Organizační úvod

Bude odevzdávací systém, zatím odevzdávat domácí úkoly. (Musíme splnit něco jako 7 domácích úkolů).

1 Úvod

.tex + .tfm (tex font metric = rozměry písmen) ← TEX ← DVI (formát nezávislý na OS) ← DVI moduly = DVI drivery (dvips, xdvi, pdftex (ten navíc potřebuje .tfm a fonty)).

TEXumí primitiva. V tom se ale sázení řídí špatně, tedy existují nadstavby (plainTEX, nadstavba od autora, nad nim jsou postavené conTEXt, opmac a LATEX) a nadstavby mimo, jako AMSTEX, XeTEX, BibTEX.

tex nebo pdftex spouští plainTEXnebo s přepínačem -ini iniTEX, který umí zkompilovat makra do formatu.

Literatura:

- Knuth: The TEXbook: nejdřív je to takový tutoriál, potom TEXuvnitř
- Olšák: TEXbook naruby: v opačném pořadí, česky
- Knuth: TEXthe program (popsaný zdroják TEXu)

1.1 Sazba odstavce

Vstup = horizontální seznam → (proces předělání se spouští příkazem par) zalámaný odstavec

Horizontální seznam obsahuje:

- box (h, d, w): písmenko, slitek (ligatura), hbox / vbox
- linka (h, d, w): hrule, vrule (liší se tím, kde se mohou vyskytnout, tedy hrule nepatří do horizontálního seznamu)
- discretionary break (pevné šířky): nobreak (stav bez rozdělení) pre-break (na konci řádku) post-break (na začátku dalšího), nesmí obsahovat pružné věci
- whatsit (?): přepínač jazyka, …
- vertikální materiál (nemá vliv na horizontální sazbu, při zlomu vypadne do vertikálního seznamu), objeví se například po řádku s `\vadjust{…}`

- lepidlo (= glue) (= pružné výplňky) – věc s fixní šířkou, roztažitelností a stlačitelností (ty se udávají buď přímo v jednotkách jako px apod, nebo v jednom ze 3 nekonečen (také jednotka) fil, fill, filll).
- kern (= pevné výplňky) – věc pouze s fixní šířkou, automaticky vzniká například pro oddálení kulatých písmen / přiblížení plochých…
- penalty (= trest) – číslo, které říká, jak moc chceme / nechceme zlomit
- math on/off – zapíná a vypíná matematiku

Prvních 5 je non-discardable (jsou vidět). Ostatní discardable.

Možná místa zlomu jsou:

- glue (když před sebou má něco non-disc. a není uvnitř math)
- před kernem (když za ním je glue a není uvnitř math)
- math off (následovaný glue)
- v penaltě (ne větší než 10000)
- v discretionary breaku (ten se ale přidává až při druhém průchodu = když se nezalomí bez něho)

V zlomu se z předchozích věcí stane box řádku, u kterého se spočítají rozměry a spočítá se, jak moc je tento zlom špatný (badness), tím se vybere ten nejlepší zlom a vysází se „jeho“ řádek. Následně se zahodí všechny discardable věci a pokračuje se na dalším řádku.

Badness boxu se počítá (pokud součet nekonečných roztažitelností není nula, pak je badness 0):

$$\left\lceil 100 \cdot \left(\frac{\text{deformace}}{\text{pružnost(součet roztažitelností / stlačitelností)}} \right)^3 \right\rceil$$

Roztahuje (stlačuje) se v nejvyšším nenulovém nekonečnu a to v poměru hodnot roztažnosti.

Nestlačuje se o více jak 100% (dá badness ∞).

`\hbadness` říká badness, při které už se vypisuje, že se TEXpokusí (už ho dělá, nevybírá) vysázet box s touto badness.

`\tolerance` (při prvním průchodu `\pretolerance`, která je zpravidla o dost nižší) říká, že větší badness už se zahazuje. Při třetím průchodu už se používá, že roztažnost zvýšíme o `\emergencystretch`

Lámání odstavce funguje tak, že se prohledají zlomy (každý bod zlomu se dá považovat za vrchol grafu, hrany jsou ohodnoceny tzv. demerits (počítá se z badness a dalších věcí)).

