Hrstka tipů pro TEXovskou sazbu

Petr Březina

Rozsah vyjádřený pomlčkou

Jednou z věcí, kterými se odlišuje knižní sazba od strojopisu, je pomlčka. Zatímco se na psacím stroji používá pro spojovník a pomlčku stejný znak, ve fontech knižního písma nalezneme kromě spojovníku dokonce dvě pomlčky. Jedna je dlouhá (čtverčíková) a druhá je krátká (půlčtverčíková). V anglických textech se každá z těchto dvou pomlček objevuje v jiné funkci. V české sazbě se však ve všech funkcích používá zpravidla jen jedna pomlčka. Začátečníci bývají na rozpacích, kterou pomlčku si mají vybrat. Podíváme-li se do starších českých knih, zjistíme, že mistři sazeči dávali přednost spíše dlouhé pomlčce. V současnosti se v knihách velmi často objevuje krátká pomlčka. Já mám však raději dlouhou pomlčku a tu také obvykle používám při sazbě svých dokumentů.

Pomlčka může mít v textu řadu funkcí. Jednou z nich je vyjádření rozsahu; to znamená, že pomlčka zastupuje slůvko „až“. V takovém případě se pomlčka klade bez mezer mezi dva výrazy, které označují začátek a konec rozsahu, např. *v letech 1985—1993*. Takto ovšem vznikne poměrně dlouhý celek, který by mohl při řádkovém zlomu činit potíže. Proto je dovoleno v případě potřeby provést řádkový zlom v místě pomlčky; pomlčka ze sazby zmizí a místo ní se do textu vloží ono již zmíněné slůvko „až“, jak to můžete vidět např. zde: *v letech 1985 až 1993*. V TEXu lze definovat makro `\až` tak, že uživatel bude moci do zdrojového textu svého dokumentu zapsat např. `v~letech 1985\až 1993` a TEX sám rozhodne, zda se v daném případě vysází pomlčka, anebo slůvko „až“. Nejdříve se podíváme na to, jak bývá obyčejně toto makro definováno, a potom nabídneme jinou definici tohoto makra.

Pro řešení uvedeného typografického pravidla TEX poskytuje primitivní příkaz `\discretionary`, který má tuto formu zápisu:

`\discretionary{`⟨*prebreak*⟩`}{`⟨*postbreak*⟩`}{`⟨*nobreak*⟩`}`

Parametr ⟨*nobreak*⟩ určuje, co se má vysázet, jestliže v místě výskytu příkazu `\discretionary` nedojde k řádkovému zlomu. V opačném případě se na konec jednoho řádku vysází parametr ⟨*prebreak*⟩ a na začátek druhého řádku parametr ⟨*postbreak*⟩. Významným omezením je skutečnost, že žádný z těchto tří parametrů nesmí obsahovat výplněk typu glue, tj. pružnou mezeru, jakou je mezislovní mezera nebo třeba mezera vytvořená pomocí příkazu `\hskip`.

Obvykle se můžeme setkat přibližně s takovouto definicí makra `\až`:

```
\def\až{\discretionary{\hbox{ až}}{}{---}}
```

Vidíme, že mezeru, kterou je potřeba vložit před slůvko „až“, se autor makra pokusil zamaskovat tím, že ji uzavřel do `\hbox`u. Tímto způsobem sice úspěšně obešel zákaz výskytu výplňků typu glue v parametrech příkazu `\discretionary`, ale mezera ztratila svoji pružnost a stala se z ní mezera s pevnou šířkou. Pevná mezera před „až“ může zaujmout čtenářovu pozornost, a tak ho při četbě rozptylovat, neboť čtenář zde podvědomě očekává obyčejnou mezeru. Jestliže se řádek stáhne, bude se čtenáři zdát, že je tato mezera příliš široká. Pokud se naopak řádek rozpálí, bude se čtenáři tato mezera jevit jako příliš úzká. Míra rušivosti této mezery bude záviset na míře deformace mezislovních mezer v daném řádku. Ve většině případů asi nebude tento jev nijak tragický a nepatřičné mezery si všimne jen vskutku vnímavý čtenář. Bylo by ovšem lepší, kdyby mezera před „až“ podléhala stejné deformaci jako ostatní mezislovní mezery v daném řádku. Tím by se také nepatrně zvětšil manévrovací prostor TEXu při zalamování odstavce do jednotlivých řádků.

Chceme-li popsaný nedostatek makra `\až` odstranit, musíme si na pomoc přibrat primitivní příkaz `\cleaders`. Tento příkaz dovede proměnit libovolný box ve výplněk typu glue. Přesněji řečeno, vytvoří výplněk typu glue požadované velikosti a na tento výplněk opakovaně vykreslí obraz příslušného boxu. V horizontálním módu můžeme příkaz `\cleaders` zapsat takto:

`\cleaders`⟨*box*⟩`\hskip`⟨*velikost*⟩

Parametr ⟨*velikost*⟩ určuje požadovanou velikost (šířku) výplňku typu glue; kromě základní velikosti lze uvést i hodnoty roztažitelnosti a stažitelnosti, uvozené klíčovými slovy `plus` a `minus`. Parametr ⟨*box*⟩ reprezentuje box, jehož obraz se má opakovaně vykreslit na výplněk typu glue; v rámci tohoto parametru je možno pomocí příkazu `\hbox`, `\vbox` či `\vtop` vytvořit nový box nebo pomocí příkazu `\box` či `\copy` použít již hotový box, který je uložen v registru. Obraz boxu bude vykreslen tolikrát, kolikrát se šířka tohoto boxu vejde do definitivní šířky výplňku typu glue. Jednotlivé kopie obrazu budou umístěny těsně vedle sebe a jako celek budou zarovnány na střed výplňku typu glue.

