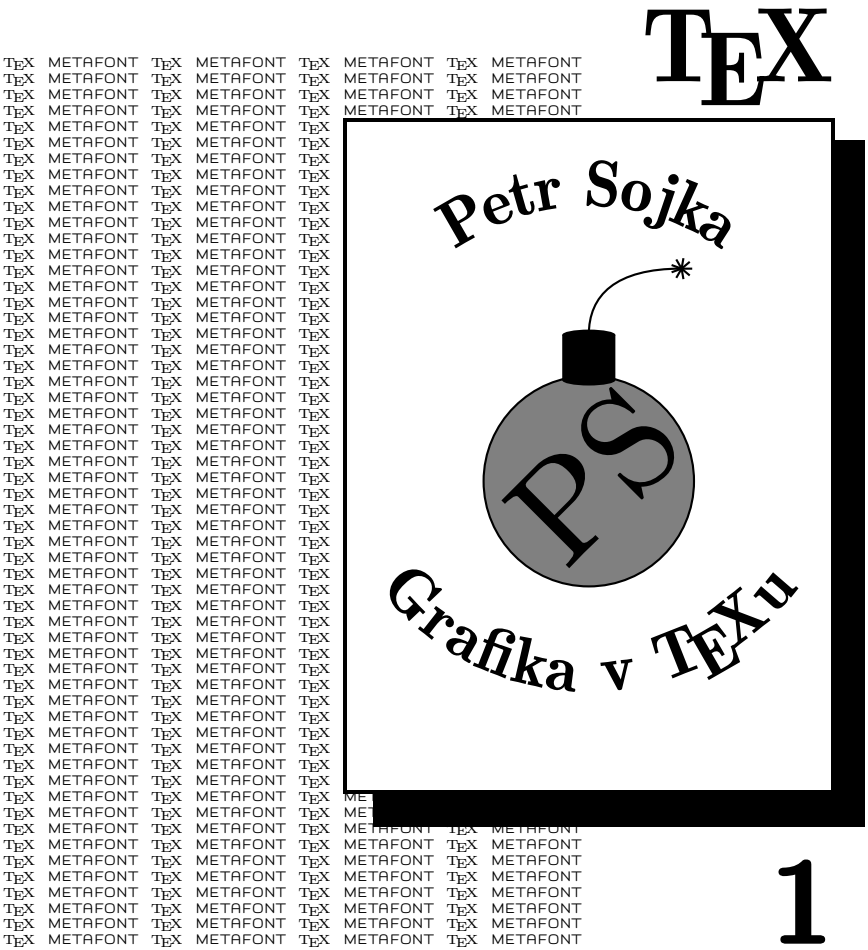


ZPRAVODAJ 94 1

ČESKOSLOVENSKÉHO SDRUŽENÍ UŽIVATELŮ TEXU

E-TEX: Sprievodca budúcimi rozšíreniami TEXu

Frank Mittelbach

Abstrakt

Ohlásením verzie TEX 3.0 opätoval Don Knuth (stále rastúcu) potrebu TEXovskej komunity ešte lepšieho systému. Ale súčasne dal jasne najavo, že nechce byť zaťahovaný do ďalšieho takého rozšírenia, ktoré by menilo *The TEXbook*.

TEX vznikol spočiatku ako systém určený na sádzanie vlastných publikácií autora. Medzičasom slúži stotisícom užívateľov. Nastal čas pohliadnuť sa po desaťročných skúsenostiach naspäť, či je, alebo nie je TEX 3.0 adekvátnou odpoveďou na sadzačské požiadavky deväťdesiatych rokov.

Výstup vytvorený TEXom má vyšší štandard než výstup automaticky generovaný väčšinou iných sádzacích systémov. Z toho dôvodu sa v tomto článku sústredíme na kvalitatívne štandardy predložené typografmi pre ručne sádzané dokumenty a spýtame sa, do akej miery sú tieto štandardy dosiahnuté TEXom. Budeme analyzovať možnosti TEXovských algoritmov a naznačíme nedostatky ako aj nové koncepcie.

1. Úvod

Minulý rok sme v Stanforde oslávili desiate narodeniny projektu TEX. TEX slúžil dodnes tisícom užívateľov a očakávame, že tak bude činiť aj v budúcnosti. Dlhovekosť TEXu spočíva

- v kvalite výstupu
- v univerzálnej dostupnosti
- v jeho stabilite.

V posledných niekoľkých rokoch stále viac a viac užívateľov prenieslo TEX z univerzít do priemyslu, kde bol postavený pred nové aplikácie [33]. Ale čas sa nezastavil a to, čo bolo vrcholom včera, môže byť zajtra už zastaralé. TEX je stále vzorom pre tie ciele, pre ktoré bol vytvorený, ale s rastúcimi skúsenosťami z niekoľkoročného používania môžeme lepšie porozumieť, kde neobstojí vo vysokej kvalite sadzby.

Ako výsledok tlaku užívateľov [27] oznámil Don Knuth v Stanforde novú verziu TEXu s priznaním skutočnosti, že nepredvídal potrebu pre 8-bitový vstup [19]. Súčasne dal najavo, že sa rozhodol odísť z tohto projektu a vrátiť sa k hodne opozdenému projektu „Umenia počítačového programovania“.

Takto je TEX v súčasnosti zakonzervovaný a každý jeho ďalší vývoj bude ústiť do iného systému, ktorý už nebude pod Knuthovým dohľadom. Preto hlavným cieľom tejto práce je podať prehľad požiadaviek pre vysokú kvalitu sádzania (bez ohľadu na to, či ich TEX spĺňa alebo nie), čím dúfame, že ovplyvníme budúci vývoj tak, aby neskončil niekoľkými nekompatibilnými systémami založenými na TEXu, ale snáď v jednom systéme s rovnakými charakteristikami (t.j. kvalita, prenosnosť a dostupnosť) ako súčasný program.

TEX bol navrhnutý ako formátovač nižšej úrovne, ako stabilné jadro sádzacieho systému, ktorého rozšírenia na oboch koncoch budú schopné akceptovať vývoj v technológii tlačenia (výstupný koniec) a spojenie s užívateľom (vstupný koniec) [14]. Sťažnosti na nepriateľskosť TEXu voči užívateľom sú neodôvodnené, lebo všetky požiadavky z tejto strany môžu byť ošetrené na vstupnom konci buď v reči TEXu (ako LATEX), a tým úplné prenositeľné, alebo pomocou externého jazyka, ako ArborText's Publisher alebo VAX Document, atď. Tieto systémy používajú TEX alebo sú založené na TEXu ako na konečnom formátovači a poskytujú užívateľsky priateľský interface [32].