Demerits:

$$(l+b)^2 \pm p^2 + extras$$

(l = `\linepenalty` – přidává se vždy (aby se nelámalo všude, kde lze), b = badness, p = penalta na níž se láme resp. `\hyphenpenalty` nebo `\exhyphenpenalty`, extras = `\adjdemerits` (10000, přidává se, když typ řádku není sousední – aby nebyl velmi roztažen, když předchozí byl stlačen) + `\doublehyphendemerits` (10000, přidává se, když vyjdou 2 neprázdné pre-breaky za sebou) + `\finaldemerits` (5000, neláme se na předposledním řádku odstavce))

Typ řádku: 0 (roztažení o více jak 100%), 1 (roztažení o 50-100%), 2 (změna max. o 50%), 3 (stlačení o více jak 50%)

Příklad (Domácí úkol)
Zkuste vymyslet, co naskládat do horizontálního seznamu, aby se mezera dala zlomit, ale zůstala na dalším řádku.

Příště tvar odstavce (left / right)skip, jak vzniká hor. seznam, algoritmus na pakování boxů.

Poznámka (Vzniky horizontálních věcí)
Boxy (znaky, ligatury, `\hbox to 10cm {…}`, `\vbox spread 5mm{…}` (podle v/h se skládá obsah uvnitř boxu, to nastavuje velikost, spread zvětšuje velikost))

Linky (`\vrule width 3pt height 10pt depth 1pt`)

Kerny (`\kern 30pt`)

Glue (`\hskip 10pt plus 2pt minus 1fil`, `hfil`, `hfill`, `hss`)

Penalty (`penalty 100`, `\nobreak`, `break`, `\allowbreak`)

Discr. break (`\discretionary{pre}{post}{no}`)

Etc.

Definice 1.1 (Box packing)

Spočítáme výsledné rozměry a deformace, pokládám věci na baseline, pomocí výšky a hloubky (příkazy `\raise 10pt \hbox{…}` a `\lower` se dají posunovat boxy vůči baseline).

Deformujeme glue.

Určí se neurčité rozměry linek.

Vertikálně se naopak pokládají referenčními body na jednu linku vlevo (zase existuje `\moveleft` a `\moveright`). Výška se pak určuje jako součet výšek a hloubek boxů uvnitř. Hloubka se počítá podle posledního boxu a pak se minimuje na `\boxmaxdepth` u explicitních `\maxdepth` u stránkových zlomů.

Poznámka
Na určování rozměrů se hodí tzv. podpěry (linky nulové šířky s nenulovou výškou nebo hloubkou).

Poznámka
Lze získat rozměry před i po deformaci.

Definice 1.2 (Sázení písmenek)

Font dimen: sklon std.mezera, roztažnost, smrštitelnost, ex, em, extra mezera…

Spacefaktor je na počátku 1000. Pokud sf znaku $\neq 0$: přenastavýme (leda že by sf znaku > 1000 a my jsme < 1000, pak nastavíme na 1000). Mezera: velikost: std.mezera + extramezera (pokud $sf \geq 1000$), roztažnost: $fd3 \cdot sf/1000$, smrštitelnost: $fd4 \cdot 1000/sf$.

Poznámka (Nastavení pro angličitnu)
A-Z: 999, a-z: 1000, .!?: 3000, ,: 1250, (): 0

Existuje i explicitní `\spacefactor 1234`, `\frenchspacing` (nastavuje češtinu), `\nonfrenchspacing` (nastavuje angličtinu), `\spaceskip=5mm` (spaceskip překřičí font dimen aktuálního fontu) a `\xspace=3mm` (použije se, když je moc velký sf).

Poznámka (Některé příkazy)
`\line{…} = \hbox to \hsize{…}`

`\centerline{…} = \line{\hss…\hss}`

`\rlap{…} = \hbox to 0pt{…\hss}` (Box nulové šířky s vyčuhujícím materiálem doleva)

`\llap{…} = \hbox to 0pt{\hss…}` (Box nulové šířky s vyčuhujícím materiálem doprava)

Definice 1.3 (Dělení slov)

Pro každý jazyk má TEXtrie, jak slovo dělit.

Existuje makro `\chyph`, které přepne do češtiny.

Definice 1.4 (Módy fungování v TEXu)

(Přesněji řečeno módy hlavního procesoru)

- Vertikální hlavní (stránkový)
- Vertikální vnitřní (`\vbox`)
- Horizontální odstavcový (zalamování)
- Horizontální vnitřní / restricted (`\vbox`)

- Matematický vnitřní (`$`)
- Matematický display (`$$`)

Na začátku je TEXv hlavním vertikálním módu. Teprve ve chvíli, kdy najde něco, co by mělo být v odstavci (písmenko, `\noindent`, `\indent`, `\leavevmode` (jako písmenko), `\hskip`, `\vrule`), tak se TEXpřesune do odstavcového horizontálního.