Horizontální výplněk typu glue vytvořený pomocí `\cleaders` je odstranitelný element, v němž je možno provést řádkový zlom, pokud bezprostředně před ním předchází nějaký neodstranitelný element. Z horizontálního seznamu ho lze odstranit pomocí příkazu `\unskip`; na jeho velikost se lze dotazovat prostřednictvím příkazu `\lastskip`. Od běžného horizontálního výplňku typu glue (tj. od pružné horizontální mezery) se liší jednak tím, že na sobě může nést nějaký obraz, jednak tím, že má výšku a hloubku odpovídající výšce a hloubce boxu uvedeného jako parametr ⟨*box*⟩.

Vyzbrojeni vědomostmi o primitivním příkazu `\cleaders` se můžeme pustit do definování makra `\až`. Makro vysází rozsahovou pomlčku, která se v případě potřeby dokáže proměnit ve slůvko „až“, před nímž bude předcházet pružná mezislovní mezera.

```
\def\až{\leavevmode
   \null
   \nobreak\ \skip0=\lastskip
   \discretionary{až}{}{}%
   \nobreak\hskip-\skip0
   \setbox0=\hbox{---}%
   \cleaders\copy0\hskip\wd0
   \null}
```

Nejdříve se ubezpečíme, že nejsme ve vertikálním módu; v zásadě to dělat nemusíme, neboť smysluplné užití makra `\až` ve vertikálním módu nepřipadá v úvahu. Potom do horizontálního seznamu vložíme prázdný `\hbox` a nezlomitelnou mezislovní mezeru; do registru `\skip0` si uschováme velikost této mezery a do horizontálního seznamu dále vložíme `\discretionary`, které se v případě řádkového zlomu promění ve slůvko „až“, umístěné na samém konci řádku. Pak následuje nezlomitelná záporná mezislovní mezera. Je dobré uvědomit si, že znaménko minus, které jsme zapsali před registr `\skip0`, ovlivní nejen základní velikost mezery, ale také hodnoty roztažitelnosti a stažitelnosti. Potom si do registru `\box0` připravíme `\hbox` s pomlčkou a pomocí `\cleaders` ho proměníme ve výplněk typu glue. Nakonec do horizontálního seznamu přijde prázdný `\hbox`.

Nyní bychom si měli vysvětlit, jak toto makro funguje. Zatím si nevšímejme prázdných `\hbox`ů, které se vkládají do sazby na začátku a na konci makra; k jejich významu se vrátíme za chvilku.

Začněme popisem situace, kdy se materiál z makra `\až` ocitne někde uprostřed řádku, takže v tomto materiálu nedojde k řádkovému zlomu. Nejprve se vysází mezislovní mezera; tato mezera bude vzápětí vyrušena zápornou mezerou, takže ve výsledné sazbě nebude žádná mezera vidět. Ve výsledné sazbě se tedy ukáže pouze pomlčka, která se vykreslí v rámci výplňku typu glue, vytvořeného pomocí `\cleaders`.

Teď si pro změnu představme, že v materiálu z makra `\až` dojde k řádkovému zlomu. V sazbě se nejdříve objeví pružná mezislovní mezera, za ní se z `\discretionary` vloží slůvko „až“, za nímž bude proveden řádkový zlom. Protože za místem řádkového zlomu automaticky mizí všechny odstranitelné elementy, ze sazby zmizí nezlomitelná záporná mezera a pomlčka.

Rozsahová pomlčka se nejčastěji klade mezi dvě číslice, např. *1985—1993;* občas se ale může vyskytnout také mezi dvěma slovy, např. *leden—červen.* Při použití rozsahové pomlčky mezi dvěma slovy by se nám většinou příliš nelíbilo, kdyby na konci řádku došlo k rozdělení jednoho nebo druhého z nich, např. *leden—červen.* Proto makro `\až` vkládá do horizontálního seznamu dva prázdné `\hbox`y. První `\hbox` zabrání dělení slova před pomlčkou; uplatní se zde toto pravidlo: Jestliže hned za slovem po přeskočení znaků, které nejsou písmeny, následuje box, linka, matematický vzorec nebo `\discretionary`, slovo nepodléhá

automatickému dělení. Druhý `\hbox` zabrání dělení slova za pomlčkou, neboť slovo podléhá automatickému dělení, pouze pokud těsně před ním po přeskočení znaků, které nejsou písmeny, předchází výplněk typu glue. Dodejme, že výrazem „znaky, které nejsou písmeny“ se rozumí znaky s nulovým `\lccode`; jedná se tedy především o interpunkční znaménka.

Pokud by uživateli nevadilo případné dělení slov obklopujících rozsahovou pomlčku, může samozřejmě z definice makra `\až` odstranit řídicí sekvence `\null`. Pak by ovšem bylo vhodné vložit na konec definice makra `\až` prázdný příkaz `\relax`, tak aby jím byl řádně ukončen parametr ⟨*velikost*⟩ z konstrukce `\cleaders`.

Zvýraznění prvního řádku v odstavci

Začátky kapitol poskytují typografovi vhodnou příležitost, aby uplatnil své umělecké nadání. V úpravě začátků kapitol hraje významnou roli vstupní odstavec; existuje řada možností jeho typografického zpracování. Pro inspiraci uvádíme na následující straně několik příkladů, jak je možno vstupní odstavec ztvárnit. V tomto textu se budeme podrobněji zabývat zvýrazněním prvního řádku odstavce verzálkami, popřípadě i jiným písmem.