Keď rozoberáme chýbajúce prvky, tak musíme obozretne rozlišovať medzi tým, čo môže a má byť ošetrené systémom na vstupnom konci, a tým, čo je naozaj úloha pre formátovač a nemôže byť ošetrené v TEXu 3.0. V nasledujúcich odsekoch budeme analyzovať požiadavky pre vysoko kvalitnú sadzbu a prediskutujeme, či môžu byť ošetrené TEXom alebo vhodným vstupným koncom, prípadne obidvoma. Ak takéto ošetrenie nie je možné, pokúsime sa nájsť cesty na dosiahnutie požadovaného výsledku. Konečne, v odseku 12 si všimneme samotný koncept jazyka TEX tým, že načrtneme základy toho, ako môže jazyk pre nový systém jasnejšie popisovať podkladové pojmy.

2. Zalamovanie riadkov

Algoritmus TEXu na zalamovanie riadkov je očividne ústrednou časťou celého jeho systému. Namiesto toho, aby zalamoval odstavec riadok za riadkom, algoritmus považuje odstavce za jednotky a hľadá ‚optimálne riešenie‘ na základe hodnôt niekoľkých parametrov. V dôsledku toho

porovnanie výsledkov vytvorených TEXom a iným systémom dopadne obyčajne v prospech TEXu.

Taký prístup má však svoje úskalia, najmä v situáciách vyžadujúcich viac než blok štýlového textu pevnej šírky. Konečné zalomenia riadkov sú určené v čase, keď informácia o obsahu aktuálneho riadku je stratená (aspoň z pohľadu TEXu, t.j. jeho vlastného makrojazyka), takže TEX neposkytuje žiadne dodatočné spracovanie konečných riadkov z hľadiska ich obsahu.

Ďalej neexistuje spôsob ovplyvniť tvar odstavca vzhľadom na aktuálnu pozíciu na stránke, pretože táto informácia nie je *a priori* známa. V odseku 4 budeme diskutovať na túto tému.

Používanie len štyroch kategórií (tight, decent, loose, very loose) na rozlíšenie nastavenia glue pre susedné riadky sa zdá trochu neadekvátne. Počet týchto kategórií by mal byť zvýšený. Navyše, globálnejší prístup (dokonca za hranice odstavcov za istých okolností), ktorý by vzal do úvahy celkovú zmenu nastavenia glue, by mohol viesť k lepším výsledkom.

2.1 Parametre zalamovania riadkov

Aj keď algoritmus poskytuje množstvo parametrov na ovplyvnenie layoutu, niektoré dôležité parametre pre kvalitnú sadzbu chýbajú. Neexistuje spôsob, ako niečo urobiť s vertikálnymi pásami (riekami) vytvorenými medzislovnými medzerami zapadajúcimi do tej istej vertikálnej pozície. Podobný problém zahŕňa rovnaké slová nad sebou, najmä na začiatku alebo na konci riadku. Obidva problémy sú rozptylujúce pre oči čitateľa a zničia každú námahu na vytvorenie krásne zalomeného odseku. Dobrý príklad je udaný v druhom odstavci časti 5, kde slovo „strane“ sa opakuje na dvoch riadkoch napravo pod sebou.

Ďalší aspekt ušľachtilej tlače je istota, že posledný riadok odstavca nebude príliš krátky. To je mimoriadne dôležité pri layoutoch, ktoré používajú odsadenie na začiatku odstavca, kde by vznikla nežiadúca medzera v prípade, že posledný riadok odstavca bol kratší než odsadenie nasledujúceho paragrafu. Pre väčšinu používateľov TEXu je neznáme, že tomuto môžeme zabrániť špeciálnym nastavením parametrov TEXu pre zalamovanie riadku tak, ako je to ukázané v príklade 1 časti 14. Kým zvyšné časti tohto článku používajú toto nastavenie, v tomto odstavci je takýto nežiadúci efekt viditeľný.

Delenie slov v po sebe idúcich riadkoch je ošetrené do rozsahu dvoch

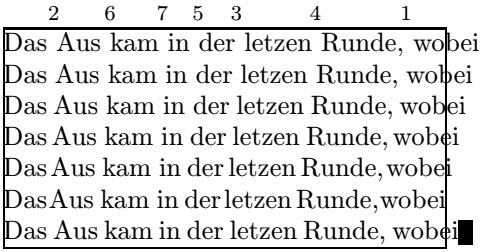
riadkov (`\doublehyphendemerits`), ale neexistuje možnosť za istých okolností zabrániť odstavcom takým ako tento a nasledujúci. Ako ľahko zbadáte, počet rozdelení v týchto odstavcoch je umelo zvýšený nastavením istých parametrov pre zalamovanie riadkov na nezvyklé hodnoty. Ale v neanglických jazykoch (s väčšou priemernou dĺžkou slova) predstavujú takéto situácie reálne životné problémy.

Iný problém predstavuje nezhoda medzi prvým a neskoršími riadkami odstavca, ktorá je dôsledkom implementácie odsadenia začiatku odstavca. Toto je podstatné v layoutoch s nulovým odsadením, pretože medzera na začiatku prvého riadku (napríklad z `\mathsurround`) nezmizne do okraja z dôvodu implicitného `\hbox`, ktorý vyjadruje odsadenie na začiatku odstavca (aj keď nie je viditeľné), zatiaľ čo rovnaká medzera bude zlikvidovaná na začiatku neskorších riadkov. To vedie k zvláštnym počiatočným medzerám.

3. Medzery

Keď je vytváraný blokový text, je nutné meniť medzislovnú alebo medziznakovú medzeru, alebo obidve. Kým medziznaková premenná medzera je prenechaná na expertov (až na malé výnimky), je úlohou zalamovacieho algoritmu naťahovať alebo zmenšovať medzislovnú medzeru od optimálnej hodnoty, ktorá je daná dizajnerom fontu, až pokiaľ nie je určená konečná poloha slova. TEX opäť má dobre navrhnutý algoritmus, ktorý berie do úvahy takúto rozťahovateľnosť. Navyše, každý znak má priradený tzv. `\spacefactor`, ktorý má vplyv na medzeru nasledujúcu za ním, čím je daná možnosť zväčšiť alebo zmenšiť medzeru medzi slovami za istými znakmi. Ako príklad porovnajte medzery za interpunkčnými znamienkami v tomto odstavci s ostatnými.

Ovšem neexistuje žiadne opatrenie na možnosť ovplyvnenia medzislovnej medzery vo vzťahu k aktuálnym znakom na obidvoch koncoch slova. Ak by bolo potrebné zúžiť daný riadok, nemali by sa zúžiť všetky medzery o rovnakú hodnotu. Namiesto toho by bolo lepšie zúžiť viac za čiarkou než napríklad medzi slovami ‚kam in‘ (viď obrázok 1) z dôvodu rôzneho tvaru znakov. Neexistuje spôsob ako dosiahnuť také jemné vyladenie v TEXu, s výnimkou manuálneho dodania `\hskip` v riadkoch, ktoré sú netolerovateľné. Jeden príklad takého prístupu je vidieť na obrázku 1. Takýto mechanizmus je samozrejme závislý od fontu a jeho implementácia by preto zmenila ako TEX, tak aj METAFONT, lebo najlepšie miesto, kde uložiť takúto informáciu, je `TFM` súbor. Ale aj tabuľky podobné k `\sfcode` (fixované formátom alebo makrom) by boli veľkým



Obrázok 1: Medzislovné medzery

Medzislovné medzery sú tak očíslované, že väčšie čísla označujú medzery, ktoré by sa menej zmenšili použitím pravidla podľa Siemoneita [28]. Posledný riadok ukazuje výsledný overfull box vytvorený v tejto situácii štandardným TEXom.

zlepšením, lebo väčšina používaných fontov má tendenciu mať podobné tvary znakov.