Zpět se přesouvá příkazy (`\par` (tj. i 2 odřádkování), vertikálními povely: `\par`, `\vskip`, …, `}` ukončující `\vbox`), což vyvolá odstavcový zlom a vrácení se do hlavního vertikálního.

Obdobně ostatní přechody (pozor, lze přecházet i z Vertikálního vnitřního do odstavcového, naopak nelze přecházet z vertikálních do matematických, tam se automaticky přechází přes horizontální).

Poznámka (Co dostane lámací algoritmus)
Na začátku prázdný *box* šířky `\parindent`. Následuje horizontální materiál odstavce a „ocásek“, ve kterém je `\unskip` (odstranění poslední mezery), `\nobreak`, glue velikosti `\parfillskip = 0 pl` `\break`.

Poznámka (Co lze)
Přenastavit `\parfillskip = \parindent` (pak bude odstavec, když to vyjde, symetrické).

Přenastavit `\parfillspi = 1cm plus 1fil` (např. když máme malou mezeru mezi odstavci a chceme uživatele upozornit na konec odstavce, i když vychází do konce řádku).

Poznámka (Sestavení řádku)
Horizontální materiál řádku se obalí `\leftskip = 0pt` zleva a `\rihtskip = 0pt` zprava a zavře se do `\hbox` velikosti `\hsize`.

Sázení na praporek lze vytvořit tím, že nastavíme `\rightskip = 0pt plus 1 fil`, ale pak se budou řádky snadno lámat (nebudou se rozdělovat slova, budou kratší řádky). Správně na to existuje makro `\raggedright`, které udělá `\rightskip 0pt plus 4em\spaceskip=…\xspacesk` (nastaví mezislovní a písmenné mezery na pevné, aby se neroztahovali podle smršťování a roztahování té mezery na konci).

Centrování `\leftskip = \rightskip = 0,4 plus 2em\parfillskip = 0pt`.

Poznámka (Tvar odstavce)
Vykousnutí se nastavuje `\handindent` (rozměr, o kolik se odsadí) a `\hadgafter` (číslo, kolik řádků se odsadí), když se nastaví záporné hodnoty, vykusují se intuitivně ostatní rohy odstavce. Na konci odstavce se nuluje.

Následuje `\parshape = n p1 w1 … pn wn` (kolik se má odsadit, o kolik které, poslední se opakuje do nekonečna). Také se nuluje.

Když zrovna nejsme ve vertikálním módu, tak se v `\prevgraf` uchovává počet řádků v předchozím odstavci.

Existuje makro `\everypar`, které spustí nastavený kód každý odstavec.

Můžeme si objednat zmenšení / zvětšení počtu řádků `\looseness = n` (- je kratší, pokud nelze vyplnit, bude ignorováno).

Poznámka (Výsledek lámání odstavce: vertikální materiál)

- ∀ řádek jako box + posunutí referenčního bodu (žádné glue).
- Dále z boxů vypadají vertikální věci (`vadjust`, `mark`).
- Penalty mezi řádky (`\interlinepenalty = 0` + `clubpenalty = 150` (po prvním řádku) + `widowpenalty = 150` (před posledním řádkem) + `\brokenpenalty = 100` (po pre-break) + `\displaywidowpenalty` (aby nebyla osamocená display matematika)).
- Ještě se objeví vertikální (zde řádkové) mezery, ale ty probereme zvlášť.

Definice 1.5 (Řádkování)

Algoritmus, aby se pokud možno dodrželo řádkování (ale řádky mohou být různě široké). Řídí se 3 parametry: `\baselineskip` (glue), `lineskiplimit = 0pt` (dimen) a `lineskip = 1pt` (glue).

Vypočítá mezeru jako $skip = bls - d_{horni} - h_{spodni}$. Pokud vyjde $skip < lsl$, nastaví se $skip = ls$. (Při více stránkách není dobré nastavovat pružnost těchto mezer).

vskip, kern, penalty ignorujeme, hrule algoritmus potlačí.

Poznámka (Jak je to doopravdy)
V registru `\prevdepth` = hloubka posledního boxu (`-1000pt`: algoritmus potlačen), linka nastaví právě ten dolní limit.

`\nointerlineskip` je `\prevdepth = -1000pt`. `\offinterlineskip` úplně zastaví tento algoritmus `\baselineskip = -1000pt, \lineskip = 0, \lineskiplimit = \maxdimen`

Poznámka (Usazení 1. řádku na stránce (pouze hlavní vertikální mód))
Snažíme se spočítat glue tak, aby výška mezery + výška 1. řádku vyšla `\topskip`, ale není nikdy záporný.

Rozdíl proti řádkovému: nemáme limit (vždy je 0pt) a uvažujeme linky.