Máme-li za úkol vysázet první řádek odstavce z verzálek, mohli bychom postupovat následujícím způsobem.

Nejdříve si definujeme makro, které pomocí příkazu `\uppercase` převede malá písmena na velká a zároveň dočasně zvětší velikost mezislovní mezery.

```
\def\verzalky#1{{\spaceskip=.5em plus.166666em minus.166666em
   \uppercase{#1}}}
```

Potom si na zkoušku vysázíme celý odstavec z verzálek. Protože nás zajímá pouze první řádek tohoto odstavce a rádi bychom dosáhli jeho optimálního vzhledu, nastavíme pomocí příkazu `\parshape` délku druhého a dalších řádků na největší možnou hodnotu. Tím vlastně potlačíme vliv těchto řádků na rozhodování o místě zalomení prvního řádku. Příslušná část kódu ve zdrojovém souboru by mohla vypadat takto:

```
\noindent
\parshape 2 0pt\hsize 0pt\maxdimen
\verzalky{Jiným vhodným způsobem zvýraznění začátků
...
řádku, je už tvořena malými písmeny.}
```

Po zpracování zdrojového souboru TEXem a zobrazení výsledné sazby získáme přesnou představu o tom, jaká část textu se vejde na první řádek. Podle toho upravíme množství textu zahrnutého do parametru makra `\verzalky`,

Chcete-li při sazbě knihy vyzdobit vstupní stránky jednotlivých kapitol, můžete u prvního odstavce každé kapitoly použít iniciálu, tj. výrazné počáteční písmeno. V češtině považujeme spřežku ch za jediné písmeno, a proto pokud text začíná touto spřežkou, měla by iniciála zahrnovat obě její složky.

První odstavec každé kapitoly může být sázen bez odstavcové zarážky. V takovém případě by serify iniciály měly přesahovat přes levý okraj sazebního obrazce tak, aby levý okraj obrazu iniciály opticky splýval s levým okrajem sazebního obrazce. Pozornost je třeba věnovat také tomu, aby text plynule navazoval na iniciálu.

Další možností je zapustit iniciálu do textu. Výška iniciály může být dva i více řádků. Účaří iniciály se musí krýt s účařím řádku, k němuž je iniciála zapuštěna. Horní okraj iniciály nesmí být níže než horní okraj prvního řádku. Pokud si budete chtít vytvořit makro, které vypočte vhodnou velikost písma pro sazbu iniciály, můžete se inspirovat v TEXbooku naruby na str. 81—82. Zbytek slova, jehož první písmeno je vyjádřeno iniciálou, má plynule navazovat na iniciálu; další řádek či řádky, do nichž iniciála zasahuje, se od iniciály oddělují půlčtverčíkem (tj. mezerou o velikosti 0,5 em). Pokud je iniciálou vyjádřena jednopísmenná předložka či spojka, zbytek prvního řádku se od iniciály odděluje stejnou mezerou jako ostatní řádky, do nichž iniciála zasahuje.

JINÝM VHODNÝM ZPŮSOBEM ZVÝRAZNĚNÍ začátků kapitol je sazba prvního řádku každé kapitoly verzálkami čili velkými písmeny. Jestliže je nutno slovo na konci prvního řádku rozdělit, sází se pochopitelně jeho první část z verzálek a druhá část, která přijde na začátek druhého řádku, je už tvořena malými písmeny.

JEMNĚJŠÍHO VZHLEDU DOCÍLÍTE TÍM, že použijete verzálky menšího písma. Font kombinující verzálky s kapitálkami vídáme často v situacích, kam se příliš nehodí; ani zde bych ho moc nedoporučoval. Takový font je vhodný především tam, kde by měl vyniknout rozdíl mezi malými a velkými písmeny.

VERZÁLKAMI ovšem nemusí být vysázen první řádek celý. Stačí, když z verzálek bude jen první slovo, první slovní spojení či jiná počáteční část prvního řádku. Zatímco v běžném textu by základní velikost mezislovní mezery měla odpovídat třetině čtverčíku, v sazbě z verzálek se doporučuje použít jako základní mezislovní mezeru půlčtverčík; tuto mezeru je pak podle potřeby možno stáhnout až na třetinu čtverčíku anebo rozpálit až na dvě třetiny čtverčíku. Zde je vhodné upozornit také na to, že PostScriptová písma mívají nastavenu příliš úzkou mezislovní mezeru. Proto pokud chcete při použití PostScriptových písem dosáhnout optimální kvality sazby, měli byste si v TEXu pohrát s parametry fontdimen 2, 3, 4 a 7.

HONOSNĚJŠÍM DOJMEM BUDE působit, když spojíte sazbu iniciály se sazbou prvního řádku z verzálek. Opět se nemusí jednat o celý první řádek, nýbrž postačí, když z verzálek bude první slovo či první slovní spojení.