V TEXovskej koncepcii pre nastavovanie glue je urobený dôležitý rozdiel medzi glue pre naťahovanie a sťahovanie: kým to posledné je povolené len po isté pevné minimum (t.j. *natural width* mínus *shrink component*), každá hodnota naťahovacieho glue je automaticky povolená po ľubovoľnú hodnotu.[1]) Dôvodom pre toto správanie je, že toto umožňuje zalamovaciemu algoritmu dosiahnuť ‚núdzové výsledky', ak nie je nájdené iné vhodné zalomenie. Toto je však nežiadúce vo väčšine situácií, takže buď by malo byť ohraničené vo všetkých prípadoch naťahovanie podobne ako sťahovanie (výsledkom čoho by boli zmeny v algoritme pre zalamovanie riadkov a strán), alebo by mala byť dodaná ďalšia trieda glue, pre ktorú by veľkosť naťahovania mohla byť určená individuálne.

Don Knuth [18, str. 394–395] podáva príklad ako dosiahnuť visiacu interpunkciu (spolu so špeciálnymi fontami, ako poznamenáva). Pretože toto je tiež znakom dobrej kvality sadzby, je otázne, či takáto schéma (ktorá urobí mechanizmus ligatúry čiastočne nepoužiteľný, spolu s inými efektmi) je vhodná, alebo či toto by nemalo byť priamym prvkom budúceho programu.[2])

[1]) Za normálnych okolností je však tomuto zabránené badness funkciou.

[2]) Počínajúc nasledujúcim odstavcom, používa tento článok visiacu interpunkciu. Zmena kvality je zreteľná, aj keď zlepšenie je možné jemnou adjustáciou všetkých znakov (napr. posunutím ‚r' máličko von, a pod.) dosiahnuť dokonalé zarovnanie.

4. Zalamovanie strán

Veľký problém v TEXu predstavuje jeho algoritmus na zalamovanie strán. Zalamovanie strán je obhospodarované asynchrónne presúvaním istých položiek v isté momenty zo zoznamu aktuálnych príspevkov na „aktuálnu stranu“, pokiaľ nie je tento zoznam naplnený viacerými jednotlivosťami, než sa to hodí na danú stranu v konečnej forme. Konečné zalomenie strany je zvolené po zvážení miery špatnosti (ako plná je strana, ak ju zalomíme na tomto mieste) a hodnoty pokút (ako drahé je to zalomiť práve tu). Takéto penalizácie sú uložené po niektorých riadkoch buď algoritmom na zalomenie riadku, alebo počas rozvoja makra.

Ovšem dobrý stranový layout vyžaduje vzatie do úvahy dvojíc oproti sebe stojacich strán tak, ako sú tieto videné čitateľom. Toto samo o sebe nie je reálnym obmedzením, lebo dvojstrana môže byť chápaná ako obrovský prípad dvojstĺpcového formátu, samozrejme za predpokladu, že obidve strany sa súčasne zmestia do pamäti. Nanešťastie žiadny z vnútorných mechanizmov TEXu nevie vhodne obhospodáriť viacstĺpcový layout, takže takýto prístup musí obísť všetky vnútorné prvky pre zalamovanie strán ako `\insert` a pod. Dobrými príkladmi, ktoré tiež ukazujú obmedzenia TEXu v tomto smere, sú výstup LATEXu [22] a implementácia viacstĺpcového layoutu [4, 24], hraničiace s nemožným.

Ale čo je viac dôležité, algoritmus na zalamovanie riadkov v spojitosti s algoritmom na zalamovanie strán predkladá neriešiteľné problémy. Keď je TEXom zvolené zalomenie strany, všetky odstavce, ktoré boli raz kandidátom pre aktuálnu stranu, sú už oddelené algoritmom na zalomenie riadku a toto oddelenie nemôže byť zrušené pre text prenesený na nasledujúcu stranu, lebo isté nutné informácie (napríklad medzera v mieste zalomenia riadku) sú stratené. Toto robí nemožným zmenu layoutu strany na istých miestach, napr. na vrchu novej strany, pre vynechanie miesta pre malý obrázok obkolesený textom. Len za veľmi obmedzených okolností môže byť nájdené riešenie vo vnútri TEXu [9], ale dokumenty priemernej zložitosti nemôžu byť spracované takýmto spôsobom. Všeobecné riešenie tohto problému môže byť zahrnuté do súčasného TEXu kompatibilným spôsobom smerom hore. Istý prototyp bol navrhnutý na univerzite v Mohuči krátko po konferencii v Stanforde [25].

Je otvorenou otázkou, či vôbec ostaneme pri TEXovskom zalamovacom algoritme pre strany, lebo bol nedotiahnutý z dôvodu pamäťových a časových obmedzení počítačov dostupných v čase jeho vývoja. Vo svojej PhD práci [26] vyšetroval M. Plass niekoľko globálnych optimalizačných stra-

tégií používajúcich dvojprechodový systém. Jeho výsledky otvorili široké pole pre ďalší výskum. Niektoré z jeho myšlienok boli použité v Type & Set systéme [3].

Hlavným príspevkom TEXu 82 k počítačovej sadzbe bol krok od riadok za riadkom odstavec zalamujúcim algoritmom ku globálne optimalizujúcemu algoritmu.[3]) Ďalším cieľom pre budúce systémy by malo byť vyriešenie podobného, ale komplexnejšieho problému globálneho zalamovania strán.

5. Layout strán

Pre úpravu strán poskytuje TEX koncepciu výstupných rutín spolu s vkladaniami (inzerciami) a značkami (márkami). Koncepty vkladania a značkovania sú šité pre potreby relatívne jednoduchého modelu layoutu strán, ktorý obsahuje len jeden stĺpec, poznámky pod čiarou a najviac tu a tam jednoduchý obrázok.[4])

Značkovací mechanizmus poskytuje akúsi informáciu o istých objektoch a ich relatívnom poradí na aktuálnej strane, alebo špecifickejšie, informáciu o prvom a poslednom z týchto objektov na aktuálnej strane a o poslednom z týchto objektov na ľubovoľnej predchádzajúcej strane. Taká informácia je nevyhnutná na konštrukciu určitých typov záhlaví, napr. takých s menom aktuálnej kapitoly alebo s informáciou o prvom a poslednom slove vysvetlenom na predchádzajúcej strane a pod.

Toto je globálny mechanizmus, avšak taký, že len jedna trieda objektov môže využívať výhody celého mechanizmu. Ak je implementovaná viac ako jedna trieda, niektoré prvky tohto mechanizmu sú stratené pre celú triedu.[5]) Ako dôsledok tohto nedostatku by bolo vhodné rozšíriť značkovací mechanizmus do systému nezávislých značiek, ktoré môžu byť alokované oddelene makro balíkom.