Poznámka (Ještě k předchozímu)
Na začátku odstavce se vloží `\parskip`.

2 registry

Definice 2.1

Registry jsou zabudované (konkrétní počet; pojmenované; spousta nastavení, o kterých jsme mluvili) a uživatelské (0…255 každého typu, často (u dalších „TeXů“) i více).

Typy:

- `\count` – číslo (31 bitů + znaménko)
- `\dimen` – rozměr (30 bitů + znaménko ve sp = 2^{-16} pt[a], tj. 14 celá část, 16 desetinná)
- `\skip` – roztažnost (13+16 bitů)
- `\muskip` – matematický (speciální jednotky)
- toks, box, …

Registry se obnovují po konci grupy na začínající stav.

Použití: lze do nich dosazovat (`\count74=32`[b], `\parskip=10pt` (rovnítka lze vyměnit za mezeru, či vynechat)), lze ho použít jako jednotky, vypsat ho (`\the\count5`) (do pdf), vypsat ho (`\showthe\count5`) (do logu), použít jako pointer (`\count\count5`), automaticky konvertovat dimen → skip nebo skip → dimen → count.

[a] pt = 1in/72,27

[b] Číslo lze napsat číslicemi s desetinnou tečkou, apostrof a osmičková soustava, 2 apostrofy a šestnáctková ve velkých písmenech, obrácený apostrof znak resp lomítko znak, hodnota registru a backslash pojmenovaný znak (pomocí *chardef*ch=kód (*=lomítko) to však TEXužívá spíše uvnitř).

Definice 2.2 (Aritmetika)

`\advanced registr by hodnota` (by lze vynechat nebo napsat BY)
`\multiply` (pouze celými čísly)
`\divide` (pouze celými čísly), zaokrouhluje se k nule

Definice 2.3 (Alokace registrů)

- count 0…9 = číslo stránky
- box 255 = přenos obsahu do output rutiny

- reg. 0…9 = pracovní (krom čísel stránek)
- `\countdef\jmeno=cislo` – nastavuje přezdívku za registr s číslem cislo
- `\newcount\pocitadlo` – (plain) alokuje nějaký registr (interně `\countdef\pocitadlo`)
- `\newinsert\…` – (plain) alokuje vše, co potřebuje na insert, viz dále

Definice 2.4 (Boxový registr)

Obsahuje nic, hbox nebo vbox.

Lze ho nastavit (`\setbox0=\hbox{…}`), přemístit na aktuální místo (`\box0`), vložit na aktuální místo (`\copy0`), přemístit / vložit jejich obsah na aktuální místo (`\unhbox0`, `unvbox`, `unhcopy`, `\unvcopy`), měřit / měnit rozměry (`\wd0`, `\ht0`, `\dp0`)[a], (`\showbox0`), (`\newbox\cs`).

[a] toho využívají plainová makra *phantom{…}, *hphantom{…}, *vphantom{…} (*=lomítko), které vytvoří prázdné boxy velikosti jejich obsahů.

3 Stránkový zlom

Definice 3.1 (Obsah vertikálního seznamu)

- box
- linka
- odkaz na insert (plovoucí obsah)
- mark
- whatsit (třeba zápis do souboru `\write`, `\special` viz dále)
- glue
- kern
- penalty

Prvních 5 je non-discardable.

Definice 3.2 (Stránkový zlom)

Místa zlomu: glue, před nímž je non-disc., kern za glue, penalta < 10000.

Nebyl dostatek pamětí na obtížnější, tedy se postupně přidávají prvky, počítá se cost ta

je na začátku 100000, protože by se obsah moc roztáhl, pak jsou rozumné a někdy dojde na nekonečno, kde algoritmus najde zpětně nejlepší zlom (pamatuje si ho, ze stejných vybere ten poslední = nejplnější) a tam zlomí.

Cost[a]: 1) $b < \infty, p \leq -10000, q < 1000 : c := p$, 2) $b < 10000, p \in (-10000, 10000), q < 10000 : c := b + p + q$, 3) $b = 10000 (underfull)$, ostatní konečné jako v 2) : $c := 100000$, 4) jinak: $c := +\infty$.

Pamatuje si `\pagetotal`, kde si pamatuje, co už má na stránce, `\pagestretch…`, kde si pamatuje počty roztažností, a `\pagegoal`, kde si pamatuje výšku (bez plovoucích tedy `\vsize`).

[a] c = cost, b = badness, p = penalta, q = dodatečná penalta

Příklad
Plain má makro `\raggedbottom`, který nechá vlát dole (pružný konec stránk), vytvořte ho.