V TOMTO ODSTAVCI je ještě jedna ukázka spojení iniciály s verzálkami. Na závěr dodejme, že zvýraznění začátku prvního odstavce jednotlivých kapitol se může uplatnit jak v krásné literatuře, tak v literatuře naučné a populárně vědné.

v případě potřeby doplníme na konec tohoto parametru spojovník a pomocí příkazu `\break` si vynutíme řádkový zlom. Samozřejmě musíme také odstranit příkaz `\parshape`. Náš kód by nyní mohl vypadat takto:

```
\noindent
\verzalky{Jiným vhodným způsobem zvý-}\break raznění začátků
...
řádku, je už tvořena malými písmeny.
```

Opakovat tento postup na začátku každé kapitoly, to by byla strašná otrava, zvláště kdybychom sázeli rozsáhlejší knihu s mnoha kapitolami. Navíc je toto řešení takříkajíc na jedno použití, neboť při změně některých parametrů sazby je zapotřebí provést tuto činnost znovu. Proto se pokusíme zvýrazňování prvního řádku odstavce zautomatizovat.

Makro, které si pro tento účel připravíme, načte text celého odstavce do parametru. Nejdříve vysází tento text nanečisto způsobem, jaký jsme si popsali výše. První řádek takto vzniklého odstavce bude vzorem, jak by měl vypadat první řádek výsledného odstavce.

Nyní stojíme před otázkou, jak v textu, který makro načetlo do parametru, určíme hranici mezi prvním a druhým řádkem, abychom mohli první řádek vysázet zvýrazněně a zbytek odstavce normálně. Pro tento účel využijeme informaci o šířce vysázeného textu: Změříme přirozenou šířku materiálu ze vzorového prvního řádku. Pak založíme dva zásobníky. Jeden zásobník bude reprezentovat text prvního řádku odstavce a na počátku bude prázdný; druhý zásobník bude představovat text zbytku odstavce a na počátku bude obsahovat text celého odstavce. Z tohoto zásobníku budeme postupně přesouvat jednotlivá slova do prvního zásobníku. Po každém přesunu slova vysázíme v pokusném `\hbox`u obsah prvního zásobníku a šířku tohoto `\hbox`u porovnáme s přirozenou šířkou materiálu ze vzorového prvního řádku. Budou-li se obě šířky shodovat, znamená to, že jsme nalezli kýženou hranici mezi prvním a druhým řádkem. Pakliže bude šířka pokusného `\hbox`u větší než přirozená šířka materiálu ze vzorového prvního řádku, musíme onu hranici hledat uvnitř posledního slova v prvním zásobníku. Proto toto slovo vrátíme do druhého zásobníku a do prvního zásobníku budeme nyní postupně přesouvat jednotlivá písmena. Po každém přesunu písmene vysázíme v pokusném `\hbox`u obsah prvního zásobníku (nesmíme ovšem zapomenout na konec tohoto `\hbox`u přidat spojovník) a šířku tohoto `\hbox`u porovnáme s přirozenou šířkou materiálu ze vzorového prvního řádku. Kýžená hranice mezi prvním a druhým řádkem bude známa v okamžiku, kdy se budou obě šířky shodovat.

V případě, že hranice mezi prvním a druhým řádkem probíhá uvnitř slova, mohla by naše metoda detekovat tuto hranici na nesprávném místě, pokud by kvůli splynutí písmen v ligaturu nebo kvůli záporným implicitním kernům měla skupina písmen, za níž by měla tato hranice probíhat, stejnou šířku jako počáteční část této skupiny. To nás ovšem nemusí příliš znepokojovat, neboť takový

jev je velice nepravděpodobný a navíc dobrý sazeč se nespoléhá na automatické dělení slov, nýbrž pokaždé ve vysázeném dokumentu zkontroluje, zda jsou všechna slova rozdělena správně. Bohužel v dnešní době je dobrých sazečů poskrovnu, a to i mezi TEXisty.

Máme-li v zásobníku `\prvni` uložen text celého prvního řádku a v zásobníku `\zbytek` zbylý text odstavce, stačí už jen obě části textu vysázet náležitým způsobem do jednoho odstavce. Tím činnost našeho makra skončí.

Při přesouvání jednotlivých slov ze zásobníku `\zbytek` do zásobníku `\prvni` se budeme muset vypořádat s tím, že slova mohou být vzájemně oddělena mezerami různého druhu. Pozornost věnujeme jednak obyčejným mezerám, jednak řídicím mezerám. Naproti tomu nás vůbec nebudou zajímat vlnky, neboť v nich není povolen řádkový zlom. Nejdříve ze zásobníku `\zbytek` vyčleníme úsek textu, který sahá od začátku zásobníku až po první řídicí mezeru. Abychom měli jistotu, že v zásobníku je alespoň jedna řídicí mezera, připojíme jednu řídicí mezeru na samý konec zásobníku; ukáže-li se, že je tato řídicí mezera nadbytečná, hned ji zase odstraníme. Z takto získaného úseku textu vyčleníme podúsek, který sahá od začátku úseku až po první obyčejnou mezeru. Abychom měli jistotu, že v úseku textu je alespoň jedna obyčejná mezera, připojíme jednu obyčejnou mezeru na samý konec úseku; ukáže-li se, že je tato mezera nadbytečná, hned ji zase odstraníme. Podúsek textu, který tímto způsobem získáme, je oním slovem, které máme přesunout do zásobníku `\prvni`. Zbylou část vyčleněného úseku textu vrátíme do zásobníku `\zbytek`.

