
\font \BF pldunh10

\def\0{\hbox{\phantom{\S0}}}

\newbox\tline % górna ramka
\newbox\bline % dolna ramka
\newbox\rline % prawa ramka
\newbox\lline % lewa ramka
\newbox\frame
\newbox\text

\setbox\tline
 \hbox{\S 000000000a}
\setbox\rline
 \vbox{\offinterlineskip\def~#1{\hbox{#1}}%
 \S~1~1~1~1~1~1~1~1~1~1~1~1~1~1~b}
\setbox\bline\hbox{\S c222222222}
\setbox\lline
\vbox to\ht\tline{\offinterlineskip
 \def~#1{\hbox{#1}}%
 \S~d~3~3~3~3~3~3~3~3~3~3~3~3~3~3\vss}

\setbox\frame\vbox{%
\hbox{\copy\lline\copy\tline}
\hbox{\copy\bline\copy\rline}}
```

Teraz znając wymiary ramki, tekst zawieramy w pudełku o identycznych wymiarach:

```
\long\def\Lcenterline#1{\line{\hss#1\hss}}

\setbox\text\vbox to\ht\frame{%
\hsize=\wd\frame\baselineskip12pt
\vfil
\Lcenterline{\vbox{\hsize=.7\hsize%
\centerline{\BF Konkurs na najpiękniejszą}
\centerline{\BF ramkę lub inny...}}}
...
\vfil}
```

Pudełko \frame zawierające ramkę kopiujemy, zaś operacja \llap{\box\text} powoduje nałożenie pudełka z tekstem na pudełko ramki:

```
\noindent\copy\frame\llap{\box\text}
```

Pełny tekst ogłoszenia o konkursie obok. REDAKCJA OCZEKUJE LAWINY ZGŁOSZEŃ. Do dzieła!

Konkurs na najpiękniejszą ramkę lub inny znak wykonany METAFONTem

Redakcja Biuletynu ogłasza konkurs na najładniejsze znaki graficzne wykonane METAFONT-em. • Znaki mogą być wszelakiego rodzaju: ramki, loga, itp. • Termin nadsyłania propozycji 31 Stycznia 1994. • Najlepsze zostaną zaprezentowane w Zeszycie 2 (1) oraz nagrodzone książkami METAFONT-owo-TEX-owymi.

Znak powinien być oryginalnym pomysłem autora. • Można zgłaszać wiele prac. • Styl programu METAFONT-owego będzie ważył znacznie na ocenie. • Do źródła METAFONT-owego należy dołączyć opis programu.

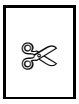
Zgłoszenia kierować na adres redakcji

The Best Border Design or Other Dingbat Character Competition

GUST announces a competition for the best dingbat character created with METAFONT. • A character could be: border design, logo, etc. • The deadline for submissions: 31 January 1994. • The best works will be published in Zeszyt 2 (1) of GUST and rewarded with books on TEX and METAFONT.

The character should be of the author's original design. • Each submission can contain several characters. • METAFONT programming style will be considered while awarding prizes • Each submission shall be accompanied by description explaining its meaning and commenting interesting parts of METAFONT source code.

Submissions should be posted to the editors of GUST bulletin.



TEXnologia na codzień

Everyday TEXnology

Włodek Bzyl

1. Zadanie na dzisiaj. Jak na stronach na których znajdą się całostronicowe wstawki (*page insertions*) skasować główkę i stopkę? Na przykład tak składa się strony, na których są umieszczone duże rysunki lub szerokie tabele (obrócone o 90°).

Aby złożyć tekst bez stopki (zawierającej zwykle numer strony), korzystamy z komendy \nopagenumbers. Główka strony w formacie plain jest pusta. Jeśli tylko jedna strona ma być złożona bez stopki, np. pierwsza strona książki lub pierwsza strona rozdziału, to «szybkim» rozwiązaniem może być umieszczenie komendy \nopagenumbers w grupie, tzn. między nawiasami { i }, zawierającej odpowiednio dużo tekstu, np. tyle ile TEX potrzebuje aby złożyć stronę i cały akapit na następnej stronie. W przypadku kilkunastu takich wstawek stosowanie «szybkiego» rozwiązania zaproponowanego powyżej może nam zająć dużo czasu.

2. Rozpoznanie sytuacji. Główka (*heading*), stopka (*footline*), wstawki (*insertions*) są dołączane do strony złożonej przez budowniczego strony (*page builder*) komendami TEX-a zawartymi w rejestrze \output. Ciąg tych komend nazywamy procedurą wyjścia (*output routine*).

Komendy składające główkę (stopkę) są zawarte w rejestrze \headline (\footline). Na przykład przy pomocy podstawienia

```
\headline={\hrulefill}
```

uzyskuje się główkę składającą się z kreski poziomej. Komenda \nopagenumbers to synonim na

```
\footline={\hfil}.
```

Dla przypomienia, w formacie plain nadaje się wartość

```
\output={\plainoutput}.
```

Makro \plainoutput zdefiniowane jest następująco

```
\def\plainoutput{
 \shipout\vbox{
  \makeheadline\pagebody\makefootline}%
 \advancepageno
 \ifnum\outputpenalty>-20000
 \else\dosupereject\fi}
```

Komenda \shipout<pudełko> wysyła złożone <pudełko> do pliku dvi.

3. Proponowane rozwiązanie można zaliczyć to kategorii chytrych sztuczek (*dirty tricks*). „Sztuczka" polega na «oznakowaniu» wstawki i modyfikacji procedury wyjścia w taki sposób, aby zostały podjęte odpowiednie działania jeśli oznakowana wstawka zostanie wykryta przez procedurę wyjścia.

Do znakowania wstawek posłuży kreska o długości \maxdimen i zerowej wysokości. Taką kreskę będziemy nazywać puzonem – \trombone.* Złożenie tej kreski daje pudełko szerokości \maxdimen. W rozwiązaniu zakładamy, że pojawienie się pudełka maksymalnej szerokości to efekt działania puzona.

```
\def\trombone{
 \hrule height0pt width\maxdimen}
```

4. Wprowadzamy nową komendę \specinsert za pomocą której będziemy jednocześnie składać i znakować wstawki.

```
\def\specinsert{\pageinsert\trombone}
```

5. Aria na puzonie. Przed wywołaniem procedury wyjścia TEX umieści naszą wstawkę w pudełku o numerze równym zawartości zmiennej \topins. Szerokość tego pudełka TEX umieszcza w zmiennej \wd\topins. Jest ona równa \maxdimen tylko wtedy, kiedy składamy wstawkę \specinsert. Wystarczy więc dokonać następującej modyfikacji.

```
\def\plainoutput{
 \ifdim\wd\topins=\maxdimen
  \shipout\vbox{\pagebody}
 \else
```

---

*: Nie-chytre rozwiązania postawionego wyżej problemu również istnieją. Pomysł na oznakowanie wstawki «puzonem» pochodzi od Piotra Pianowskiego i Bogusława Jackowskiego.
Patrz też *The TEXbook* strona 416 (makrodefinicja \titlepage), oraz strona 400 (punkt 8).

```
 \shipout\vbox{
  \makeheadline\pagebody\makefootline}
\fi%
\advancepageno
\ifnum\outputpenalty>-20000
\else\dosupereject\fi}
```

6. I to by było na tyle. A teraz kilka słów na temat *Literate Programming*. LP to styl programowania autorstwa Donalda Knutha, natomiast FWEB jest dialektem WEB-a. Przy pomocy FWEB-a możemy również pisać programy w języku C, C++, FORTRAN. Autor chciałby podziękować Johnowi Krommesowi, autorowi systemu FWEB, za cenne wskazówki na temat trybu TEX w jego programie.

Powyższy tekst napisano w FWEB-ie i stanowi on zawartość pliku 3whole.web. FWEB to dwa programy: fweave i ftangle. Plik 3whole.web został przetworzony przez każdy z tych programów. Jako rezultat otrzymaliśmy zbiory: 3whole.tex, 3whole.sty, index.tex, contents.tex.

Pierwszy z tych plików został złożony TEX-em i po dodaniu nagłówka i wydrukowaniu jest tekstem, który czytasz.

Zawartość pliku 3whole.sty została złożona poniżej.

```
% FTANGLE v1.30a, created with IBM-PC/DOS
% on "Friday, June 25, 1993 at 10:01."
%
% COMMAND LINE:
% "C:\EMTEX\FTANGLE.EXE -v -# 3whole"
% RUN TIME: "July 7, 1993 at 18:08."
% WEB FILE: "3whole.web"
% CHANGE FILE: (none)

\def\trombone{
 \hrule height0pt width\maxdimen}

\def\specinsert{\pageinsert\trombone}

\def\plainoutput{
 \ifdim\wd\topins=\maxdimen
  \shipout\vbox{\pagebody}
 \else
  \shipout\vbox{
   \makeheadline\pagebody\makefootline}
 \fi%
 \advancepageno
 \ifnum\outputpenalty>-20000
 \else\dosupereject\fi}
```

7. Przykład wykorzystania opisanych makr.