Algoritmus na lámání stránky lze vyvolat i explicitně pomocí `\vsplit (číslo boxu s vertikálním m` jež se pokusí vysázet vertikální materiál na stránku výšky rozměr (na konci se poslední hloubka „minimalizuje“ s rozměrem `\splitmaxdepth`), odstraní všechno discardable a na začátek „druhé poloviny“ vloží `\splittopskip`. Nedozvíme se však, jak moc se to povedlo (krom zařvání do logu při přetečení).

Poznámka (Motivace k inzertům)
Chceme sázet plovoucí věci (poznámky pod čarou, figury, …), navíc chceme zajistit, aby se např. poznámky mohly lámat (např při omezení zabraného místa poznámkami, nebo když značka (odkaz na poznámku) vyjde na úplný konec řádky). Navíc se musí zajistit třeba vytvoření místa pro čáru nad poznámkami.

Definice 3.3 (Inzerty)

`\newinsert\footins` zarezervuje (vše s jedním číslem `N` a aliasem `\footins`) box `\box N` (Tam se skladuje vertikální materiál k tomuto insertu), `\count N` (nějaké promile udávající kolik místa zabere vertikální materiál (např. pro tři sloupečky by to bylo 333)), `\dimen N` (maximální zabraná výška tímto insertem), `\skip N` (místo před prvním objektem daného typu na stránce).

Insert vložíme `\insert N{vertikální materiál}`, kterýžto vloží značku do vertikálního módu (v horizontálním pomocí `\vadjust`), pozor značka nevypadne z boxu.

Inserty se řadí ke třídám podle čísla, kterým se rezervuje.

Definice 3.4 (Algoritmus)

Na vstupu je insert třídy `N` (odkaz na).

1. Jestliže zatím nebyl žádný insert této třídy, snížíme `\pagegoal` o (výška + hloubka

(toho co tam zůstalo z minule)) `\count N`/1000 + velikost `\skip N` (další rozměry `skip N` se přidají do `\pagestretch` a spol.).

2. Jestliže byl předchozí insert třídy `N` rozdělen (skončíme): `\insertpenalties += \floatingpenalt`

3. Vejde se na stránku vcelku? (a) výška + hloubka insertů třídy `N` po vložení < `\dimen N`. (b) `x` = (výška + hloubka) `\count N`/1000 ≤ 0 nebo `\pagetotal + \pagedepth + x -` ≤ `pagegoal`. Pokud ano, `\pagegoal -= x` a materiál se přesune do aktuálního obsahu insertů třídy N.

4. Pokud se nevešlo: spočteme maximální výšku `h`, která neporuší (a) ani (b) bez `\pageshrink`, a zavolá se `\vsplit to h` (vršek se připraví na přidání na stránku a `\pagegoal` se sníží o jeho výšku + hloubku, spodek se pošle zpět do tzv. přípravné oblasti, penalta ze zlom -> přičteme k `\insertpenalties`)

4 Output rutina

Lze ji předefinovat `\output={…}` (tj. přiřazení do registru `\tokens`).

Definice 4.1

Vstup = box 255 (hlavní obsah strany), box N (inserty, které se vešly), značky(TODO), `\outputpenalty` (penalta, na které se zlomilo, jinak 10000).

Výstup `\shipout{box}` (vysypání boxu do výstupního souboru (dvi / pdf)), vertikální materiál (vrací se zpět do přípravné oblasti).

Pozor
Output rutina běží uvnitř implicitních složených závorek (ve vlastní grupě, lze obejít nastavováním věcí přes `\global`).

Nepředvídatelný stav TEXu (nemůžeme vědět, jak jsou nastavené parametry, které v dokumentu měníme).

Kde je následný `\shipout` usazen nastavují parametry `\hoffset` a `\voffset`, které jsou zřejmé, až na to, že jejich měření začíná palec od okraje stránky. PdfTeX má navíc (`\pdfvorigin` a `\pdfhorigin`, kterými lze změnit ten palec a `\pdfpageheight` a `\pdfpagewidth`).

(Přípravná oblast obsahuje zbytky rozlomených insertů, vertikální materiál z output rutiny, element, na kterém se lámalo (resp. penalta 10000, pokud se lámalo na penaltě), materiál za místem zlomu, zbytek předchozí přípravné oblasti.)

Definice 4.2 (Plain output rutina)

`\headline={…}` a `\footline={…}`, kde si můžeme nastavit obsah hlavičky a patičky, sází se s `\line` roztažený na `\hsize`.

Čísla stránek: `\folio` (záporné `\pageno` způsobí (ve `\folio`) vysázení římského čísla a snížení `\pageno` o jedna).