Zdá sa, že vkladací mechanizmus bol odvodený z ‚aplikácie poznámok

---

[3]) Musíme poznamenať, že podobný algoritmus bol nezávisle vyvinutý J. Achugbueom [2]. Porovnanie by mohlo viesť k ďalšiemu zvýkoneniu.

[4]) Termín ‚jednostĺpcový výstup' znamená, že celý text je skladaný z riadkov rovnakej šírky. Problémy s premenlivou šírkou riadku diskutujeme v časti 4. Toto samozrejme pokrýva široký rozsah viacstĺpcových layoutov, napr. spracovanie poznámok pod čiarou v originále tohto článku (viacstĺpcový layout). Ale podobný rozsah zaujímavých layoutov nie je definovateľný v TEXovskom modeli box – glue – penalty.

[5]) Implementácia LATEXu poskytuje rozšírený značkovací mechanizmus s dvoma druhmi nezávislých značiek s výsledkom, že jeden sa správa ako `\firstmark` a ten druhý ako `\botmark`. Informácia obsiahnutá v primitívnom pojme `\topmark` je stratená.

pod čiarou‘ a neskoršie rozšírený tak, aby umožnil istý jednoduchý typ pohyblivých vkladaní.[6]) Ovšem umiestnenie pohyblivých objektov vyžaduje viac než ich púhe uloženie v obrovskom boxe, z ktorého sa vyjmú v istom momente, keď je vyvolaná výstupná rutina. Pohyblivé objekty sú doprevádzané titulkami a podobnými záležitosťami, ktoré vyžadujú rozličný prístup v závislosti od ich konečného umiestnenia na strane. Pohyblivé objekty sa odlišujú, aj keď prislušia do tej istej triedy. Na druhej strane, objekty uložené v jednej triede môžu ovplyvniť, alebo dokonca zamedziť umiestnenie pohyblivých objektov z iných tried.

Niektoré triedy vkladaní, ako marginálne poznámky, nemôžu byť vôbec obhospodárené pomocou primitívnych pojmov. Napríklad LaTeX pre poskytnutie takých možností definuje vlastnú správu pamäti pre pohyblivé objekty. Takýto mechanizmus je prirodzene pomalý a náročný na pamäť. Navyše je ďalej redukovaná kvalita zalomení strán, lebo je ťažké udržovať voľne všetku informáciu poskytovanú koncepciou vkladania.

Iným problémom je rozhodnutie, že zalamovací mechanizmus strán je, aspoň v jeho rozhodujúcej fáze, dostupný len vo vonkajšom vertikálnom režime, a tak napríklad priestor pre vsúvanie vkladaní sa neberie na zreteľ, keď sa delí `\vbox` pomocou `\vsplit`.

Pre navrhovateľa je TeXovský model medziriadkového určenia glue úplne neznámy, lebo nedovoľuje špecifikáciu medzery od základnej čiary k základnej čiare v stranovej špecifikácii bez použitia zdĺhavých a komplikovaných interných výpočtov (viď obrázok 2 na ďalšej strane). To znamená, že je skoro nemožné implementovať mrežovo orientované špecifikácie, t.j. kde (skoro) všetky účaria padnú na dopredu určené pozície. Tento článok používa mrežovo orientovanú špecifikáciu (ktorá je pripravená ručne), aby sme ukázali tento aspekt vysoko kvalitnej sadzby. Detaily sú udané v príklade 3.

Návrh vhodných primitívnych pojmov pre tento komplex musí ísť ruku v ruke s novým algoritmom pre zalamovanie strán, pravdepodobne predstavujúcim najdrastickejšiu zmenu v TeXovom systéme.

6. Penalizácia — rozhodovacia miera

Zalamovanie riadkov a strán v TeXu je hlavne determinované vážením „badness“ výsledného výstupu[7]) a penalizácie za zalomenie v aktuálnom

[6]) Toto je len dohad zo štúdia [17].

[7]) Toto je v istom zmysle miera rozdielu medzi optimálnym a aktuálnym množstvom bieleho priestoru v riadku a na strane akurát spracovávaných.

Obrázok 2: Priestory medzi účariami

Pre implementáciu dimenzie medzi účariami, napríklad medzi odstavcom a názvom (označené otáznikom), by hodnota `\parskip` mala byť určená v závislosti od `\baselineskip` druhého odstavca. Nanešťastie, používaná hodnota `\baselineskip` je aktuálna z konca druhého odstavca, zatiaľ čo `\parskip` musí byť vypočítaný na začiatku.

bode. Takáto penalizácia je buď zadaná ručne užívateľom (počas rozvoja makra), alebo dodaná neskoršie istými formátovacími rutinami TEXu.

Hlavný problém spôsobený implicitnými penalizáciami je, že tieto nemôžu byť odstránené. Ak napríklad algoritmus pre zalomenie riadku rozhodne uložiť penalizáciu po riadku (napr. z `\widowpenalty`), neexistuje spôsob, ako zamedziť na tomto mieste zalomeniu strany prostredníctvom rozvoja makra, prirodzene s výnimkou nastavenia predmetnej penalizácie na nekonečno. Toto je výsledkom TEXovského algoritmu, že následné penalizácie p_1 a p_2 sa správajú ako $p_3 := \min(p_1, p_2)$.

Lokálne zmeny prehrešujúcich sa penalizačných parametrov majú tendenciu k chybám a sú časovo náročné, lebo za každou zmenou môže nasledujúce zalomenie strany padnúť na inú pozíciu. Navyše, budúce editovanie dokumentu je zložitejšie, lebo každá korekcia tohto typu môže produkovať nežiadúce výsledky za každou jednotlivou zmenou.

Ak myslíme v reláciách súčasného TEXu (t.j. predpokladajúc, že všetky hlavné algoritmy zostanú nezmenené), tak by bolo lepšie prijať inú stratégiu v prípade za sebou nasledujúcich penalizácií, buď $p_3 := \max(p_1, p_2)$, alebo $p_3 := \frac{1}{2}(p_1+p_2)$. Pri obidvoch funkciách vyžadujú hraničné prípady $p_i = \pm\infty$ špeciálnu starostlivosť. Reťazec penalizácií môžeme preraziť ako obyčajne zoskupením alebo pomocou `\kern0pt`, alebo podobnými procedúrami, ako sa už používajú pri ligatúrach, atď.

Ako sme už povedali, toto je riešenie v rámci TEX82. Ak bude na-

vrhnutý úplne iný algoritmus pre zalomenie strán, koncepcia lokálnej penalizácie by mala byť tiež preskúmaná a pravdepodobne nahradená inou stratégiou.