```
\input 3whole.sty
...
% prosta główka i stopka
\headline{\hrulefill}
\footline{\hfil--\folio--\hfil}

% font do wstawek
\font\big=plinch
...
jakikolwiek tekst
...
\specinsert
\big JK \vfill
\endinsert
...
więcej tekstu
...
\bye
```

8. Indeks. Większość haseł została automatycznie umieszczona w indeksie przez program fweave.

Informacja

Zasoby TEX-owe w rozległych sieciach komputerowych*

Elżbieta Kuczyńska

Wymiana dokumentów poprzez rozległe sieci komputerowe wymaga ustalenia standardowej postaci przesyłanych plików tekstowych. W sieciach naukowych takim standardem stał się TEX. Dokumenty sformatowane w TEX-u, jako pliki tekstowe ASCII, mogą być łatwo przesyłane pocztą elektroniczną. Standardem w sieciach nadal pozostaje transmisja 7–bitowa, a więc TEX jest tu najlepszym rozwiązaniem przy przesyłaniu tekstów nie-angielskich. Oprogramowaie systemu TEX jako rozpowszechniane bez ograniczeń (public domain), może być bezpłatnie udostępniane użytkownikom sieci. Z tych powodów obserwujemy stały rozwój aplikacji sieciowych, umożliwiających gromadzenie i rozpowszechnianie dokumentów TEX-owych, oraz oprogramowania do ich tworzenia. W tym opracowaniu przedstawimy przykłady tego typu usług.

Elektroniczne Biblioteki Preprintów

Elektroniczna Biblioteka Preprintów (*Electronic Preprint Library* — EPL), jest to pełnotekstowa baza danych zawierająca preprinty z danej dziedziny wiedzy, w tym przypadku z wybranych działów fizyki, astronomii i matematyki. Opisywana tu baza posadowiona jest na komputerze o adresie internetowym: `babbage.sissa.it`, należącym do Międzynarodowej Szkoły Studiów Podyplomowych w Trieście (Włochy).

Dla osób zainteresowanych korzystaniem z zasobów tej bazy podajemy nazwy archiwów składowych, występujące jako identyfikatory w adresach elektronicznych:

`astro-ph` — astrofizyka
`cond-mat` — fizyka ciała stałego
`funct-an` — analiza funkcjonalna
`gr-qc` — ogólna teoria względności
`hep-ph` — fizyka doświadczalna cząstek elementarnych
`hep-th` — fizyka teoretyczna cząstek elementarnych
`nucl-th` — teoretyczna fizyka jądrowa

Użytkownicy bazy mogą za pomocą odpowiednich komend formułować zlecenia wyszukiwawcze i przesyłać je do bazy w elektronicznym liście. Zlecenie umieszcza się w polu „subject" listu. Na wstępie zaleca się wydanie komendy `help`, celem uzyskania informacji o składni komend wyszukiwawczych. Można też komendą `get bighelp.tex` zamówić obszerniejszy podręcznik, sformatowany w TEX-u.

Każda praca znajdująca się w bazie, posiada swój numer identyfikacyjny w postaci yymmnnn (yy — 2 cyfry oznaczające rok, mm — 2 cyfry oznaczające miesiąc, nnn — trzycyfrowy numer kolejny publikacji w danym miesiącu).

Po odnalezieniu interesującej nas pracy, możemy zamówić jej pełny tekst komendą:

```
get yymmnnn
```

lub też tylko abstrakt, komendą:

```
get yymmnnn.abs
```

Możliwe jest regularne otrzymywanie abstraktów nowosprowadzonych prac, dzięki zapisaniu się na listę abonentów biuletynu informacyjnego dla użytkowników archiwum. Realizuje to komenda:

```
subscribe nazwa-podbazy imię nazwisko
```

Autorzy nadsyłający do bazy własne publikacje umieszczają tekst pracy w treści listu, a w polu `subject` wydają komendę

```
put nazwa-pracy.tex
```

Nadesłana praca otrzymuje automatycznie kolejny numer identyfikacyjny.

Użytkownicy sieci BITNET mogą korzystać z EPL wyłącznie za pomocą poczty elektronicznej, natomiast w ramach usług Internetu można pobierać prace z bazy, oraz umieszczać własne publikacje korzystając z anonymous `ftp`. Archiwa składowe EPL, są w tym przypadku podkatalogami katalogu wejściowego (*home directory*) usługi ftp. Nazwy tych podkatalogów są takie same, jak identyfikatory w adresach pocztowych.

Prace mogą być pisane w dowolnym spośród poszechnie używanych formatów TEX-owych, jak plain, LATEX, $\mathcal{AMS}$-TEX, RevTEX. Jeśli autor stosuje własne, niestandardowe makra to powinien je nadesłać wraz z tekstem pracy.

Tego typu elektroniczne archiwa powstały początkowo na potrzeby fizyki, a zasięg ich wykorzystania ograniczony był do wąskiego kręgu specjalistów. Można postawić pytanie, czy ta

*: Tekst wystąpienia wygłoszonego na Pierwszej Ogólnopolskiej Konferencji TEX-owej—Bachotek'93

forma rozpowszechniania publikacji jest w stanie konkurować z uznawanymi czasopismami, które publikują prace poddane uprzednio wnikliwej selekcji. Taka selekcja nie jest możliwa w przypadku archiwum w pełni zautomatyzowanego, (jak opisane powyżej). Można jednak utworzyć archiwum „moderowane", w którym prace zanim zostaną publicznie udostępnione są oceniane pod względem merytorycznym przez kompetentny zespół specjalistów.

GOPHER

Przykładową usługą sieciową, dostępną wyłącznie w sieci Internet, za pomocą której możemy uzyskiwać dostęp do zasobów TEX-owych jest GOPHER. Zakres tej usługi jest bardzo szeroki, udostępnia ona przeglądanie plików na odległych komputerach, dostęp do baz danych, węzłów anonymous ftp itp. Z punktu widzenia niniejszej publikacji interesujący jest dostęp poprzez GOPHER do archiwów oprogramowania TEX-owego, oraz dokumentów formatowanych w TEX-u. Takim archiwum zawierającym software do TEX-u dostępnym przez tą usługę jest np. popularna baza ARCHIE. Do korzystania z usługi niezbędne jest posadowienie na własnym hoście oprogramowania GOPHER CLIENT, natomiast GOPHER SERVER umożliwia udostępnianie w sieci własnych zasobów.

◇

Celem mojego wystąpienia nie mogło być oczywiście przedstawienie szczegółowej instrukcji korzystania z opisanych zasobów, ani też dokładne zaprezentowanie wszystkich możliwości wykorzystania TEX-a jako standardu dla plików tekstowych w sieciach rozległych. Zamierzałam jedynie wskazać, dostępne obecnie również w Polsce możliwości korzystania ze światowych zasobów TEX-owych. Osoby zainteresowane mogą skorzystać ze szczegółowego opracowania [1], w którym opisane jest korzystanie z EPL. Gorąco zachęcam czytelników „Biuletynu GUST" do samodzielnych prób w tym zakresie.

Literatura

[1] Bogumiła Rykaczewska–Wiorogórska, Elżbieta Kuczyńska, „Elektroniczne biblioteki i bazy informacyjne z dziedziny nauk fizycznych" — artykuł przewidziany do publikacji w Postępach Fizyki, czerwiec 1993.

Dodatek nadzwyczajny

Anonymous

Co to jest CTAN?

CTAN to skrót od *Comprehensive TEX Archive Network*. CTAN tworzą obecnie trzy komputery z posadowionym oprogramowaniem TEX-owym:

- `ftp.uni-stuttgart.de`
- `ftp.tex.ac.uk`
- `pip.shsu.edu`

Archiwa TEX-owe są takie same na każdym z komputerów. Co więcej układ katalogów jest taki sam. Wyrastają one z następujących korzeni:

- `soft/tex/`
- `pub/archive/`
- `tex/archive/`,

odpowiednio. Poniżej zobaczysz układ katalogów pierwszego poziomu.