Nyní si ukážeme kód celého makra, jehož činnost jsme si vysvětlili v předchozích odstavcích.

```
% deklarace registrů
\newdimen\sirka % přirozená šířka materiálu z prvního řádku
\newtoks\prvni % zásobník s textem prvního řádku
\newtoks\zbytek % zásobník s textem zbytku odstavce
\newtoks\oldprvni % předchozí obsah zásobníku \prvni
\newtoks\oldzbytek % předchozí obsah zásobníku \zbytek
\newtoks\mezera % mezera mezi textem v \prvni a \zbytek

% hlavní makro, které vysází první řádek odstavce zvýrazněně
% #1 = způsob sazby (zvýraznění) prvního řádku
% #2 = text odstavce
\def\prvniradek#1#2\par{\par
   \def\zvyrazneni{#1}
   % na zkoušku vysázíme celý odstavec zvýrazněně
   \setbox0=\vbox\bgroup
      \zvyrazneni{#2\looseness=0 \overfullrule=0pt
      % aby fungoval \hangindent, započteme jeho velikost do šířky
      % prvního řádku ve zkušebním odstavci, je-li toho zapotřebí
```

```
      \dimen0=\hsize
      \ifnum\hangafter<1
         \advance\dimen0
            by\ifdim\hangindent>0pt -\fi \hangindent
      \fi
      \parshape 2 0pt \dimen0 0pt \maxdimen \endgraf}
      \ifnum\prevgraf<2 % odstavec má pouze jediný řádek
         \egroup {\zvyrazneni{#2\looseness=0 \endgraf}}\par
      \else % odstavec má více řádků
         % rozebereme odstavec, abychom se dostali k prvnímu řádku
         \count255=\prevgraf
         \loop
         \ifnum\count255>0
            \setbox0=\lastbox \unskip \unpenalty
            \advance\count255 by-1
         \repeat
         % určíme přirozenou šířku materiálu z prvního řádku
         \setbox0=\hbox{\unhbox0}
         \global\sirka=\wd0 \egroup
         % nastavíme počáteční obsah zásobníků
         \prvni={}
         \zbytek={#2}
         % začneme přesouvat jednotlivá slova
         \expandafter \presunslovo
      \fi}

% makro, které přesouvá jednotlivá slova ze zásobníku \zbytek
% do zásobníku \prvni a zároveň porovnává šířku sazby textu ze
% zásobníku \prvni s přirozenou šířkou vzorového prvního řádku
\def\presunslovo{
   % zazálohujeme původní obsah zásobníků \prvni a \zbytek
   \oldprvni=\prvni
   \oldzbytek=\zbytek
   % přesuneme jedno slovo ze zásobníku \zbytek do zás. \prvni
   \expandafter\osetriridicimezeru\the\zbytek\ \end
   % v pokusném \hboxu vysázíme obsah zásobníku \prvni
   \setbox0=\hbox{\hskip\leftskip\indent
      \def\noindent{\setbox0=\lastbox}%
      \expandafter\zvyrazneni\expandafter{\the\prvni}%
      \hskip\rightskip}
   \ifdim\wd0=\sirka % nalezli jsme hranici mezi 1. a 2. řádkem
      \let\next=\vysazejodstavecN
```

```
   \else % hranici mezi 1. a 2. řádkem jsme ještě nenalezli
      \ifdim\wd0<\sirka
         \edef\temp{\the\zbytek}
         \ifx\temp\empty % něco je v nepořádku
            \errmessage{Chyba 1}
            \let\next=\vysazejodstavecN
         \else % přesuneme další slovo
            \edef\temp{\prvni={\the\prvni\the\mezera}}\temp
            \let\next=\presunslovo
         \fi
      \else % hranice probíhá uvnitř slova
         % obnovíme předchozí obsah zásobníků
         \prvni=\oldprvni
         \zbytek=\oldzbytek
         % nyní začneme přesouvat jednotlivá písmena
         \let\next=\presunpismeno
   \fi\fi
   \next}

% makro, které z textu oddělí první část ukončenou řídicí mezerou,
% případný zbytek textu vrátí do zásobníku \zbytek a oddělenou
% část předá ke zpracování makru \osetrimezeru
\def\osetriridicimezeru#1\ #2\end{
   \def\temp{#2}
   \ifx\temp\empty % text neobsahuje řídicí mezeru
      \zbytek={}
      \mezera={}
   \else % text obsahuje řídicí mezeru
      \smazridicimezeru#2\end
      \mezera={\ }
   \fi
   \osetrimezeru#1 \end}
\def\smazridicimezeru#1\ \end{\zbytek={#1}}

% makro, které z textu oddělí první část ukončenou mezerou,
% tuto část přidá do zásobníku \prvni a zbytek textu vrátí
% do zásobníku \zbytek
\def\osetrimezeru#1 #2\end{
   \prvni=\expandafter{\the\prvni#1}
   \def\temp{#2}
   \ifx\temp\empty
   \else % text obsahuje mezeru
      \smazmezeru#2\end
```

```
      \mezera={ }
      \def\temp{#1}
      \ifx\temp\empty
         \edef\temp{\the\prvni}
         \ifx\temp\empty
            \mezera={} % počáteční mezera bude ignorována
   \fi\fi\fi}
\def\smazmezeru#1 \end{
   \toks0={#1}
   \edef\temp{\zbytek={\the\toks0 \the\mezera\the\zbytek}}\temp}

% makro, které přesouvá jednotlivá písmena ze zásobníku \zbytek
% do zásobníku \prvni a zároveň porovnává šířku sazby textu ze
% zásobníku \prvni s přirozenou šířkou vzorového prvního řádku
\def\presunpismeno{
   % přesuneme jedno písmeno ze zásobníku \zbytek do zás. \prvni
   \expandafter\sejmipismeno\the\zbytek\end
   % v pokusném \hboxu vysázíme obsah zásobníku \prvni
   \setbox0=\hbox{\hskip\leftskip\indent
      \def\noindent{\setbox0=\lastbox}%
      \expandafter\zvyrazneni\expandafter{\the\prvni-}%
      \hskip\rightskip}
   \ifdim\wd0=\sirka % nalezli jsme hranici mezi 1. a 2. řádkem
      \let\next=\vysazejodstavecR
   \else
      \edef\temp{\the\zbytek}
      \ifx\temp\empty % něco je v nepořádku
         \errmessage{Chyba 2}
         \let\next=\vysazejodstavecR
      \else
         \ifdim\wd0<\sirka % hranici musíme hledat dále
            \let\next=\presunpismeno
         \else % něco je v nepořádku
            \errmessage{Chyba 3}
            \let\next=\vysazejodstavecR
   \fi\fi\fi
   \next}
\def\sejmipismeno#1#2\end{
   \prvni=\expandafter{\the\prvni#1}
   \zbytek={#2}}

% makra, která provedou konečnou sazbu celého odstavce
\def\vysazejodstavecN{\vysazejodstavec{}{\relax}}
```

```
\def\vysazejodstavecR{\vysazejodstavec{-}{\odstranspoj}}
\def\vysazejodstavec#1#2{\setbox0=\hbox{%
      \def\noindent{\setbox0=\lastbox}\indent
      \expandafter\zvyrazneni\expandafter{\the\prvni#1}}
   \noindent\unhbox0\break \expandafter#2\the\zbytek \par}
\def\odstranspoj{\futurelet\temp\xodstranspoj}
\def\xodstranspoj{\ifx-\temp \expandafter\removetoken \fi}
\def\removetoken#1{}
```