`\footnote{odkaz}{poznamka}`, `\topinsert` (vloží materiál na začátek některé z dalších stránek) + `\midinsert` (spočítá, jestli se to ještě vejde, když ano, vysází na aktuální místo, když ne, chová se jako `\topinsert`) + `\pageinsert` (podobné jako `\topinsert`, jen se to roztáhne na celou stranu) = (ukončuje se `\endinsert`), `\raggedbottom` (udělá pružný `\topskip` a nastaví output rutinu na to, aby ho přesunula dolu).

`\eject` (přechod do vertikálního módu a `\break`, tedy ukončí stránku (pozor stránka se roztáhne na celou výšku, tedy je potřeba buď `\raggedbottom` nebo přidat `\vfill` před)), `\supereject` (ukončí stránku a navíc vysype (klidně i na další stránky) všechny inserty) a `\bye` (= `\vfill\supereject\end`).

Definice 4.3 (Značky)

`\mark{…}` udělá to, že vloží tzv. značku (nemá žádný vizuální význam). Output rutina se může ptát na `\firstmark` (první značka na stránce), `\botmark` (poslední značka na stránce) a `\topmark` (vrací poslední značku předchozí stránky). Pokud na stránce nejsou značky, zůstávají značky z předchozí strany.

Lze je použít například na slovník. (Většinou se používá poslední heslo této a poslední heslo minulé strany (protože přetéká na aktuální), proto top + bot).

5 Rozebírání TEXu

Input procesor - řádky -> token procesor - tokeny -> expand procesor - tokeny -> hlavní procesor (celé to řídí hlavní procesor, může expand procesoru říct, dej mi následující token neexpandovaný, nebo naopak expand procesoru vrátit poslední token. Zároveň přenastavuje parametry (protože přiřazuje do registrů), kterými se řídí ostatní procesory).

Definice 5.1 (Token)

2 typy: dvojice (znak, kategorie) a řídící sekvence.

Definice 5.2 (Input procesor)

Dostane pokyn přečti řádek a načte 1 řádek do řádkového bufferu. Převede kódování znaků do ASCII (rozšířeného o 1 bit). Následně odstraní konec řádku a mezery před ním (pozor, ne tabulátory apod., pouze mezery, pravděpodobně přežitek z konstatně dlouhých řádků na děrných štítcích) a na konec se přidá (pokud je mezi 0 a 255) `\endlinechar` defaultně nastavený na 13.

Definice 5.3 (Kategorie)

0-15:

0 = \, 1 = {, 2 = }, 3 = $, 4 = &, 5 = newline, 6 = #, 7 ^, 8 _, 9 = ignored, 10 = space, 11 = písmeno, 12 = ostatní, 13 = aktivní (~), 14 = %, 15 = invalid.

0, 5, 9, (10), 14, 15 „nevylezou“ z token procesoru, 10 vyjde jako ASCII mezera (ať to bylo cokoliv).

\catcode64=14 změnilo zavináč na vlastnost komentáře (procento se nezměnilo) (to je to samé jako \catcode`@ nebo \catcode`\@). Viz plain makro \makeatletter a \makeatother nebo to, že čísla jsou zapisována římsky (nesjsou písmena).

Definice 5.4 (Token procesor)

3 režimy: N (newline), M (middle of line), S (shipping spaces)

Kategorie 9 se vždy ignoruje, kategorie 15 se ignoruje a seřve tě.

N = čte, dokud je to kategorie 10. Jakmile přečte něco jiného přemístí se na M. Když přijde náhodou newline, tak pošle dál token \par.

M = čte, doku nenačte kategorii 10. Jakmile se narazí na mezeru, přemístí se na S. Když přijde newline, pošle mezeru a přesune se na N. (Tedy od komentáře se liší jen mezerou.)

S = čte, dokud je to kategorie 10. Jakmile přečte něco jiného přemístí se zpět na M. Když přijde newline, nic se nestane a přemístí se na N.

Při potkání (kategoricky) ^^xy $x, y \in \{0-9, a-f\}$ vyrobí znak s tímto kódem a znovu se podívá na kategorie, pokud potká ^^z $z \in ASCII$, překlopí 6. bit (např. ^^M…13…CR, …).

Pokud narazí (kategoricky) na \, pokud následuje newline, tak vznikne token s prázdným názvem (a asi přejde do N), pokud následuje jiné nepísmeno, tak pošle token s jedním znakem (po mezeře přejde do S, po zbytku do M), pokud následují písmena, tak dočte do konce písmenek a přejde do (S).