```
└── soft
    └── tex
        ├── bibliography
        ├── dante
        ├── digests
        ├── documentation
        ├── dviware
        ├── fonts
        ├── graphics
        ├── help
        ├── indexing
        ├── languages
        ├── macros
        ├── misc
        ├── support
        ├── tex-implementors
        ├── systems
        ├── utilities
        ├── vol-task
        └── web
```

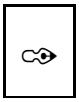
System emTEX znajdziesz w

`soft/tex/systems/msdos/emtex`

na pierwszym z komputerów i w

`pub/archive/systems/msdos/emtex`

na drugim z nich.



LaTeX

LaMeX-owe ABC*

Wiesław Pawłowski

Artykuł ten pomyślany jest jako początek nowej, miejmy nadzieję stałej, rubryki w Biuletynie, poświęconej LaMeX-owi, czyli polskiej wersji popularnego formatu TeX-owego o nazwie LaTeX.

Zacznijmy może od kilku słów na temat genezy samego LaTeX-a. Powstał on w połowie lat osiemdziesiątych na uniwersytecie w Palo Alto w Kalifornii. Jego twórcą jest znany informatyk amerykański Leslie Lamport. Pomysł na LaTeX-a zrodził się z obserwacji, że mimo niewątpliwych zalet TeX-a jego popularność była stosunkowo niewielka. Przyczyn tego zjawiska dopatrywał się Lamport w tym, że TeX (wraz z opracowanym przez Knuth'a formatem `plain` — patrz [Knuth 86]) był zbyt skomplikowany dla przeciętnego śmiertelnika. LaTeX miał być narzędziem, które umożliwiałoby korzystanie z TeX-a nawet zupełnym laikom. W zamyśle autora jego użytkownikami mieli być głównie ludzie z kręgów akademickich. Mimo pewnych wad i ograniczeń LaTeX-a, jego popularność stale rosła i w chwili obecnej jest on jednym z najpowszechniej używanych formatów TeX-owych na świecie. Podstawowym źródłem wiedzy na temat LaTeX-a jest książka [Lamport 85]. Ukazało się również jej polskie tłumaczenie — [Lamport 92].

Autorami adaptacji LaTeX-a do języka polskiego są Bogusław Jackowski i Marek Ryćko. LaMeX powstał niejako na zasadzie „produktu ubocznego" kilku lat pracy, którą autorzy włożyli w stworzenie polskojęzycznej wersji formatu `plain` — czyli opisanego w poprzednim numerze Biuletynu MeX-a ([Lichoński 93]). Było to możliwe, gdyż LaTeX w dużej mierze bazuje na nieznacznie zmodyfikowanej wersji `plain`-a, gdzie też znajduje się większość, zależnych od konkretnego języka, definicji i parametrów.

W dalszej części artykułu zajmiemy się omówieniem najważniejszych możliwości, które oferuje LaMeX, a które odróżniają go od formatu MeX. Należy przy tym podkreślić, że opisywane mechanizmy występują również w oryginalnym LaTeX-u. Jedyne różnice pomiędzy LaMeX-em, a LaTeX-em związane są z dostosowaniem do języka polskiego tego pierwszego. Postaramy się również odpowiedzieć na pytanie, do jakich zadań LaMeX się nadaje, a w jakich przypadkach należy raczej z niego zrezygnować na rzecz jakiegoś innego formatu.

[1] Artykuł ten jest skróconą i zmienioną wersją notatek do wykładu — [Pawłowski 93], jaki autor wygłosił na Pierwszej Ogólnopolskiej Konferencji TeX-owej w Bachotku, w kwietniu br.

1 Struktura manuskryptu LaMeX-owego

LaMeX narzuca dosyć rygorystyczne wymagania, co do układu przygotowywanego manuskryptu. Każdy manuskrypt LaMeX-owy składa się z dwóch podstawowych części — tzw. *preambuły* oraz *części głównej*.

W preambule umieszcza się polecenia, które określają różne parametry, od których zależy sposób, w jaki TeX przetwarzał będzie następnie cały manuskrypt. Wśród tych poleceń musi pojawić się co najmniej polecenie `\documentstyle`. Określa ono tzw. *styl*, zgodnie z którym przygotowywany będzie skład. Na przykład polecenie:

```
\documentstyle[11pt]{iarticle}
```

spowoduje, że manuskrypt będzie składany przez TeX-a zgodnie ze stylem *iarticle* (modyfikacja standardowego stylu *article* — czyli „artykuł"). W nawiasach prostokątnych podaje się tzw. *opcje*. Wprowadzają one pewne, niewielkie na ogół, modyfikacje do używanego stylu. W powyższym przykładzie opcja `11pt` oznacza, że do składu zamiast domyślnej czcionki 10-punktowej, użyta zostanie czcionka o wielkości 11 punktów. W preambule nie wolno umieszczać tekstu przeznaczonego do złożenia.

Część główna manuskryptu LaMeX-owego musi zaczynać się poleceniem `\begin{document}`, a kończyć — poleceniem `\end{document}`. Cały tekst przeznaczony do złożenia musi znajdować się między tymi dwoma poleceniami. Wszystko, co znajdzie się po poleceniu `\end{document}`, zostanie przez TeX-a pominięte.

Podsumowując: manuskrypt LaMeX-owy posiada następującą postać ogólną:

```
\documentstyle[opcje]{styl}
  dodatkowe polecenia
  określające parametry składu
\begin{document}
  treść manuskrypu
\end{document}
```

2 Podział logiczny manuskryptu

Teksty, z myślą o składaniu których LaTeX został stworzony, czyli wszelkiego rodzaju publikacje i opracowania naukowe, na ogół mają strukturę hierarchiczną — dzielą się na rozdziały, podrozdziały, punkty itd. Jedną z wielu zalet LaTeX-a jest wspieranie użytkownika w tworzeniu i utrzymywaniu takiej struktury.

Służą do tego celu polecenia:

```
\chapter   \subsection      \paragraph
\section   \subsubsection   \subparagraph
```

Tworzą one hierarchię podziału. Jej „głębokość" zależy od *stylu* używanego do składania manuskryptu. Obok wspomnianego już stylu `iarticle` — czyli „artykuł", dostępne są również style `ireport` — „raport" oraz `ibook` — „książka". Tak, jak to sugerują ich nazwy, style te przeznaczone są opracowywania coraz to obszerniejszych publikacji.

Dla stylów `ibook` i `ireport` największą jednostką w hierarchii podziału jest *rozdział*. Do zaznaczenia miejsca, w którym rozpoczynamy nowy rozdział służy polecenie `\chapter`.

W stylu `iarticle` największą jednostką w hierarchii podziału jest *punkt*. W nomenklaturze LaTeX-owej odpowiada mu polecenie `\section`.

Polecenia podziału mogą również powodować umieszczenie informacji o bieżącej jednostce podziału w spisie treści — o czym za chwilę.

Każda publikacja składana z użyciem LaTeX-a może być dodatkowo dzielona na *części*. Służy do tego polecenie `\part`. Nie jest ono jednak poleceniem podziału w tym samym sensie, w jakim są nimi `\section`, czy `\chapter`. Użycie `\part` jest bowiem opcjonalne. W przypadku „normalnych" poleceń podziału, o ile nie zależy nam na nieoczekiwanych efektach, zawsze powinniśmy zaczynać od najwyższej, dopuszczalnej w danym stylu, jednostki. Pisząc na przykład artykuł (tzn. używając stylu `iarticle`) i chcąc wprowadzić w nim podział powinniśmy zacząć od *punktu*. Spowoduje to w szczególności, że kolejne niższe jednostki podziału będą prawidłowo numerowane. W przypadku polecenia `\part` jego użycie lub nieużycie nie spowoduje żadnych nieoczekiwanych interferencji z efektami działania innych poleceń podziału. W szczególności numeracja *części* jest całkowicie niezależna od wszelkich innych numeracji.

3 Wybrane elementy składu

Oprócz narzędzi do wprowadzania struktury hierarchicznej w składanej publikacji, LaTeX ułatwia również przygotowywanie pewnych stałych lub też często spotykanych elementów takich, jak strona tytułowa, spis treści czy też indeks.

3.1 Strona tytułowa

Do zdefiniowania zawartości strony tytułowej służą w LaTeX-u polecenia:

`\title{`*tytuł*`}`
`\author{`*nazwiska i adresy autorów*`}`
`\date{`*data*`}`

Są one typowymi przykładami poleceń umieszczanych w *preambule* manuskryptu LaTeX-owego. Aby otrzymać stronę tytułową w składzie, należy dodatkowo po `\begin{document}` użyć polecenia `\maketitle`.

Przykład:

```
\documentstyle{iarticle}
...
\title{Kwantowa teoria bytu}
\author{Piotr Zieliński\\
        {\it Instytut Fizyki\\
             Uniwersytetu
             Mikołaja Kopernika}
        \and
        Adam Łęcki\\
        {\it Instytut Filozofii
             i Socjologii\\
             Uniwersytetu Poznańskiego}}
\date{1993}
...
\begin{document}
\maketitle
...
```

Kwantowa teoria bytu

Piotr Zieliński
Instytut Fizyki
Uniwersytetu Mikołaja Kopernika

Adam Łęcki
Instytut Filozofii i Socjologii
Uniwersytetu Poznańskiego

1993

Jak widać w przykładzie, jeśli publikacja ma kilku autorów, to informacje o nich w argumencie polecenia \author oddziela się poleceniem \and.

Reakcja TEX-a na polecenie \maketitle zależy od *stylu*, który używany jest do składania manuskryptu. Prawdziwa, oddzielna strona tytułowa tworzona jest tylko w przypadku stylów ireport i ibook. W przypadku stylu iarticle informacje zdefiniowane w preambule poleceniami \title, \author i \date umieszczane są w miejscu wystąpienia polecenia \maketitle. Na tej samej stronie umieścić można np. streszczenie lub spis treści. Aby otrzymać oddzielną stronę tytułową również w stylu iarticle, należy w poleceniu \documentstyle użyć opcji ititlepa:

```
\documentstyle[ititlepa]{iarticle}
```

3.2 Spis treści

Przy omawianiu poleceń podziału wspomnieliśmy, że jednym z „efektów ubocznych" ich użycia może być umieszczenie informacji na temat bieżącego elementu podziału w spisie treści.

W celu utworzenia spisu treści w części głównej manuskryptu należy umieścić polecenie \tableofcontents. Spowoduje ono, że TEX podczas przetwarzania manuskryptu wygeneruje dodatkowy plik, zawierający informacje o znalezionych w tekście użyciach poleceń podziału. W spisie treści umieszczone zostaną jednostki podziału jedynie do pewnej głębokości. Standardowo głębokość ta wynosi 3. Oznacza to, że w przypadku użycia stylu iarticle do spisu treści powędrują informacje związane z użyciem poleceń: \section, \subsection i \subsubsection, a w przypadku stylów ireport i ibook będą to — \chapter, \section oraz \subsection.

Aby spis treści pojawił się w składzie, niezbędne jest powtórne przetworzenie manuskryptu przez TEX-a. Podobnie rzecz się ma w przypadku wprowadzenia do tekstu manuskryptu większych zmian, które zaburzają dotychczasową strukturę podziału. Również wtedy dla uaktualnienia spisu treści konieczne jest dwukrotne przetworzenie manuskryptu.

3.3 Indeks

Przygotowywanie indeksu jest zawsze rzeczą skomplikowaną i bardzo pracochłonną. W pewnych fazach tego procesu LAMEX oferuje jednak dosyć daleko idącą pomoc. Rzeczą, której nie da się rozsądnie zautomatyzować jest oczywiście dobór haseł. Podobnie trudno spodziewać się, że da się programowo zdecydować, czy jakiś fragment tekstu związany jest w sposób istotny z danym hasłem. Wszystkie te decyzje musi podjąć autor manuskryptu. Od tego jednak momentu LAMEX zaczyna podawać swoją „pomocną dłoń". Prawdę mówiąc, to nie tyle sam LAMEX, co LAMEX we współpracy ze specjalnym programem o nazwie MakeIndex.

Do oznaczania miejsc w tekście, które naszym zdaniem związane są z hasłem, które chcielibyśmy umieścić w indeksie, używamy polecenia LAMEX-owego \index. W najprostszym przypadku argumentem polecenia \index jest hasło. W LAMEX-u istnieje jednak możliwość tworzenia indeksów wielostopniowych (maksymalnie 3-stopniowych). W takim przypadku argumentem polecenia \index jest ciąg haseł oddzielonych od siebie odpowiednim separatorem (czyli znakiem rozdzielającym).

Oprócz zaznaczenia w tekście manuskryptu miejsc „indeksowanych", za pomocą polecenia index, należy umieścić w nim kilka dodatkowych poleceń, które umożliwią wygenerowanie samego indeksu. Wśród opcji w poleceniu \documentstyle należy umieścić imakeidx, w *preambule* musi znaleźć się polecenie \makeindex, a w miejscu, gdzie chcielibyśmy, aby umieszczony został sam indeks — polecenie \printindex.

Po przetworzeniu za pomocą TEX-a tak przygotowanego manuskryptu powstanie plik o rozszerzeniu idx, zawierający dane dla programu MakeIndex.

Stosując MakeIndex do pliku idx otrzymujemy plik o rozszerzeniu ind, zawierający posortowany indeks w postaci zrozumiałej dla polecenia \printindex. Aby indeks znalazł się w składzie, musimy powtórnie przetworzyć manuskrypt za pomocą TEX-a.

Jeśli do tekstu manuskryptu wprowadzimy zmiany, to cykl generowania indeksu zmuszeni będziemy powtórzyć.

4 Odsyłacze

Jednym z najbardziej użytecznych mechanizmów, które udostępnia LAMEX jest mechanizm odsyłaczy. Kto kiedykolwiek napisał choćby kilkustronnicową broszurkę ten wie, jak bardzo potrzebna jest możliwość odwoływania się do informacji zawartych w pisanym tekście. Równie istotną rzeczą,

zwłaszcza w zastosowaniach akademickich, jest posiadanie wygodnej metody odwoływania się do pozycji bibliograficznych. W obu tych sprawach LaTeX okazuje się bardzo pomocny.

4.1 Odsyłacze do tekstu

Miejsce w tekście, do którego chcemy się następnie odoływać „oznakować" musimy najpierw stosując polecenie:

\label{*etykieta*},

gdzie *etykieta* może być dowolnym ciągiem liter, cyfr i znaków interpunkcyjnych.

Do każdego miejsca oznaczonego za pomocą polecenia \label można odwoływać się za pomocą polecenia:

\pageref{*etykieta*}

Polecenie to spowoduje, że w miejscu jego występowania, do składu wpisany zostanie numer strony, na której aktualnie znajduje się miejsce oznaczone etykietą.

Oprócz możliwości odwołania się do numeru strony, na której znajduje sie oznakowany tekst, istnieje również możliwość odwoływania się do numerowanych fragmentów tekstu takich, jak na przykład rozdziały, punkty, podpunkty itd. W tym celu należy użyć polecenia \label bezpośrednio po rozpoczęciu fragmentu tekstu, do którego chcemy się odwoływć. Następnie, w odpowiednim miejscu należy użyć polecenia:

\ref{*etykieta*}

Do składu wpisany zostanie wówczas odpowiedni numer — np. w przypadku rozdziału będzie to jego aktualny numer.

4.2 Odsyłacze bibliograficzne

Oprócz omówionego powyżej mechanizmu odsyłaczy do tekstu, w LaTeX-u istnieją również narzędzia służące do powoływania się na pozycje bibliograficzne. Żeby jednak było się do czego odwoływać trzeba najpierw stworzyć spis cytowanej literatury. Do tego celu służy w LaTeX-u środowisko o nazwie thebibliography.

```
\begin{thebibliography}{XXXXXXXX}
  \bibitem[Knuth 86]{TeXbook}
   Donald E.~Knuth, {\it The \TeX book\/},
   Volume A {\it`Computers and
   Typesetting'\/} Reading, Massachusetts,
   Addison Wesley Publishing Company,
   1986.
  \bibitem[Lamport 85]{LaTeX}
   Leslie Lamport, {\it\LaTeX: A~Document
   Preparation System\/} Reading,
   Massachusetts, Addison Wesley
   Publishing Company, 1985.
  \bibitem[Lamport 92]{LaTeX-pl}
   Leslie Lamport, {\it \LaTeX: System
   przygotowywania dokumentów\/}
   (tłum. Piotr Wyrostek), ARIEL 1992.
\end{thebibliography}
```

W powyższym spisie zdefiniowane zostały trzy pozycje. Każda z definicji ma postać:

\bibitem[*znacznik*]{*etykieta*}*dane_o_pozycji*

Omówimy teraz krótko użyte w powyższej konstrukcji elementy. „Znacznik" jest to tekst, który będzie się pojawiał w składzie każdorazowo przy powoływaniu się na daną pozycję bibliograficzną. Znacznik jest opcjonalny. Jeśli się go nie użyje, to pozycja otrzyma domyślne oznaczenie liczbowe — np. [1]. „Etykieta" służy do powoływania się na daną pozycję w tekście za pomocą polecenia \cite{*etykieta*}. „Dane_o_pozycji" to dowolne informacje na temat danej pozycji bibliograficznej. Zwykle podaje się autora, tytuł, nazwę wydawnictwa i rok wydania.

Tajemniczy ciąg iksów w drugim argumencie polecenia \begin, otwierającego spis literatury, służy do określania maksymalnej szerokości znacznika i wykorzystywany jest przez TeX-a w trakcie składania spisu cytowanej literatury.

5 LaTeX — a sprawa polska

Standardowo LaTeX posługuje się polskimi regułami składu tekstu. Algorytm dzielenia wyrazów zgodny jest z zasadami języka polskiego. Oczywiście dostępne są polskie znaki diakrytyczne (tzn. ą, ć, ę,...). Mogą być one wprowadzane do tekstu manuskryptu LaTeX-owego na dwa sposoby — „bezprefiksowo" albo przy użyciu tzw. notacji „prefiksowej". Pierwsza z tych metod polega na wprowadzaniu polskich liter bezpośrednio z klawiatury z użyciem jednego z kilku standardów takich jak „Mazovia" czy „Latin 2". Polskie litery mogą być wówczas widoczne na ekranie co w sposób istotny poprawia czytelność przygotowywanego manuskryptu. Możliwe jest wówczas również

definiowanie „polskich" poleceń takich jak \odstęp, \wcięcie itp.

Niezależnie od standardu kodowania polskich liter możliwe jest przygotowywanie tekstów dla LaMeX-a w notacji *prefiksowej*. W notacji tej polskie znaki diakrytyczne zapisywane są za pomocą dwóch znaków, z których pierwszy to standardowo znak „/". Aby uzyskać litery:

ą ć ę ł ń ó ś ź ż
Ą Ć Ę Ł Ń Ó Ś Ź Ż

należy napisać:

```
/a /c /e /l /n /o /s /x /z
/A /C /E /L /N /O /S /X /Z
```

O tym, która z notacji używana jest w sposób domyślny, należy zdecydować w momencie generowania „skompilowanej" wersji LaMeX-a. Niezależnie jednak od dokonanego wyboru notacje można dynamicznie zmieniać stosując polecenia \prefixing oraz \nonprefixing.

Jak już wspomnieliśmy, wraz z LaMeX-em dostarczane są polskie wersje standardowych stylów LaTeX-owych. Potrzeba ich stosowania do składu tekstów w języku polskim wynika z faktu, że w oryginalnych stylach LaTeX-owych zakodowane są informacje o angielskim nazewnictwie różnych elementów składu (takich jak rozdział, spis treści, spis literatury itp.). Powodowałoby to wstawianie słów angielskich takich jak „Chapter", „Contents", czy „Bibliography" do tekstu polskiego — co jest oczywiście niedopuszczalne. Należy podkreślić, że używanie polskich stylów nie powoduje zmian w składni LaTeX-a. I tak, rozpoczynając nowy rozdział przy użyciu polskiej wersji stylu book — czyli stylu ibook — nadal piszemy \chapter{...}, a nie np. \rozdział{...}. Zmieniane jest jedynie znaczenie polecenia \chapter.

Inne szczegóły związane z językiem polskim, a wspólne dla MeX-a i LaMeX-a, opisane zostały w artykule [Lichoński 93].

6 Parę uwag na koniec

Jak już wspomnieliśmy, początkowymi adresatami dzieła Lamporta byli głównie ludzie ze środowiska akademickiego. Nic więc dziwnego, że LaMeX najlepiej nadaje się do przygotowywania wszelkiego rodzaju prac naukowych. Nie oznacza to jednak, że nie da się go używać do innych celów. Przy jego pomocy mogą być składane manuskrypty o praktycznie nieograniczonej objętości — od krótkiego listu, czy notatki — do wielotomowej monografii.

Pewnym ograniczeniem LaMeX-a jest niestety fakt, że jego pierwowzór — czyli LaTeX nigdy nie został precyzyjnie opisany na poziomie umożliwiającym w miarę bezbolesne dostosowywanie go do niestandardowych zastosowań. Książka [Lamport 85] pod tym względem w najmniejszym nawet stopniu nie dorównuje dziełu [Knuth 86]. Także wszędzie tam, gdzie wymagane jest dokładne dopasowanie szaty typograficznej składanej publikacji do z góry zadanego wzorca, należy raczej skorzystać z innego formatu TeX-owego — np. MeX-a.

Jeśli jednak wymagania co do szaty typograficznej nie są tak sztywne, to ułatwiające pracę nad składem cechy LaMeX-a w pełni zrekompensują pewną „standardowość" wyglądu końcowego efektu naszej pracy.

Cytowana literatura

[Knuth 86] Donald E. Knuth, *The TeXbook*, Volume A *'Computers and Typesetting'* Reading, Massachusetts, Addison Wesley Publishing Company, 1986.

[Lamport 85] Leslie Lamport, *LaTeX: A Document Preparation System* Reading, Massachusetts, Addison Wesley Publishing Company, 1985.

[Lamport 92] Leslie Lamport, *LaTeX: System przygotowywania dokumentów* (tłum. Piotr Wyrostek), ARIEL 1992.

[Lichoński 93] Bogusław Lichoński, *Opis pakietu MeX*, Biuletyn Polskiej Grupy Użytkowników Systemu TeX, Zeszyt 1 (1993), str. 3–5.

[Pawłowski 93] Wiesław Pawłowski, *LaTeX czyli TeX dla ludzi*, Bachotek 1993.

PLAIN

Tam gdzie minus oznacza dzielenie

Bogusław Jackowski
i Marek Ryćko

Ten nieco przewrotny tytuł dotyczy oczywiście przenoszenia, czyli inaczej dzielenia wyrazów, zwyczajowo zaznaczanego w tekstach kreską przypominającą minus.

Automatyczne dzielenie wyrazów należy do podstawowych zadań wykonywanych przez programy komputerowe służące do składania tekstów.

Jednym z najciekawszych algorytmów dzielenia wyrazów, a przy tym dokładnie opisanych i udokumentowanych, posługuje się system TEX. Jego „ciekawość" polega na tym, że jest to algorytm uniwersalny, tzn. dający się przystosować do dzielenia wyrazów w różnych językach, a jednocześnie zdumiewająco prosty i efektywny w implementacji. Warto się temu algorytmowi przyjrzeć.

Reguły dzielenia wyrazów w języku polskim

Żeby zrozumieć, jak trudne zadanie postawił przed sobą Knuth, zajmijmy się przez chwilę regułami dzielenia wyrazów w języku polskim. M. Szymczak („Słownik ortograficzny języka polskiego", PWN, Warszawa, 1986) pisze:

> „(...) Dzielenie wyrazów przy przenoszeniu wyrazów z jednego wiersza do następnego opiera się w ortografii polskiej na dwóch kryteriach: fonetycznym i morfologicznym. Kryterium fonetyczne nakazuje przenoszenie części wyrazu z jednego wiersza do drugiego zgodnie z podzielnością na sylaby, np. *bu-rza*, *nie-wy-trzy-ma-nie*, *li-tość*, *po-na-pi-sy-wa-li-by*, *za-nie-po-ko-jo-ny*, *wy-so-ko*, *le-piej*, *te-go*, *prze-ciw*. Kryterium morfologiczne nakazuje przenoszenie części wyrazu z jednego wiersza do drugiego zgodnie z podzielnością na przedrostek i rdzeń, a w wyrazach złożonych — w miejscu złożenia, np. *przed-murze*, *pod-punkt*, *wy-brzeże*, *roz-łąka*, *od-dźwięk*, *przy-kład*, *za-płata*, *na-trafić*, *wy-ścigi*, *o-brona*, *bez-wstyd*, *u-kładowy*, *po-dróż*, *pod-oficer*, *pod-oddział*, *na-przód*, *pod-opieczny*, *po-dmuch*, *noc-leg*, *konio-krad*. (...)"

przy czym

> „(...) kryterium morfologiczne jest nadrzędne w stosunku do kryterium fonetycznego (...)"

Zasady ogólne wyglądają prosto i przejrzyście. Omówienie zasad szczegółowych zajmuje Szymczakowi pięć stron i o ile człowiekowi mogłoby to wystarczyć, to jednak zalecenia takie jak:

> „(...) Jeżeli przejrzystość podziału na przedrostek i rdzeń jest w ***dzisiejszej świadomości językowej zatarta*** (podkr. aut.), grupy spółgłoskowe znajdujące się na granicy cząstek dzielimy dowolnie, np. *o-błok* a. *ob-łok*, *o-tręby* a. *ot-ręby*, (...)"

bardzo trudno zalgorytmizować.

Spróbujmy się zastanowić, czy nie ma takich zasad, od których nie byłoby wyjątków.

Dobrym kandydatem wydawać by się mogła reguła niedzielenia głosek *cz*, *dz*, *rz* oraz *sz*. Niestety, są wyjątki od tej reguły: *tysiąc-złotowy*, *pod-zwrotnikowy*, *mar-znąć* i *em-es-zet*. Podobnie miękkie *ni* może ulec rozbiciu, na przykład w słowie *in-iekcja*.

To może chociaż da się ustalić, że sylaba nie może składać się z samych spółgłosek? Jest to w zasadzie dobra reguła, ale ma (co najmniej) jeden wyjątek: szkocki przedrostek *Mc* wolno oddzielić. Na przykład nazwisko *McMillan* wolno podzielić *Mc-Millan*. Oczywiście implikuje to kłopoty z rzymskim zapisem liczb — rok *MCMXCII* komputer mógłby uznać za szkockie nazwisko.