7. Rozdeľovanie slov

Pri sadzbe textu, špeciálne s úzkymi stĺpcami, je rozdeľovanie slov nevyhnutné, aby sa zabránilo nečitateľným medzerám. Ovšem čitateľnosť má veľa podôb, jedno zo základných pravidiel hovorí [28]: „Vyhni sa viac než dvom rozdeleniam v riadkoch za sebou.“ Ako sme už uviedli v časti 2, toto nie je možné špecifikovať v TEXu (pokiaľ dokonca úplne neznemožníte dve rozdelenia za sebou).

Iným problémom je rozdeľovanie slov na miestach, ktoré sú síce prípustné, ale menia význam:

Stiefel-tern	Steif-eltern
Spargel-der	Spar-gelder[8])

Mali by sme pravdepodobne vždy zabrániť takýmto problematickým rozdeleniam zvolením vhodných vzorov v Liangovom algoritme [23]. Čitateľnosť však môže byť narušená aj rozdelením veľmi krátkych slabík, ktoré nedávajú skoro žiadnu informáciu o rozdelenom slove, a preto spomaľujú proces čítania. V TEXu 3.0 je teraz možné nastaviť minimálny počet písmen napravo a naľavo od rozdeľovacieho znamienka. Toto je dôležité pre veľa jazykov, ktoré majú často dlhé slová a napríklad veľa dvojpísmenových slabík podobne ako nemčina. Neexistuje ale spôsob ako priradiť váhu rôznym rozdeľovacím miestam, t.j. jedná sa o jednoduchú áno alebo nie situáciu. Jeden z možných spôsobov riešenia tejto situácie by mohol byť prostredníctvom novej triedy demerits

$$\frac{\text{užívateľova hodnota}}{\text{dĺžka zalamovanej časti}}$$

alebo podobnej funkcie. Pritom ovšem musíme pravdepodobne vo formule rozlišovať medzi textom pred zalomením a po ňom (a/alebo jeho dĺžkou).

Pretože kvalita niektorých miest zalomenia tiež závisí aj od slova (t.j. významu slovných častí), mali by sme premýšľať, či by taká informácia mala byť poskytovaná rozdeľovacím algoritmom.

[8]) Slová znamenajú „nevlastní rodičia“ a „úspory“, kým prvé časti slov v ľavom stĺpci znamenajú ‚čižmy‘ a ‚špargľa‘.

8. Rotácia boxov

Koncepcia TEXovskej reprezentácie dokumentov je orientovaná prísne horizontálne a zľava doprava. Popri probléme spracovania dokumentov, ktoré obsahujú jazyky orientované sprava doľava alebo zhora dole (ktoré môžu byť do istej miery spracované špeciálnymi verziami TEXu [21]), spôsobuje toto v štandardných aplikáciách tiež zbytočné obmedzenia. Odhliadnuc od použitia `\special` (pre PostScriptové zariadenia) je nemožné otočiť isté časti dokumentu. Kým ľubovoľné otočenie je naozaj temer nemožné, pre väčšinu výstupných zariadení môže byť otočenie o 90° urobené jednoduchým spôsobom otočením znakových buniek. Malo by byť jednoduché zahrnúť istý druh primitívneho pojmu `\rotate` do jazyka TEXu, ktorý by dovolil otáčanie vodorovných a zvislých boxov o násobky 90 stupňov.[9]) To by umožnilo zahrnutie otočených tabuliek a pod. do dokumentov bez toho, aby sme potrebovali skutočné lepidlo a nožnice na pridanie čísla strany alebo záhlavia.

9. Informácia o fontoch

ISO Draft Standard [1] obsahuje stovky charakteristík popisujúcich fonty. Zatiaľ čo niektoré z nich sú dostupné v TEXu prostredníctvom `\fontdimen`, väčšina z nich nie je. Zdá sa, že by bolo záhodné pridať viac z nich do množiny parametrov TEXu (napríklad doporučené `\baselineskip`), aby sa dali robiť jednoduchšie zmeny v triedach fontov.

9.1 Virtuálne fonty

Použitie tried fontov v TEXu, ktoré sa odlišujú v pozícii znaku, atď. od štandardu z triedy Computer Modern, je ťažké, ale môže byť dosiahnuté, ako to bolo ukázané v niekoľkých projektoch [7, 35]. Navrhované použitie virtuálnych fontov [20] môže zjednodušiť mnohé záležitosti v tomto smere. Keby bolo možné dohodnúť sa na štandardoch pre pozíciu a na spôsobe prístupu k istým znakom s diakritikou v najspoločnejšej abecede latinkou píšúcich jazykov, tak by bolo možné sádzať viacjazyčné dokumenty (použitím nových prvkov TEXu 3.0) bez zavádzania zbytočných variant štandardných fontov, odlišujúcich sa len v dostupnosti istých znakov s diakritikou ako ‚reál-

[9]) Toto by vyžadovalo zmeny v `dvi` jazyku, a preto zmeny v programoch všetkých ovládačov.

nych‘ znakov.[10]) Dobrý prehľad znakov s diakritikou v jazykoch používajúcich latinku s návrhom k ich prístupu cez ligatúry dal Haralambous [8].

9.2 Ligatúry a kerny

Ligatúry a kerny sa na nešťastie odlišujú od jazyka k jazyku. Uveďme ako príklad ligatúru ffl, ktorá sa nepoužíva v tradičných nemeckých dokumentoch. Na druhej strane tieto dokumenty používajú ligatúry ch, ck a ft na dosiahnutie lepšieho písma. Nasledujúce príklady ukazujú rozdiel:

Druckschrift	Druckschrift	(štandardne)
Druckschrift	Druckschrift	(nemecká ligatúra)

Pretože tieto špeciálne ligatúry nezahŕňajú nové tvary písmen (aspoň nie vo väčšine tried fontov), je možné dosiahnuť žiadúce výsledky jednoducho použijúc kerning. V triede fontov Computer Modern [15] sú obidva prípady ch a ck zahrnuté do kerningových programov, ale len pre písmo patkové. Iné triedy fontov vykazujú podobné nedostatky. Preto na sádzanie nemeckých dokumentov buď potrebujeme špeciálne fyzické fonty (alebo aspoň virtuálne fonty), alebo spôsob na manipuláciu s ligatúrami a kerningovými programami v rámci programu TEX. Z dôvodu prenosnosti sa zdá, že ovládateľný prístup k ligatúrnym a kerningovým programom počas zavádzania fontov je lepší.

10. Tabuľky

Pekne proporcionálne tabuľky je ťažko sádzať aj pre skúseného ručného sadzača. Primitívne pojmy TEXu `\halign` a `\valign` robia báječnú službu v tomto smere, a to i v komplikovaných situáciách. Existuje však dôležitá podtrieda tabuliek, s ktorou sa vôbec ťažko pracuje, s výnimkou možnosti ručného prispôsobovania. Nie je možné špecifikovať kombináciu horizontálne a vertikálne rozpätých stĺpcov, napr. otvorenú krútenú zátvorku cez niekoľko riadkov v jednom stĺpci, kým na obidvoch jej stranách pokračuje riadková štruktúra.

Iným často požadovaným prvkom sú tabuľky zaberajúce niekoľko strán. Toto je ťažko dosiahnuť, lebo primitívne príkazy TEXu pre tabuľky za normálnych okolností prečítajú celú tabuľku, kým určia šírku stĺpca atď.