Povšimněme si definice `\vysazejodstavec`: Nejprve sestavíme první řádek v `\hbox`u a potom obsah tohoto `\hbox`u vypustíme do horizontálního seznamu v odstavcovém módu. Děláme to proto, abychom na konci prvního řádku neměli `\discretionary`, pokud by první řádek končil spojovníkem. V odstavcovém módu TEX automaticky připojuje prázdné `\discretionary` za každý spojovník, kdežto ve vnitřním horizontálním módu nikoli. Toto `\discretionary` by nám dělalo neplechu při přetečeném prvním řádku, a to i v případě, že by velikost přetečení byla menší než hodnota registru `\hfuzz`, tedy i tehdy, kdy by nám nepatrné přetečení řádku samo o sobě vůbec nevadilo. Při hodnotě registru `\exhyphenpenalty` menší než 10 000 by TEX přetečený první řádek zalomil v tomto `\discretionary` a hned za ním by provedl ještě jeden zlom, vynucený příkazem `\break`, takže by se pod prvním řádkem objevil jeden prázdný řádek a pak by teprve pokračoval další text odstavce.

Příkaz `\odstranspoj` v případě potřeby odstraní přebytečný spojovník ze začátku druhého řádku. Představme si, že v německém nebo anglickém textu na konci prvního řádku je slovo, které obsahuje spojovník (např. *Drucker-Zeugnis*), rozděleno právě v místě spojovníku. Jak vyplývá z kódu našich maker, tento spojovník zůstane na začátku zásobníku `\zbytek` a na konec prvního řádku bude přidán další spojovník. V němčině a angličtině (na rozdíl od češtiny) je zvykem v takových případech psát spojovník jen na konci řádku, takže by nám zde jeden spojovník přebýval; proto musí být pomocí makra `\odstranspoj` odstraněn. V češtině je zapotřebí dělení slov obsahujících spojovník řešit komplexně v celém dokumentu; obvykle postačí mít nastaveno `\exhyphenpenalty=10000`.

Teď už nám zbývá říci si pár slov o užívání našich maker. Před odstavec, jehož první řádek má být zvýrazněn, zapíšeme příkaz `\prvniradek`. Jako první parametr tohoto příkazu uvedeme příkaz (nebo příkazy), který provede požadované zvýraznění. Nemusí se jednat pouze o zvýraznění verzálkami, také je možno nastavit jiné písmo, třeba kurzivu či tučné písmo, anebo použít verzálky menšího písma. Sázíme-li text písmem o velikosti deseti bodů, můžeme první řádek zvýraznit devítibodovými verzálkami. Nejdříve tedy zavedeme devítibodové písmo:

```
\font\ninerm=csr9
```

a potom už můžeme do zdrojového souboru zapsat celý odstavec, uvozený příkazem `\prvniradek`:

```
\prvniradek{\ninerm\verzalky}
\noindent
Jiným vhodným způsobem zvýraznění začátků
...
řádku, je už tvořena malými písmeny.
```

Všimněte si, že příkaz `\noindent` pro zahájení odstavce bez odstavcové zarážky je zapsán těsně na začátku textu až po řídicí sekvenci `\prvniradek`. Pokud chceme upravit délku odstavce pomocí `\looseness`, musíme příkaz `\looseness` zapsat až na konec odstavce. Má-li ovšem odstavec jen jediný řádek, `\looseness` nemá žádný význam, neboť v takovém případě naše makro automaticky nastaví `\looseness` na nulu. Podobně bezvýsledná by byla snaha o zkrácení dvouřádkového odstavce. Jestliže chceme upravit tvar odstavce pomocí `\hangindent` a `\hangafter`, musíme tyto dva příkazy uvést také až na konci odstavce. Úprava tvaru odstavce pomocí `\parshape` není možná.