Po načtení % zahodí zbytek řádku a přesune se do stavu N.

Například
\def\ahoj{Ahooóój!} (token \ahoj se nahradí Ahooóój!).

\def\cara{\the\count0} (se expanduje dvakrát).

\def\b#{{\bf #1}} (se při zavolání expanduje na {\bf …} tedy na 3+? tokenů).

\def\claim#1.{{\bf #1.}}.

Definice 5.5 (Expand procesor)

U každého makra si pamatujeme seznam parametrů (separátory (jiné znaky) + čísla parametrů (předcházené `#`) v pořadí (1, 2, …, 9)) a seznam náhrady (pravá strana definice = tokeny + placeholdery za parametry `#`, musí být správně uzávorkováno kategoriemi).

Expanze makra: ověříme separátory (před `#1`), další parametry můžou být separované (v definici) parametr je pak do prvního výskytu separátoru na správné úrovni uzávorkování (pokud je správně uzávorkovaný, tak se odeberou krajní závorky (`{}{} -> {}{}` ale `{abc} -> abc`)), nebo neseparované, pak se smažou mezery a vezme se nejkratší neprázdný správně uzávorkovaný text s odebranými vnějšími závorkami (nebo jako samostatný znak, pokud to byl jen 1 znak) jako parametr.

Výjimka, `#{` značí, že `{` je separátor, ale zůstane i po použití makra (`\def\a#1#{<#1>}` se po zavolání `\a123{456}` expanduje na `<123>{456}` místo `<1>23{456}`).

Při definici pomocí `\def\xyz…{tělo}` se neexpanduje nic, při zavolání makra se volání s parametry nahradí tělem s doplněnými parametry, expand procesor se „vrátí“ (vytvoří nový input kanál, který se po projití zavře) a projde „znovu“ tělo.

`\def` je lokální, tedy se občas musí použít `\global\def` nebo zkráceně `\gdef`. Při definici se může expandovat i tělo příkazem `\edef`, např. `\edef\x{\the\pageno}` uloží do `\x` číslice „aktuální“ (pozor na vracení se při zlomu) stránky. Dalším příkladem je `\def\seznam{} \def\pridej#1{\edef\seznam{\seznam #1}}` (ne moc dobrá definice, nevýhody jsou kvadratická složitost přidání a expandující se vložené prvky (lze opravit `\noexpand`, ale expanduje se příště)). Globální `\edef` je `\xdef`.

Existuje ochrana, že v parametrech běžně definovaných maker nemůže být `\par` (pro lepší hlášení chyb), ta se dá vypnout `\long\def…`. Naopak můžeme o makru říct, že nelze volat uvnitř skupiny, a to makrem `\outer\def…` (lze odstranit přenastavením `\outer` na `\relax` před kompilací).

TEXsi pamatuje definice ve dvou tabulkách: table of meanings (řídící sekvence (cs) -> primitiva, registry, makra, znak, token typu dvojice) a definice maker. Makro `\let cs [=] token` přiřazuje řídící sekvenci meaning daného tokenu. Meaning lze vypsat příkazem `\meaning`.

`\noexpand token` zabrání (jednou) expanzi tokenu. `\expandafter token1 token2` expanduje token2 a token1 „vrátí“ před něj a pokračuje expanzí od token1. Pomocí makra `\number registr` (podobně jako `the`) lze dostat číslice čísla v registru. Dále existují `\romannumeral, \string^^A = ^_12 ^_12 A_12` , pro velká římská čísla lze použít makro (zpracovávající se v hlavním procesoru) `\uppercase{…}` (pozor, vnitřek neexpanduje, posílá ho zpět expand procesoru, takže to chce `[\expandafter]\uppercase\expandafter{\romannum` první nemusí být, protože `\uppercase` expanduje, aby „našel“ parametry), podobně `\lowercase`, `\jobname` vypíše název souboru, ze kterého TEXaktuálně běží? `\csname a_12 b_12 c_12\endcsname` vyrobí krabičku označenou ASCII hodnotami `abc`. Takováto krabička se netestuje při expanzi (třeba na existenci, nebo právě jestli smí obsahovat `\par`, nebo zda není `outer`).

Příklad
`{\catcode`@=\active\expandafter}\expandafter\def\noexpand @ \{tělo @\}` definuje @ jako aktivní znak, pak mu nadefinuje meaning, pak zavře skupinu a dodefinuje meaning zase se správnou kategorií.

Poznámka (Debug)
Existuje makro `\showbox N`, které do logu vypíše registr `\box N`. Modifikovat (jak moc toho bude vypisovat) se dá pomocí hodnot „countů“ `\showboxbreath` a `\showboxdepth`.