[10]) Použitie diakritických znakov ako primitívnych pojmov sa nedoporučuje pre štandardnú diakritiku jazyka [18, str. 54] z dôvodu znemožnenia rozdeľovania slov.

Nespôsobuje to ale neriešiteľné problémy s pokročilými prvkami TEXu 3.0.

11. Matika

Matematická sadzba je jednou z hlavných TEXovských domén, kde ju žiadny iný automatický sádzací systém nebol schopný dohoniť. Ale aj v tejto oblasti by sa dalo niekoľko vecí zlepšiť.

Zdrojový program $\mathcal{AMS}$-TEXu [30] poskytuje veľa zaujímavých príkladov, kde Spivak preľstil obmedzenia TEXovských formátovacích pravidiel vytvorením komplexného programu na definovanie funkcií, ktoré majú za úlohu realizovať štandardné úlohy matematickej sadzby. Detailná analýza týchto problémov (dvojitá diakritika, spodná diakritika, umiestnenie číslovania rovníc, atď.) by ľahko zaplnila niekoľko strán; isté poznámky môžete nájsť v Spivakovej dokumentácii [29].

Kým TEXovské pravidlá pre medzerovanie v matike sú celkom dobré, zdá sa, že je prinajmenšom otázne, ak množstvo z nich je natvrdo zadrôtovaných do programu namiesto toho, aby boli dosiahnuteľné prostredníctvom parametrov. Tabuľka pre medzerovanie medzi rôznymi matickými atómami je pravdepodobne najdôležitejší príklad tohto druhu.

Iným problematickým prvkom TEXovských sádzacích rutín pre matiku je, že subformuly sú vždy zaboxované v prirodzenej šírke, a to aj vtedy, keď najvyššia úroveň math listu je predmetom roztahovania alebo sťahovania. To môže za určitých okolností viesť k ohyzdným výsledkom. Koncepcia zaboxovania subformúl má dodatočnú nevýhodu, že tieto časti formúl nemôžu byť zalomené cez riadok. Preto programátorské štruktúry ako `\left...\right`, ktoré automaticky určujú výšku delimiterov s variabilnou výškou, nemôžu byť použité pri zložitých výstupoch.

12. Jazyk TEXu

Jazyk TEXu je rozdelený na dve časti, ktoré sa popisujú ako ústa a žalúdok TEXu [18]. Tento rozdiel je rozhodujúci vo veľa aplikáciách, lebo výstup produkovaný gastronomickými rutinami nemôže byť zavedený opäť do jeho úst, t.j. do jeho scanneru. V skutočnosti sú skonštruované boxy post-procesovateľné v ohraničenom rozsahu (cez `\lastbox`, `\unpenalty`, atď.), ale ľubovoľné konštrukcie nemôžu byť spracované týmto spôsobom, lebo chýbajú primitívne pojmy pre manipuláciu s prvkami ako písmená, linky, atď. Tento rozdiel vo všeobecnosti je dôvodom pre veľa prekážok

v TEXovskom programovaní, čo niekedy úplne znemožňuje akékoľvek riešenie. Nový systém by mal odstrániť túto dvojtriednu spoločnosť vnútorných príkazov.

Iným vážnym problémom jazyka je jeho neúplnosť ohľadne štandardných programovacích konštrukcií (takých, ako isté podmieňovacie príkazy, prijateľný aritmetický parser atď.), ako aj špeciálnych konštrukcií vhodných pre sadzbu. Napríklad na určenie dĺžky posledného riadku odstavca (čo je robené TEXom automaticky pri sadzbe samostatných formúl), musíme použiť komplikovaný a zdĺhavý výpočet tak, ako je ukázané v príklade 2. Tento príklad tiež ukazuje jednu nedôslednosť jazyka TEXu: `\prevgraf` musí byť menený použitím pomocného registra, lebo priame použitie príkazu `\advance` je zakázané. Veľa problémov tohto druhu nájdete v prílohe D *The TEXbooku* [18], str. 373–401, ktorá má názov „Špinavé triky“. V skutočnosti deväť z týchto desiatich príkladov je použitých v implementácii LATEXu [22], čo ukazuje, že tieto príklady sú zďaleka nie tak exotické, ako predslov tejto prílohy naznačuje. Ako príklad chýbajúcich programovacích konštrukcií uveďme podmieňovací príkaz typu `\ifmathopen`.

Takéto problémy vysvetľujú skutočnosť, že všeobecný aplikačný software napísaný v TEXu (ako LATEX) ľahko zaberie viac ako tretinu prístupnej pamäti bez toho, aby sme napísali jediné písmeno. Pre lepšie a stabilnejšie vstupné a výstupné konce potrebujeme jazyk, v ktorom takéto úlohy môžu byť špecifikované oveľa elegantnejším spôsobom.

Niektoré obmedzenia jazyka TEXu sú spôsobené reprezentáciou dimenzií ako ‚reálnych čísiel‘ vo väčšine inštalácii TEXu, ktoré sú strojovo závislé.[11]) Na zabezpečenie strojovej nezávislosti TEXu sa Knuth pokúsil predísť tomu, aby sa strojovo závislé výsledky generované v žalúdku TEXu rozplynuli do častí prístupných scanneru, alebo aby vnútorne ovplyvňovali akékoľvek rozhodnutia o zalamovaní riadkov alebo strán. Výsledkom tejto stratégie je, že použitie programu z príkladu 2 spolu s konečným `\parfillskip` (ako v príklade 1) skoro vždy produkuje hodnotu `\maxdimen` namiesto decentnej.[12]) Z toho istého dôvodu povedal Knuth v rozhovore s autorom na univerzite v Stanforde, že nemôže dovoliť odstránenie ľubovoľných položiek v zostrojovaných zoznamoch, lebo toto by viedlo k prístupu k aritmetike s pohyblivou rádovou čiarkou.

[11]) Toto sa de facto vzťahuje len na interné dimenzie reprezentujúce naťahovanie a sťahovanie glue, počítaných kernov pre diakritiku a niektorých ďalších prípadov.

[12]) Dôvod je daný v module 1148 programu TEX [16, str.470].

Ovšem existuje iný spôsob ako dosiahnuť strojovú nezávislosť, ktorý by tiež odstránil uvedené obmedzenia, a to prechod od aritmetiky v pohyblivej rádovej čiarke k aritmetike s pevnou rádovou čiarkou,[13]) ktorý môže byť urobený jednoduchým spôsobom, ako sám Knuth s poďakovaním kvitoval [16, str. 46].

TEX je makrojazyk so všetkými výhodami i nevýhodami. Ktokoľvek raz napísal relatívne dlhší program v TEXu, vie, že jeho odladovanie je mimoriadne ťažké. Prehľadné programovanie tak, ako bolo navrhnuté autorom TEXu [10], je temer nemožné: nie je problém napísať tri riadky programu v TEXu, ktorý je nezrozumiteľný dokonca pre TEXperta bez toho, že by si ho prečítal dvakrát trikrát. Je však ešte ťažšie napísať program v TEXu, ktorý vykoná požadovanú akciu, aby tento bol zrozumiteľný pre priemerného užívateľa. Príklady uvedené v časti 14 sú dobrými testovacími príkladmi; sú to jednoduché programy v TEXu, ale ich presné pochopenie je ťažké bez vysvetľujúceho textu.