Hranice mezi prvním a druhým řádkem nesmí být uzavřena ve skupině. Pro sazbu uvozovek je tedy vhodné v celém odstavci používat příkazy `\clqq` a `\crqq` místo obvyklejšího příkazu `\uv`. Pro náročnějšího čtenáře dodejme, že makro `\uv`, jak je definováno v csplainu, by mohlo způsobit neplechu, i kdybychom se pomocí tohoto makra snažili dát do uvozovek část textu, která by celá zůstala na prvním řádku. Toto makro potlačuje případné implicitní kerny mezi uvozovkami a uvozovaným textem, neboť mezi uvozovky a uvozovaný text vstupuje příkaz hlavního procesoru. Tímto způsobem potlačené implicitní kerny a ligatury TEX obnovuje při druhém průchodu řádkového zlomu (obnova ovšem probíhá pouze v těch skupinách znaků, v nichž TEX vyhledává potenciální místa dělení). Ve vzorovém prvním řádku by tedy implicitní kerny mohly být obnoveny, kdežto v textu vysázeném v pokusném `\hbox`u nikoli. V takovém případě by naše makro nejspíše havarovalo s chybou č. 3.

Visící spojovník

V typografii se někdy využívá tzv. visící interpunkce, která spočívá v tom, že interpunkční znaménka vyčnívají z okraje sazby. Jakým způsobem je možno v TEXu udělat visící interpunkci, je dostatečně popsáno v dodatku D Knuthova TEXbooku a v kapitole 6 Olšákova TEXbooku naruby. Zde se zaměříme na jednu zajímavost, kterou ve zmíněných knihách nenaleznete.

Zvláštní problém při tvorbě visící interpunkce představuje spojovník. V originálním TEXbooku i v českém TEXbooku naruby se doporučuje pro visící spojovník použít speciální font, ve kterém by spojovník měl nulovou šířku. Pro tento účel by se nejspíše hodil virtuální font. My si ale ukážeme, že při sazbě visícího spojovníku si TEXista může vystačit s obyčejným fontem, v němž má spojovník svoji přirozenou šířku.

Důležitým předpokladem našeho řešení je, aby použitý font měl alespoň jednu pozici neobsazenu žádným znakem. Tento předpoklad je splněn jak u originálních fontů písma Computer Modern, tak u tzv. CS-fontů s československou variantou tohoto písma. Dále využijeme toho, že TEX přičítá k mezířádkové penaltě hodnotu z registru `\brokenpenalty`, jestliže řádek, který je bezprostředně nad mezířádkovou penaltou, končí rozděleným slovem.

Do registru `\hyphenchar` přiřadíme číslo libovolné pozice, která není obsazena žádným znakem; pro originální fonty písma Computer Modern i pro CS-fonty vyhovuje např. pozice `"90`. Protože tím vlastně dosadíme do funkce spojovacího znaménka znak, který ve skutečnosti neexistuje, nebude za rozdělená slova na konci řádku připojeno vůbec nic. Aby nás TEX nehuboval, že v použitém fontu není na dané pozici přítomen žádný znak, nastavíme `\tracinglostchars` na nulu. Budeme-li mít odstavec vysázen bez spojovníků na konci řádků, podle velikosti mezířádkových penalt zjistíme, které řádky končí rozděleným slovem, a na konec takových řádků připojíme spojovník, který bude uzavřen do `\hbox`u s nulovou šířkou, takže tento spojovník bude vyčuhovat napravo ze sazebního obrazce. Mezi každé dva řádky v odstavci TEX vkládá penaltu o hodnotě `\interlinepenalty`. Mezi prvním a druhým řádkem je tato penalta zvětšena o hodnotu `\clubpenalty` a mezi předposledním a posledním řádkem je zvětšena o hodnotu `\widowpenalty`. Končí-li první ze dvojice řádků rozděleným slovem, je k této penaltě přičtena ještě hodnota `\brokenpenalty`. Abychom tedy z velikosti mezířádkové penalty mohli vyčíst, zda je na konci řádku, který této penaltě předchází, rozdělené slovo či nikoli, budeme potřebovat, aby `\brokenpenalty` nebyla rovna nule. Plain i csplain nastavují `\brokenpenalty=100`.

Pro uživatele vytvoříme příkazy `\vystrkuj` a `\nevystrkuj`. Mezi tyto dva příkazy uživatel napíše libovolný počet odstavců, v nichž bude chtít mít visící spojovníky. Makro `\vystrkuj` zahájí skupinu, nastaví registry `\hyphenchar` a `\tracinglostchars` na hodnoty, o nichž jsme se zmínili výše, dále lokálně předefinuje příkaz `\par` a pak otevře pracovní `\vbox`, označený číslem 0. Pro zajištění správné mezířádkové mezery nad jednotlivými odstavci je zapotřebí přenášet `\prevdepth` mezi vnějším vertikálním módem a pracovním `\vbox`em 0. Po sestavení odstavce v pracovním `\vbox`u 0 bude tento odstavec v cyklu `\loop` postupně odspodu rozebrán, přitom bude na konec řádků s rozděleným slovem připojen visící spojovník. Následně budou jednotlivé části odstavce opět poskládány do pracovního `\vbox`u 1. Po rozebrání a opětném složení celého odstavce bude uzavřen pracovní `\vbox` 0 a obsah `\vbox`u 1 bude vložen do vnějšího vertikálního seznamu; před tento materiál bude ještě vložena mezera o velikosti `\parskip`. Makro `\nevystrkuj`, pokud bude v pracovním `\vbox`u 0 ještě rozpracován nějaký odstavec, si vynutí jeho ukončení. Protože během toho ukončování bude pracovní `\vbox` 0 uzavřen a znovu otevřen, makro `\nevystrkuj` ho pomocí `\egroup` definitivně uzavře; potom uzavře skupinu, která byla otevřena makrem `\vystrkuj`. Tím se registr `\tracinglostchars` vrátí ke své původní