W takim razie może istnieją nie zakazy, a nakazy dzielenia, od których nie ma wyjątków? Np. nakaz rozdzielania par identycznych spółgłosek (*świec-cy*, *Jagieł-ło*, *łam-my*, *pan-na*, *ar-ras*, *las-so*, *get-to* itp.). Odpowiedź, jak łatwo było przewidzieć, brzmi: „nie". Np. kryterium morfologiczne nakazuje podział *ode-ssie*, a nie *odes-sie*, chyba że chodzi o *Odessę*. A jeśli się zgodzić, że przedrostek rodzimy musi być oddzielony, to będzie kłopot ze słowem *zeppelin*, bo jak komputer miałby się domyślić, że w tym przypadku nie chodzi o przedrostek *ze*? Mógłby wprawdzie wiedzieć, że żadne słowo w języku polskim nie zaczyna się od *pp* (chociaż jest słowo zaczynające się od trzech spółgłosek, z czego dwie pierwsze to *ww* — proszę zgadnąć jakie to słowo). Ale jeśli jakiś autor chciałby oddać mowę jąkającego się bohatera pisząc np. *zeppsuuty*, to co wtedy?

Niejednoznaczności typu „*odessie*" stanowią dla komputera trudność praktycznie nie do pokonania.

Choć jest ich niewiele, to jednak zdarzają się i utrudniają życie, np.: *podróżować* — być w podróży lub pokryć różem lica; *odziewać* — ubierać lub odpowiedzieć w taki sam sposób komuś, kto ziewnął; *podrobić* — np. karmę ptactwu lub podpis; *Tarzanie* — słowo to może oznaczać wzywanie znanego bohatera komiksów lub czynność tarzania się; *narwali* — to słowo z kolei można rozumieć jako czas przeszły dokonany albo dopełniacz liczby mnogiej.

W sytuacjach tego typu jedynie kontekst semantyczny decyduje o podziale słowa. Rozsądnie jest nie żądać zbyt wiele od programów składających teksty i zostawić człowiekowi możliwość ingerencji w bardziej skomplikowanych przypadkach.

Sporo trudności nastręcza dzielenie nazwisk i nazw własnych. O nazwiskach szkockich już była mowa. Ale i polskie nazwy potrafią sprawić kłopot. Na przykład czy w nazwisku *Utnik* głoskę *u* należy traktować jako przedrostek, tak jak w słowie *u-tnie*? Szczególnie trudne do podziału są słowa będące zbitką dwóch lub więcej słów. Zalecany jest oczywiście podział w miejscu złożenia: *noc-leg*, *trzech-set-letni* itp. Zdarzają się jednak czasem bardzo osobliwe nazwy własne, np. *Wierzchlas*. Chociaż taki właśnie podział wydaje się uzasadniony, to przecież trudno dać głowę, że nazwy tej nie da się wywieść od *wierzenia* i *chlastania*. Podobnie gdyby jakaś miejscowość nazywała się *Szynkwas*, to nazwa mogłaby się wywodzić ostatecznie — choć jest to mało prawdopodobne — od *szyn* i *kwasu*.

Zeppelin i *szynkwas* to przykłady słów obcych bądź obcego pochodzenia, które zadomowiły się w naszym języku. Słowa tej kategorii, takie jak np. *er-zac*, *jazz-man* czy *soft-ware*, to źródło nowych utrapień dla kogoś, kto chciałby sformalizować reguły dzielenia wyrazów w języku polskim.

W tej sytuacji nie dziwią wyniki testów porównawczych W. Puzy (W. Puza, praca magisterska pt. „Analiza porównawcza wybranych programów Desktop Publishing (DTP)", Instytut Poligrafii Politechniki Warszawskiej), który stwierdził, co następuje:

> „(...) Test sprawdzający poprawność dzielenia polskich słów wyodrębnił 3 różne poziomy skuteczności testowanych programów:
>
> - Cyfroset, TEX – ok. 10 % błędów
> - Dywiz (dedykowany dla Quark'a) – powyżej 20 % błędów
> - Bizon (dedykowany Calamusowi), VP Commander (dedykowany Venturze) – powyżej 50 % błędów. (...)"

Tak więc pozornie trywialny problem okazuje się w praktyce trudnym orzechem do zgryzienia.

Można oczywiście polemizować z wytycznymi językoznawców. Zalecany przez Szymczaka podział *za-jrzeć* lub *po-jmać* wydaje się dziś anachroniczny. Pójście dalej tym tropem, przy równoczesnym zasłanianiu się „wymaganiami komputerów", powoduje, że zbyt kuszące staje się maksymalne upraszczanie reguł, oczywiście z uszczerbkiem dla języka polskiego.

Autor TEX-a obrał zupełnie inną drogę: szukał takiego algorytmu, który byłby w zasadzie niezależny od reguł dzielenia wyrazów; reguły dzielenia miałyby być określone za pomocą danych do tego algorytmu. Algorytm spełniający te wymagania skonstruował F. M. Liang (rozprawa doktorska pt. „Word Hy-phen-a-tion by Com-put-er", Uniwersytet Stanforda, USA, 1983). Udało mu się nawet coś więcej: dane są stosunkowo łatwe do utworzenia, nie jest do tego bynajmniej potrzebna wiedza informatyczna.

Jak TEX dzieli wyrazy

Algorytm dzielenia wyrazów jest niezwykle prosty. Oczywiście przy założeniu, że odpowiednie dane zostały utworzone. Załóżmy więc, że te dane mamy. Składa się na nie zbiór ciągów utworzonych na przemian z liter i cyfr, np.: `2s0z1z0`, `2s0z0l0n0`, `2t1ć0` itp. Zamiast cyfry skrajnej może wystąpić kropka, np.: `.w0e3s2`, `.w0w8`, `8r0s0z.`, `8r0z0ł.` itp. Ciągi te nazywać będziemy wzorcami.

Przypuśćmy teraz, że mamy dane jakieś słowo, np. `odkaszlnąć`. Najpierw zamieniamy je na postać taką, jaką mają wzorce, wstawiając pomiędzy literami cyfrę 0, a na brzegach kropki, otrzymując `.o0d0k0a0s0z0l0n0ą0ć.` (kropka — jak stąd widać — oznacza skraj słowa). Następnie wyszukujemy w naszym zbiorze danych wszystkie wzorce, które na to słowo dają się nałożyć w taki sposób, że zgadzają się położenia liter i kropek, ale niekoniecznie cyfr; w szczególności cyfra może się nakładać na kropkę, ale nie na literę. Powiedzmy, że znaleźliśmy następujące wzorce: `.o2d2`, `.o0d3k2`, `0ą1`, `2d1k0`, `2l1n0`, `2s0z1l0`, `2s0z0l0n0`,

8ć., 0a1, 0o1, 0s4z0. Jeśli je nałożymy na słowo .o0d0k0a0s0z0l0n0ą0ć. i spośród cyfr nakładających się na siebie weźmiemy największe, to otrzymamy .o2d3k2a2s4z2l1n0ą8ć. — i to już wszystko. Zgodnie z algorytmem Lianga słowo wolno podzielić jedynie w takim miejscu, w którym pojawia się cyfra nieparzysta. Oznacza to, że algorytm dopuszcza w tym wypadku dwa punkty podziału: *od-kaszl-nąć*.

Wzorce przechowywane są w pamięci komputera jako struktura drzewiasta (dokładniej *trie*; p. D. E. Knuth, „The Art of Computer Programming", tom 3, „Sorting and Searching"). Pozwala to na znaczne upakowanie danych przy zachowaniu szybkiego dostępu do informacji, a równocześnie struktura danych i algorytm są stosunkowo łatwo implementowalne.

Warto podkreślić, że dzięki efektywności zastosowanej metody TEX może przechowywać w pamięci kilka różnych zestawów wzorców i dzielić wyrazy w kilku językach w obrębie jednego dokumentu (a nawet akapitu).

Problem jedynie w tym skąd wziąć wzorce?

Automatyczne tworzenie danych dla algorytmu dzielenia wyrazów

Główna część pracy Lianga dotyczyła metody tworzenia wzorców na podstawie słownika zawierającego słowa poprawnie podzielone. Przygotowanie takiego słownika jest oczywiście czaso- i pracochłonne, ale za to dalsza część zadania jest już względnie łatwa. W największym uproszczeniu algorytm Lianga można zapisać następująco:

```
m := maksymalna długość wzorca
for c := 1 to 9 do
begin
  ustal kryteria akceptowalności dla wzorców
  for l := 1 to m do
  begin
    dla każdego słowa w słowniku sprawdź,
    czy nie dostarcza ono informacji o możliwości wstawienia cyfry c we wzorcach długości l przy zadanych kryteriach akceptowalności
  end
end
```

Innymi słowy program uzupełnia zestaw wzorców kolejno dla coraz większych cyfr i coraz dłuższych wzorców, zostawiając użytkownikowi na każdym etapie możliwość ingerencji. Kryteria akceptowalności określa się przez podanie trzech liczb: wagi dla podziałów poprawnych w_p, wagi dla podziałów niepoprawnych w_n i poziomu akceptowalności p. Wzorzec jest akceptowany, gdy $k_p w_p - k_n w_n \geq p$, gdzie k_p i k_n oznaczają odpowiednio liczbę poprawnych i niepoprawnych podziałów generowanych przez dany wzorzec.

W niektórych przypadkach może się zdarzyć, że program nie znajdzie zestawu wzorców dzielącego poprawnie wszystkie słowa. Powodem mogą być błędne dane (np. w słowniku może się znaleźć dwa razy to samo słowo, ale inaczej podzielone) bądź niewłaściwy dobór liczb w_p, w_n oraz p w kolejnych iteracjach. Różne strategie dobierania wartości tych współczynników wpływają na objętość wynikowego zestawu wzorców. Minimalizacja liczby wzorców w wygenerowanym automatycznie zestawie wymaga pewnej wprawy i — oczywiście — przestudiowania pracy doktorskiej Lianga, gdzie zagadnienie to jest szczegółowo omówione.

Jak widać „nauczenie" TEX-a zasad dzielenia wyrazów w danym języku jest właściwie kwestią odpowiednio zasobnego słownika podzielonych poprawnie wyrazów. Dostarczenie takiego słownika to zadanie dla językoznawców. Natomiast dalszy etap jest ideowo prosty, nie wymaga znajomości technik programowania, ani żadnych innych specjalnych umiejętności. Potrzebna jest jedynie cierpliwość.

Ręczne tworzenie danych dla algorytmu dzielenia wyrazów

Możliwość automatycznego tworzenia zestawu wzorców nie wyklucza możliwości stworzenia takiego zestawu ręcznie. Początkowo wydawało się, że język polski ma na tyle regularne zasady dzielenia wyrazów, że łatwiej będzie utworzyć zestaw wzorców w ten właśnie sposób.

W 1984 J. Désarménienowi udało się zamknąć reguły dzielenia wyrazów dla języka francuskiego w zestawie liczącym 804 wzorce i dla języka włoskiego w zestawie liczącym zaledwie 88 wzorców, podczas gdy standardowy zestaw wzorców dla amerykańskiego TEX-a zawiera ich 4447 (J. Désarménien, „The use of TEX in French: hyphenation and typography", w: D. Lucarella (ed.), „TEX for Scientific Documentation, Proc. of the First European

Conference, Addison-Wesley, 1984). Sukces podejścia Désarméniena zainspirował H. Kołodziejską, która w roku 1987 opublikowała listę 2168 wzorców dla języka polskiego (H. Kołodziejska, „Dzielenie wyrazów w systemie TEX", Sprawozdania Instytutu Informatyki Uniwersytetu Warszawskiego, nr 165, 1987).

Wynik Kołodziejskiej zdawał się potwierdzać wstępne założenia. Niestety. Po kilku latach intensywnego testowania (maczali w tym palce m. in. autorzy niniejszej pracy) wzorców znacznie przybyło. Po gruntownej przeróbce liczebność zestawu wzorców wzrosła do 4053 i wolno przypuszczać, że to jeszcze nie koniec.

W tych warunkach zasadne staje się pytanie, czy nie należałoby zrewidować założenia o regularności zasad podziału słów w języku polskim. Jest prawdopodobne, że program Lianga mógłby wygenerować mniej liczny zestaw wzorców.

Tym niemniej, jak wynikało z porównań poczynionych przez Puzę (p. wyżej), zestaw wzorców dla języka polskiego w swojej obecnej postaci daje wyniki zadowalające. Zważywszy, że Puza w istocie dysponował jedną z wersji pośrednich zestawu i że zestaw aktualny jest znacznie udoskonalony w stosunku do tamtej wersji, można oczekiwać jeszcze lepszych rezultatów.

Najaktualniejsza wersja tego zestawu wchodzi w skład pakietu MEX (polskiej adaptacji TEX-a). Pakiet ten jest dystrubuowany w postaci źródłowej jako produkt *public domain* przez GUST.

Inne aspekty automatycznego dzielenia wyrazów

Automatyczne znajdowanie miejsc podziału słów nie wyczerpuje spraw związanych z dzieleniem wyrazów do celów składu komputerowego. Jak zostało już wspomniane, czasami ingerencja człowieka staje się niezbędna. Poprawnie zaprojektowany system składu taką ingerencję powinien umożliwiać. Użytkownik powinien mieć w szczególności możliwość wprowadzenia zakazu dzielenia niektórych słów.

W przypadku TEX-a nierozsądne byłoby manipulowanie za każdym razem przy zestawie wzorców. Użytkownik TEX-a może podzielić słowo według własnego uznania na dwa sposoby.

Pierwszy sposób to umieszczenie go na liście wyjątków, nadrzędnej w stosunku do wzorców. Jest to, niestety, metoda wygodna jedynie w językach niefleksyjnych. W języku polskim umieszczenie jakiegoś słowa na liście wyjątków wiąże się (w przypadku TEX-a) z podaniem od razu wszystkich jego odmian, co jest raczej nieporęczne. Na szczęście problem taki nie pojawia się zbyt często. Głównym powodem tego typu zabiegów bywają słowa złożone lub obcego pochodzenia (bądź i jedno, i drugie).

Drugi sposób polega na wskazaniu w tekście dodatkowych miejsc podziału (ang. *discretionary hyphens*). Większość systemów DTP oferuje taką możliwość użytkownikom. W przypadku TEX-a należy ostrożnie posługiwać się tą metodą, bo można przeszkodzić TEX-owi w automatycznym wstawianiu drobnych odstępów między znakami (inaczej podcięć, ang. *implicit kerns*).

TEX oferuje użytkownikowi jeszcze parę innych możliwości, pozwalających na panowanie nad algorytmem podziału wyrazów.

Zabronienie podziału wyrazu jest w TEX-u sprawą trywialną, wystarczy wyraz „zamknąć" w tzw. pudełko (TEX-owa komenda \hbox).

Użytkownik może też określić minimalną długość oddzielonego początku i końca słowa. Domyślnie TEX nie podzieli słowa krótszego niż pięcioliterowe: początek nie może być krótszy niż dwie litery, koniec — nie krótszy niż trzy litery. Przy składaniu w wąskiej szpalcie może się pojawić konieczność złagodzenia tych rygorów, zwłaszcza że w języku polskim za dopuszczalne uznaje się oddzielanie pojedynczej samogłoski na początku słowa i dwuliterowej sylaby na końcu. Z drugiej strony w składzie wysokiej jakości powinno się unikać dzielenia wyrazów w ogóle, a tym bardziej na zbyt krótkie fragmenty. Możliwość sterowania wielkością dzielonego słowa jest więc bardzo przydatna w praktyce.

Zasady dobrego składu wymagają, aby wyrazy dzielone nie pojawiały się w sąsiednich wierszach i by ostatni wiersz na prawej stronie nie zawierał słowa dzielonego. Składy nie spełniające tych wymogów uważa się (i słusznie) za brzydkie. TEX dostarcza parametrów pozwalających skutecznie utrudniać brzydkie składanie. W szczególności można wręcz zabronić umieszczania słowa dzielonego w ostatnim wierszu na stronie. Może to sprawić pewne kłopoty algorytmowi łamiącemu wiersze na strony, ale to już zupełnie inna historia.

Zostaje jeszcze problem słów zawierających łącznik, np. *biało-czerwony*. W języku angielskim słowa takie jak *machine-oriented* przenosi się

machine-
oriented

podczas gdy w języku polskim stanowczo zaleca się dostawienie dodatkowo łącznika na początku następnego wiersza:

biało-
-czerwony

TEX pozwala dołączyć do zestawu komend standardowych komendy definiowane przez użytkownika. W MEX-u została zdefiniowana komenda `\=`, którą należy podczas przygotowywania tekstu umieścić w miejscu łącznika. Kontynuując przykład, *biało-czerwony* należałoby podać TEX-owi jako `biało\=czerwony`. Użytkownik musi pamiętać o konsekwentnym przestrzeganiu tej zasady, a o resztę zadba już TEX. Pozostawienie zwykłego łącznika oznacza słowo niepodzielne. Jest to przydatne w przypadku takich słów jak np. *K-202*, których oczywiście dzielić nie należy.

Uwagi końcowe

Jak widać zagadnienie automatycznego dzielenia wyrazów to problem bardzo obszerny. Wchodzą tu w grę zagadnienia językoznawcze, typograficzne i informatyczne.

Praktyka pokazuje, że bardzo efektywny algorytm automatycznego podziału słów, w połączeniu z niekoniecznie bardzo wygodnymi możliwościami ingerowania ręcznego, stwarza użytkownikowi możliwość panowania w pełni nad procesem dzielenia słów przez system komputerowego składu tekstów.

Oprogramowanie

Z notatnika oblatywacza TEX-owego

Stanisław Wawrykiewicz

Witam w nowym dziale GUST-ownego biuletynu, poświęconym nowinkom dotyczącym szeroko pojętego oprogramowania TEX-owego oraz praktyce jego konfigurowania i obsługi. Nie roszczę sobie wyłączności do pisania w tym dziale i wobec tego dołączam się do apelu PT Redaktorów o przesyłanie artykułów i notatek, poświęconych szczególnie programom działającym na innych niż IBM PC/MS DOS platformach sprzętowo-systemowych.

Na początek oprogramowanie podstawowe, czyli sam program TEX. Większość z nas używa (na komputerach PC) bardzo popularnego emTEX-a Eberharda Mattesa. Zaprezentuję tutaj krótko SB38TEX, będący implementacją TEX-a w wersji 3.141, autorstwa Wayne G. Sullivana i Petera Breitenlohnera. Program charakteryzuje się zwartym kodem, uruchamiany może być nawet na komputerach klasy XT, jest bardzo szybki i sprawniej niż standardowy emTEX radzi sobie z większą ilością fontów na stronie składu.

SB38TEX dostępny jest jako dobro wspólne (*public domain*) w pakiecie `sb38tex.zip` spakowanym programem `pkzip`. Zawiera on 31 plików zajmujących po dekompresji ok. 0.5MB, w tym: `tex.exe` — właściwy program datowany na 18.05.92 (143552 bajtów), `initex.exe` — program do generowania *formatu*, czyli pliku binarnego zawierającego definicje makr, wzorce przenoszenia itp., `plain.tex` — plik źródłowy podstawowego formatu, `tex.poo` i `nohelp.poo` — pliki unikalne dla każdej implementacji TEX-a, zawierające komunikaty generowane przez program, niezbędne przy tworzeniu formatu (`tex.poo` jest pełnym zestawem komunikatów i podpowiedzi TEX-a), `sb38tex.doc` — dokumentacja pakietu przygotowana do złożenia przy użyciu standardowego formatu `plain.fmt`, 16 plików metrycznych (z rozszerzeniem `.tfm`) podstawowych fontów TEX-a. SB38TEX pozwala na zdefiniowanie zamiany kodów znaków czytanych przez TEX-a na kody wewnętrzne, zgodnie z którymi znaki umieszczone są w plikach fontów. Definicje takie powinny być zawarte w pliku `codep.age` obecnym w bieżący katalogu podczas generowania formatu. Przykład takiego pliku znajduje się w pakiecie, zaś poprawność notacji można sprawdzić dołączonym programem `cdpgtest.exe`. Z kolei program `sb38set.exe` służy do modyfikacji pakietu w przypadku gdy będzie on instalowany na dysku innym niż `C:`. Wymagana jest wtedy obecność w bieżącym katalogu plików: `tex.exe`, `initex.exe`, `tex.poo` i `nohelp.poo`.

Dodatkowo dołączono do pakietu źródła Pascalowe `sbmenu.pas` prostego programu typu *Shell* uruchamiającego TEX-a, edytor i sterowniki (*drivers*) ekranu i drukowania. Program wymaga modyfikacji zgodnie z lokalną konfiguracją TEX-a i używanymi programami sterowników, po czym kompilacji Turbo Pascalem.

Przygotowanie formatu Plain

Przygotowanie formatu `plain.fmt` wymaga następujących plików z pakietu sb38tex: `initex.exe`, `tex.poo`, `plain.tex`, `hyphen.tex` i ewentualnie zmodyfikowanego pliku `sbcodep.age` Należy określić zmienne środowiskowe DOS-a używane przez program `tex.exe`, proponuję tutaj utworzenie katalogów o krótkich nazwach:

```
SET FMTSB=c:\tex\sbtex
SET SBFTMP=C:\;-r04
SET FONTTFMS=.;c:\tex\tfms
SET TEXINPUTS=c:\tex\lib;c:\tex\input
```

Zmienne oznaczają kolejno (w nawiasach podano wartości domyślne): `FMTSB` (`c:\tex\formats`) — katalog zawierający format(y), czyli plik(i) `*.FMT`, `SBFTMP` (`c:\;-r06`) — katalog na tworzone podczas pracy TEX-a pliki tymczasowe i sposób gospodarki pamięcią (tu maksymalne wykorzystanie pamięci na pamięć podstawową programu, parametry opisane są bardziej szczegółowo w dokumentacji), `FONTTFMS` (`c:\tex\fonttfms`) — katalog(i) z plikami metrycznymi `.TFM`, wreszcie `TEXINPUTS` (`c:\tex\inputs`) — katalog(i) przeszukiwane przez TEX-a po użyciu komendy `\input`. Katalogi standardowo oddzielamy średnikiem.

Wymienione wyżej programy najlepiej umieścić w katalogu wskazywanym przez zmienną `FMTSB`. Generowanie formatu wymaga uruchomienia polecenia:

```
initex plain \dump
```

TEX wraz z utworzonym właśnie formatem najlepiej uruchamiać za pośrednictwem pliku wsadowego, np. `t.bat`:

```
c:\tex\sbtex\tex &plain %1 %2
```

Przygotowanie formatu MEX

Korzystamy z tych samych plików i katalogu co wyżej. Dodatkowo z pakietu mex103 niezbędne są: `mex.tex`, `mex1.tex`, `mex2.tex`, `mexconf.tex`, `plhyph.tex`, oraz wygenerowane METAFONT-em pliki `*.TFM` 16 podstawowych fontów MEX-a predefiniowanych w formacie: `plr5`, `plr7`, `plr10`, `plmi5`, `plmi7`, `plmi10`, `plsy5`, `plsy7`, `plsy10`, `plex10`, `plbx5`, `plbx7`, `plbx10`, `pltt10`, `plsl10`, `plti10`. Jak widać są to odpowiedniki fontów predefiniowanych formatu `plain`. Modyfikujemy plik `sbcodep.age` zgodnie z wybranym „standardem" kodowania polskich liter w pliku do składu (zawartość pliku dla kodów MAZOVIA i LATIN2 przedstawiono poniżej, należy zachować dokładną notację, włącznie z końcowymi znakami <<).

Ponieważ format MEX zawiera zwykle wzorce przenoszenia dla dwóch języków musimy na czas jego generowania zmodyfikować gospodarkę pamięcią, zwiększając pamięć wzorców (*trie size*). Wykonajmy polecenia:

```
SET SBFTMP=C:\;-r04;-t13
initex mex \dump
```

Uruchamiamy TEX-a za pomocą pliku, np. `mex.bat` (zakładam dostęp do samego programu wyspecyfikowany zmienną `PATH`):

```
tex &mex %1 %2
```

Poniższy skrypt (np. `plain.bat`) umożliwia emulację formatu `plain`, w formacie MEX:

```
tex &mex \emulateplain\input %1 %2
```

Po wygenerowaniu formatów zostawiamy na dysku jedynie `tex.exe`, plik(i) `*.fmt` i oczywiście pliki `*.TFM`. Ludzie, nie zaśmiecajcie sobie dysków!

Po lewej zawartość pliku `sbcodep.age` dla przekodowania znaków wejściowych w kodach MAZOVIA, po prawej — w kodach LATIN2 (oba „widziane" we własnych środowiskach edycyjnych)

```
Ą=^^81          Ą=^^81
Ć=^^82          Ć=^^82
Ę=^^86          Ę=^^86
Ł=^^8a          Ł=^^8a
Ń=^^8b          Ń=^^8b
Ó=^^d3          Ó=^^d3
Ś=^^91          Ś=^^91
Ź=^^99          Ź=^^99
Ż=^^9b          Ż=^^9b
ą=^^a1          ą=^^a1
ć=^^a2          ć=^^a2
ę=^^a6          ę=^^a6
ł=^^aa          ł=^^aa
ń=^^ab          ń=^^ab
ó=^^f3          ó=^^f3
ś=^^b1          ś=^^b1
ź=^^b9          ź=^^b9
ż=^^bb          ż=^^bb
^^81=^^8f       ^^81=^^a4
^^82=^^95       ^^82=^^8f
^^9b=^^90       ^^f3=^^a8
^^8a=^^9c       ^^8a=^^9d
^^8b=^^a5       ^^8b=^^e3
^^d3=^^a3       ^^d3=^^e0
^^b9=^^98       ^^91=^^97
^^99=^^a0       ^^99=^^8d
^^f3=^^8d       ^^9b=^^bd
^^aa=^^92       ^^a1=^^a5
^^ab=^^a4       ^^a6=^^a9
^^b1=^^9e       ^^aa=^^88
^^bb=^^a7       ^^b9=^^e4
<<              ^^b1=^^98
                ^^bb=^^be
                <<
```

Statut Polskiej Grupy Użytkowników Systemu TEX „GUST”

(*fragmenty*)

Rozdział. I Przepisy ogólne

art. 1 Polska Grupa Użytkowników Systemu TEX „GUST”, zwana dalej „Grupą”, jest stowarzyszeniem zrzeszającym osoby fizyczne, prowadzącym w różnych formach działalność na rzecz upowszechnienia systemu TEX i systemu generowania fontów METAFONT, oraz związanego z nimi środowiska.

art. 2 Grupa działa na podstawie ustawy z dnia 7 kwietnia 1989 r. „Prawo o stowarzyszeniach” (Dz. U. z 1989 r. nr 20 poz. 104 z późn. zm.). Grupa posiada osobowość prawną.

art. 3 Grupa działa na terenie Rzeczypospolitej Polskiej i za granicą zgodnie z przepisami prawa miejscowego. Siedzibą władz Grupy jest m. st. Warszawa.

art. 4 Grupa może być członkiem organizacji krajowych i zagranicznych.

art. 5 Grupa może używać własnych odznak i pieczęci zgodnie z obowiązującymi w tym zakresie przepisami.

art. 6 Grupa opiera swoją działalność na społecznej pracy członków.

Rozdział. II Cele i formy działania

art. 7 Celem Grupy jest:

1) zrzeszanie użytkowników systemu TEX-a;
2) upowszechnianie systemu TEX, systemu generowania fontów METAFONT, ich środowiska, jak również związanego z nimi oprogramowania stanowiącego tzw. dobro wspólne (*public domain*);
3) propagowanie ochrony praw autorskich;
4) reprezentowanie członków Grupy, ich opinii i potrzeb;
5) ułatwianie kontaktów między członkami Grupy.

art. 8 Grupa osiąga swoje cele przez:

1) współdziałanie z instytucjami, towarzystwami naukowymi oraz stowarzyszeniami, tak krajowymi jak i zagranicznymi;
2) inspirowanie, wspieranie i patronowanie działalności mającej na celu upowszechnianie systemów TEX i METAFONT oraz rozwój ich środowiska;
3) prowadzenie działalności szkoleniowej poprzez organizowanie kursów, konferencji, odczytów, wystaw i pokazów w dziedzinach objętych działalnością Grupy;
4) prowadzenie działalności wydawniczej;
5) ułatwianie wymiany informacji poprzez tworzenie archiwów i prowadzenie dystrybucji oprogramowania *public domain* związanego z systemami TEX i METAFONT.
6) rozwijanie innych form działalności merytorycznej służącej realizacji celów statutowych.

Rozdział. III Członkowie, ich prawa i obowiązki

art. 9 Członkowie Grupy dzielą się na:

1) zwyczajnych,
2) wspierających,
3) honorowych.

art. 10 Członkiem zwyczajnym Grupy może być osoba fizyczna, posiadająca pełną zdolność do czynności prawnych, która poprzez złożenie deklaracji zobowiązuje się do działania na rzecz celów Grupy i przestrzegania postanowień jej Statutu.

art. 11 Członkiem wspierającym może być osoba prawna lub osoba fizyczna posiadająca pełną zdolność do czynności prawnych, która zadeklaruje na cele Grupy pomoc finansową lub rzeczową. Osoba prawna działa w Grupie przez swojego przedstawiciela.

art. 12 Członkiem honorowym może być osoba fizyczna, która wniosła wybitny wkład w rozwój systemu TEX lub w inny, szczególny sposób zasłużyła się Grupie.

art. 13

§1 Przyjęcie na członka zwyczajnego lub wspierającego następuje w drodze uchwały Zarządu.

§2 Godność członka honorowego nadaje Walne Zebranie Członków.

art. 17

§1 Członkowie zwyczajni posiadają czynne i bierne prawo wyborcze a nadto mają prawo do:

1) wyrażania swoich opinii i propozycji dotyczących działalności Grupy,
2) korzystania z urządzeń, świadczeń i pomocy Grupy.

§2 Członkowie wspierający i honorowi mają prawo członków zwyczajnych z wyjątkiem praw wyborczych oraz prawa głosu stanowiącego w obradach władz Grupy. Członkowie honorowi mają nadto prawo brania udziału z głosem doradczym w posiedzeniach wszystkich organów Grupy.

art. 18

§1 Obowiązkiem członków Grupy jest postępowanie zgodne ze Statutem, regulaminami i uchwałami władz Grupy.

§2 Członkowie zwyczajni zobowiązani są nadto do czynnego udziału w pracach Grupy oraz opłacania składek członkowskich.