Veľa problémov povstáva z dizajnérskych rozhodnutí založených na úplne rozdielnych sémantických konštrukciách, ktoré majú podobnú alebo identickú syntaktickú štruktúru. Najdôležitejšími príkladmi sú krútené zátvorky a znak dolára.[14]) Krútené zátvorky sa používajú jednak na delimitáciu argumentov počas rozvoja makier, a jednak ako začiatok a koniec blokovej štruktúry, ktorá definuje rozsah istej deklarácie. V matematickom móde majú dodatočný význam delimitovania rozsahu subformúl. Dva za sebou stojace doláre obyčajne začínajú a končia výpis formúl na samostatnom riadku, ale v obmedzenom horizontálnom móde označujú jednoducho prázdnu matematickú formulu. Takéto koncepcie by mali byť z dôvodu jasnosti rozmotané.

Jazyk TEXu je vhodný pre jednoduché programátorské práce. Je to ako krok od strojového jazyka (formátovača) k asembleru. Pre komplexné programátorské úlohy všeobecného aplikačného softwaru, najmä z hľadiska logicky označených dokumentov [5], bol by výhodnejší mocný jazyk s dobre definovanou koncepciou väzby premenných, procedúr atď. Kým tento aspekt môže byť dosiahnutý na vstupnom konci programovacieho jazyka (ktorý sa skompiluje do jazyka TEXu), je lepšie, z dôvodu prenosnosti, zahrnúť ho do jadra TEXu. Podľa autorovho názoru, ideálny jazyk by mal kombinovať výhody procedurálneho jazyka s prospešnými

[13]) Snáď použitie vždy toho istého programu pre aritmetiku v pohyblivej rádovej čiarke (buď dostupného v knižnici kompilátora, alebo simulovaného programom) by bolo ešte lepšie.

[14]) Aby sme boli presnejší, tri znaky s `\catcode` jeden, dva a tri.

črtami interpreteru.[15]) Vedľajším efektom by bolo, že takýto jazyk by bol čiastočne kompilovateľný.

13. Záver

Súčasný TEX nie je dostatočne mocný na realizovanie všetkých potrieb vysokokvalitnej (ručnej) sadzby. Autor zdieľa Knuthov sen o stabilnom formátovači na úrovni nižšieho jazyka, ktorý je schopný vytvárať dokumenty najvyššej kvality. Na rozdiel od Knutha však chápe súčasný TEX len ako veľmi dobrý prototyp na ceste k tomuto cieľu.

Ako je to naznačené v tomto článku, veľa dôležitých koncepcií sadzby vysokej kvality nie je podporovaných TEXom 3.0. Pre ďalší výskum je nutné navrhnúť sádzací jazyk, ktorý vie vhodne riešiť tieto problémy.

Pospolitosť užívateľov TEXu potrebuje prístupnosť a nezaujatosť pre nový rozvoj, ktorý udrží ‚systém TEX' vrcholovým v oblasti počítačovej sadzby. Keďže Knuth už nie je zapojený do výskumu v typografii, je dôležité pre TUG, aby sa zjednotila v *podpore* a *pestovaní* ‚najlepšieho sádzacieho programu' a nielen v propagovaní programu, ktorý dal Knuth svetu. Ak sa nezmobilizujeme pre ešte lepšiu kvalitu, môže naša pospolitosť upadnúť do bezvýznamnosti.

Jeden dôležitý krok pre TUG by bolo inicializovať a (ak je to vhodné) podporovať ďalšie výskumné projekty, ktoré zdvihnú rukavicu hodenú Stanfordským projektom.

14. Príklady

Príklad 1. Nasledujúci program môže byť použitý na zabránenie skoro prázdnych riadkov na konci odstavca:

```
\parfillskip \hsize
\advance\parfillskip by -1.5\parindent
\advance\parfillskip by 0pt minus \parfillskip
\advance\parfillskip by 0pt minus -1em
```

Toto nastavenie bolo použité v tomto článku a napríklad viedlo k istým zmenám v druhom odstavci abstraktu. So štandardným nastavením (napr. `0pt plus 1fil`) by tento odstavec ukončil časťou slova ‚kov' (od ‚ro-kov').) Nanešťastie toto riešenie nie je tiež dokonalé, lebo vyprodu-

[15]) Akýsi druh jazyka lispovského typu, ale s primitívnymi pojmami vhodnými pre sadzbu.

kuje úsmevný výsledok s riadkami pozostávajúcimi z dvoch krátkych slov.

Príklad 2. Nasledujúci program určuje dĺžku posledného riadku predchádzajúceho odstavca, využívajúc istý prvok TEXu, ktorý je zabudovaný do vypisovania matematických formúl na samostatný riadok. Tento program môže byť určený napríklad na určenie množstva bieleho miesta pred zoznamom jednotlivostí alebo niečoho podobného. Ukážkový program nie je naozaj vhodný na priame použitie takéhoto druhu, lebo jednoducho vypíše nájdenú hodnotu na terminál. Dá sa však ľahko rozšíriť.

```
\def\getlastlinewidth{\ifhmode $$%
  \predisplaypenalty\@M \postdisplaypenalty\@M
  \abovedisplayskip-\baselineskip \belowdisplayskip\z@
  \abovedisplayshortskip\abovedisplayskip
  \belowdisplayshortskip\belowdisplayskip
  \showthe\predisplaysize
  $$\count@\prevgraf \advance\count@-\thr@@
  \prevgraf\count@ \else\typeout{*Not hmode}\fi}
```

Tento príklad ilustruje niekoľko dôležitých vecí. Po prvé, neexistuje žiadna elementárna metóda na výpočet takýchto dôležitých informácií. Po druhé, je to jeden (nie neobvyklý) z prípadov, keď informácia o sádzacom procese môže byť získaná jedine zavedením nežiadúcej (zrejme miesto zaberajúcej) penalizácie, glues a nulových boxov vo výstupe.

Príklad 3. Pri zavedení mrežovo orientovanej špecifikácie musia byť všetky flexibilné glue odstránené (s výnimkou `\skip\footins`) a `\vsize` musí byť adjustovaná. Nadpisy sú nastavené s rozpaľom `8pt + 4pt=` `\baselineskip` a musíme zabezpečiť, že vrchný priestor je zachovaný po každom zalomení strany. Zoznamy sú nastavené s `6pt + 6pt`, takže vnútorné linky sú o polovicu riadku posunuté. Zalomenia strán vo vnútri zoznamov vyžadujú špeciálny prístup, napr. zvýšenie `\topskip` na udržanie podmriežky. Obrázky a príklady s rozdielnymi typmi veľkostí sú vyvážené a nevyhnutné kerny pridané na udržanie okolitého materiálu v súlade. Tento prístup opäť funguje, jedine ak nezasiahne žiadne zalomenie strany, čo je aj prípad tohto článku. Pri použití TEXovského počítania špatnosti na určenie zalomenia strán môže byť použité naťahovacie `\topskip`. Počas výstupnej rutiny musí byť toto extra natiahnutie opäť zrušené.