hodnotě a příkaz \par dostane svůj původní význam. Původní hodnota registru \hyphenchar musí být ovšem obnovena ze zálohy, neboť přiřazení do tohoto registru je vždy globální. Nakonec je nastaveno správné \prevdepth.

```
\newcount\spojovnik % pro uschování hodnoty \hyphenchar
\newdimen\hloubka % pro uschování velikosti \prevdepth
\newcount\radek % číslo právě zpracovávaného řádku
\newcount\vdova % číslo předposledního řádku v odstavci

\def\vystrkuj{\par
   \spojovnik=\hyphenchar\font
   \hyphenchar\font="90 % tento znak nesmí být ve fontu přítomen
   \begingroup
   \tracinglostchars=0
   \def\par{\ifhmode\ukonciodstavec\fi}
   \global\hloubka=\prevdepth
   \setbox0=\vbox\bgroup
   \prevdepth=\hloubka}

\def\ukonciodstavec{\endgraf
   \global\hloubka=\prevdepth
   \setbox2=\lastbox
   \global\setbox1=\vbox{\box2}
   \radek=\prevgraf \advance\radek by-1
   \vdova=\radek
   \loop
   \ifnum\radek>0
      \skip2=\lastskip \unskip
      \count2=\lastpenalty \unpenalty
      \count4=\count2
      \advance\count4 by-\interlinepenalty
      \ifnum\radek=\vdova \advance\count4 by-\widowpenalty \fi
      \ifnum\radek=1 \advance\count4 by-\clubpenalty \fi
      \setbox2=\lastbox
      \ifnum\count4=0 \else
         % připojíme visící spojovník
         % a ošetříme mezeru podle \rightskip
         \setbox2=\hbox to\wd2{\unhbox2
            \skip0=\lastskip\unskip \rlap{-}\hskip\skip0} \fi
      \global\setbox1=\vbox{\box2
         \ifnum\count2=0 \else \penalty\count2 \fi
         \vskip\skip2 \unvbox1}
      \advance\radek by-1
   \repeat
```

```
   \skip2=\lastskip
   \global\setbox1=\vbox{\vskip\skip2\unvbox1}
   \egroup % ukončíme pracovní \vbox
   \vskip\parskip \unvbox1 \endgraf
   \setbox0=\vbox\bgroup
   \prevdepth=\hloubka}

\def\nevystrkuj{\par\egroup\endgroup
   \hyphenchar\font=\spojovnik
   \prevdepth=\hloubka}
```

Při výpočtu demerits řádku s rozděleným slovem TEX normálně použije hodnotu z registru `\hyphenpenalty`. V odstavcích s visícím spojovníkem je to trochu jiné — TEX většinou použije hodnotu z registru `\exhyphenpenalty`, někdy však také hodnotu z registru `\hyphenpenalty`. Abychom dokázali rozhodnout, zda TEX v daném případě použije registr `\hyphenpenalty` nebo `\exhyphenpenalty`, musíme vědět, co přesně do sazebního materiálu vkládá algoritmus na automatické vyhledávání potenciálních míst dělení slov. Nuže, většinou na příslušné místo vkládá `\discretionary{\char\hyphenchar\font}{}{}`. Pokud se však potenciální místo dělení objeví v implicitním kernu nebo uvnitř ligatury, musí zmíněný algoritmus takový případ pečlivě ošetřit, aby kern či ligatura zůstala zachována, kdyby v daném místě k řádkovému zlomu zrovna nedošlo; když se např. v anglickém textu vyskytne slovo *difficult,* zmíněný algoritmus ho upraví tímto způsobem: `di\discretionary{f-}{fi}{ffi}\discretionary{-}{}{}cult`. Poněvadž máme při sazbě visících spojovníků nastaven registr `\hyphenchar` na neexistující znak, je příkaz `\discretionary{\char\hyphenchar\font}{}{}` stejný jako příkaz `\discretionary{}{}{}`, takže se při výpočtu demerits použije `\exhyphenpenalty`, neboť parametr ⟨*prebreak*⟩ je prázdný. Je-li však slovo rozděleno v implicitním kernu nebo uvnitř ligatury, použije se `\hyphenpenalty`.

Summary: Some tips and tricks for TEX typesetting

The article presents an original solution of three challenging typographic tasks by means of TEX's macrolanguage. The reader can find here not only a detailed description of the sophisticated macros but also a number of recommendations concerning typography.

1. In Czech typography, the interval dash cannot stand at the end of a line nor can it be moved to the beginning of the following line. If the split between lines is inevitable, the dash is replaced by a word. While writing a macro for the interval dash, it is necessary to overcome a certain disadvantage of the command `\discretionary`, namely that it does not allow to append glue to the current

list, as the word which replaces the dash has to be separated from the text by a space, while the dash is not separated.

2. The setting of the whole first line of a paragraph in capitals (or possibly in a different font) can be applied at the beginning of individual chapters.

3. For hanging hyphenation, the TEXbook recommends to use a special font with zero-width `\hyphenchar`. The solution of this typographic task can, however, be achieved using an ordinary font with real-width `\hyphenchar`.