Literatúra

1. *Information Processing — Font InformationInterchange, ISO/IEC JTC 1/SC 18/WG8 N1036*, February 1990
2. Achugbue, James O. "On the line breaking problem in text formatting." *Proc. of the ACM SIGPLAN/SIGOA,* 2 (1, 2), 1981.
3. Asher, Graham. "Type & Set: TEX as the engine of a Friendly Publishing System." In *TEX applications, uses, methods*, 91–100, Malcolm Clark [6].
4. Benson, Gary, Debi Erpenbeck, and Jannet Holmes. "Inserts in a multiple-column format." In *1989 Conference Proceedings*, 727–742, Christina Thiele [31].
5. Bryan, Martin. *SGML: an author's guide to the standard generalized markup language*, Addison-Wesley, Woking, England; 1988, Reading Massachusetts, second edition.
6. Clark, Malcolm, editor. *TEX applications, uses, methods*, Chichester, West Sussex, England; 1990, Ellis Horwood Limited. Exeter conference July 1988.
7. Conrad, Arvin C. "Fine typesetting with TEX using native autologic fonts." In *1989 Conference Proceedings*, 521–528, Christina Thiele [31].
8. Haralambous, Yannis. "TEX and latin alphabet languages." *TUGboat,* 10 (3), November 1989, 342–345.
9. Hoenig, Alan. "Line-Oriented Layout with TEX." In *TEX applications, uses, methods*, 159–184, Malcolm Clark [6].
10. Knuth, Donald E. "Literate programming." *The Computer Journal,* 27 1984, 97–111, An expository introduction to WEB and its underlying philosophy.
11. Knuth, Donald E. *Computers & Typesetting*, Addison-Wesley, Reading, Massachusetts; 1986, Consists of [18, 16, 13, 12, 15].
12. Knuth, Donald E. "METAFONT: The Program." In *Volume D of Computers & Typesetting* [11], 1986.
13. Knuth, Donald E. "The METAFONTbook." In *Volume C of Computers & Typesetting* [11], 1986.
14. Knuth, Donald E. *Talk given at Gutenberg Museum Mainz*, September 1987.
15. Knuth, Donald E. "Computer Modern Typefaces." In *Volume E of Computers & Typesetting* [11], July 1987, Reprint with corrections.

16. Knuth, Donald E. "TEX: The Program." In *Volume B of Computers & Typesetting* [11], May 1988, Reprint with corrections.
17. Knuth, Donald E. "The errors of TEX." In *Technical Report STAN-CS-88-1223*, Stanford University, Department of Computer Science, Stanford, California 94305; September 1988.
18. Knuth, Donald E. "The TEXbook." In *Volume A of Computers & Typesetting* [11], May 1989, Eighth printing.
19. Knuth, Donald E. "The now versions of TEX and METAFONT." *TUGboat,* 10 (3), November 1989, 325–328.
20. Knuth, Donald E. "Virtual Fonts: More fun for Grand Wizards." *TUGboat,* 11 (1), April 1990, 13–23.
21. Knuth, Donald E. and Pierre MacKay. "Mixing right-to-left text with left-to-right text." *TUGboat,* 8 (1), April 1987, 14–25.
22. Lamport, Leslie. `latex.tex`, February 1990, LATEX source version 2.09.
23. Liang, Franklin Mark. "Word Hy-phen-a-tion by Com-put-er." In *PhD thesis*, Stanford University, Depertment of Computer Science, Stanford, CA 94305; August 1983, Report No. STAN-CS-83-977.
24. Mittelbach, Frank. "An enviroment for multicolumn output." *TUGboat,* 10 (3), November 1989, 407–415.
25. Mittelbach, Frank. "Letter to Don Knuth." In *Suggestions for the TEX 3.0 release*, September 1989, Published by R. Wonneberger in [34].
26. Plass, Michael Frederick. "Optimal Pagination Techniques for Automatic Typesetting Systems." In *PhD thesis*, Stanford University, Depertment of Computer Science, Stanford, CA 94305; June 1981, Report No. STAN-CS-81-970.
27. Rynning, Jan Michael. "Proposal to the TUG meeting at Stanford." *TEXline,* 10 May 1990, 10–13, Reprint of the paper that triggered TEX 3.0.
28. Siemoneit, Manfred. *Typographisches Gestalten*, Polygraph Verlag, Frankfurt am Main; 1989, second edition.
29. Spivak, Michael. `amstex.doc`, 1990, Comments to [30].
30. Spivak, Michael. `amstex.tex`, 1990, $\mathcal{AMS}$-TEX source version 2.0 (without comments).
31. Thiele, Christina, editor. *1989 Conference Proceedings*, volume 10#4 of TUGboat, December 1989, TEX Users Group.
32. Wittbecker, Alan E. "TEX enslaved." In *1989 Conference Proceedings*, 603–606, Christina Thiele [31].

33. Wonneberger, Reinhard. "TEX in an industrial enviroment." In *Brüggemann-Klein, Anne, editor, 1989 EuroTEX*, Conference Proceedings1990, To appear.
34. Wonneberger, Reinhard. "TEX yesterday, today and tomorrow." *TEXhax*, 90 (5), January 7, 1990.
35. Youngen, R. E., W. B. Woolf, and D. C. Latterner. "Migration from computer modern fonts to times fonts." In *1989 Conference Proceedings*, 513–519, Christina Thiele [31].

Frank Mittelbach
Electronic Data Systems (Deutschland) GmbH
Eisenstraße 56 (N 15), D-6090 Rüsselsheim,
Spolková republika Nemecko
Tel. +49 6142 803267
Bitnet: `pzf5hz@drueds2`

z anglického originálu (TUGBOAT 11 (1990), 337–345)
přeložil *Štefan Porubský*

Grafika v TEXu (2)

PETR SOJKA

V dnešním pokračování seriálu o grafice v sázecím systému TEX se zaměříme na grafické možnosti jazyka POSTSCRIPT[1]) a na způsoby integrace grafiky vytvořené v jazyce POSTSCRIPT do TEXových dokumentů.

POSTSCRIPT

Jazyk POSTSCRIPT je poměrně jednoduchý programovací jazyk s bohatými grafickými možnostmi. Byl vyvinut firmou ADOBE za účelem popisu vzhledu textu, grafiky a bitových map na tištěné stránce (POSTSCRIPT) nebo stránce zobrazované na obrazovce (DISPLAY POSTSCRIPT). Popis stránky je nezávislý na používaném zařízení (tiskárně,

[1]) POSTSCRIPT, DISPLAY POSTSCRIPT, ADOBE jsou ochranné známky firmy ADOBE Systems Incorporated.