Definice 5.6 (Podmínky)

Podmínky se vyhodnocují v expand procesoru.

`\if…, YES[\else NO]\fi`

`\ifnmum \count 0 < 100`, `\ifdim`, `\ifhmode`

Poznámka
I ve větvi, která se neexpanduje se počítá správné uzávorkování `\if \fi` (dívá se do tabulky významů, nekontroluje text, neexpanduje).

Definice 5.7 (ifcase)

`\ifcase číslo větev 0 \or větev 1 \or větev 2 \or … \or větev n \else … \fi` expanduje tolikátou větev, jaké jsme mu dali číslo (zbytek zahazuje)

Definice 5.8 (Příklady podmínek)

`\iftrue` – vždy pravdivý, `\iffalse` – vždy nepravdivý, `\if Token1 Token2` (expanduje) – porovnává znaky (krabiky porovnává jako znak 256 + kategorie 16), `\ifcat Tok1 Tok2` (expanduje) – porovnává kategorie, `\ifx Tok1 Tok2` (neexpanduje) – porovnává význam (makra porovnává jako seznamy, porovnává znak i kategorii, krabičky bez významu jsou shodné), `ifnum Num1 Rel Num2` – porovná celá čísla (relace jsou pouze 3: =<>, zbytek se dělá `\else`), podobně `\ifdim` pro rozměry, `\ifhbox N`, `\ifvbox N`, `\ifvoid N` – stav boxového registru, `\ifhmode/vmode/mmode` – v jakém jsme módu hlavního procesoru, `\ifinner` – ptá se na vnitřní mód hlavního procesoru, `\ifodd Num` – lichost čísla, `\ifeof Kanál` – ptá se na EOF na daném kanálu.

Pozor
Definování svého if většinou strašně vybuchne.

Poznámka
Podmínka se správně definuje pomocí `\newif\ifabc` vyrobí 3 makra `\ifwide`, které se chová, jak bychom očekávali (má význam podmínky), `\abctrue/\abcfalse`, což jsou přepínače, které dané if nastaví na `\iftrue/\iffalse` (podmínky se pak vytvoří tak, že definujeme testovací makro, které pak spustí `\abctrue/\abcfalse`).

Definice 5.9 (Smyčky)

Všechny smyčky v TEXu jsou vytvářeny jako tail-rekurze (viz dekorátor v Kotlinu).

V Plainu `\loop…\if…\repeat`, kde `\iffalse` provede klasický break (`\repeat` má význam `\fi`). Při vnoření cyklů to chce vnitřní cyklus uzavřít do group.

Poznámka
Pokud potřebujeme vyrobit nepárové uzávorkování, tak máme příkazy `\begingroup` a `\endgroup` na přenastavování „proměnných“. Na druhý význam uzavření parametrů lze využít `\bgroup` a `\egroup`, které mají významy závorek přiřazený pomocí `\let`.

Definice 5.10 (Token-listové registry)

`\toks0…255`, `\newtoks`, jako ostatní registry.

`\newtoks\t` → `\t[=]{token…}` se bere bez expanze, expanze proběhne `\the\t` (pozor, uvnitř `\edef` se obsah neexpanduje). Toho lze využít pro definici seznamů.

Příklady (interní = možná mají záporné indexy, ale to je jedno ;)): `\output`, `\everypar` – provede se každý odstavec (po vložení `\indent` nebo `\noindent`) a jeho obdoby `\everyhbox/vbox/math` `\aftergroup token` – spustí se po ukončení aktuální group (tyto tokeny se akumulují), `\afterassignment token` – spustí se při dalším přiřazení (tyto tokeny se ne!akumulují, hodí se na parsování dimenzí), `\futurelet\cs Tok1 Tok2` – odpovídá `\let\cs=Tok2`, ale nechává Tok1 a Tok2 na místě (lze použít na nepovinné parametry).

Definice 5.11 (Rozebrání krabičky)

V makru `\newif` se musí rozebrat krabička. To se udělá pomocí `\string` (pozor, vyrábí vše s kategorií 12) a `\escapechar=-1` (\ s hodnotou mimo 0…255 se přeskočí).

Existuje špinavý trik, jak vyrobit libovolný znak s libovolnou kategorií: přenastavíme tabulku `\uppercase` (protože nechává kategorii, mění jen znak) pomocí `\uccode`znak=`znak`.

6 Tabulky

Definice 6.1 (Tabulka)

Tabulka se vyrábí `\halign[to/spread <dimen>]{šablonka\cr řádek \cr řádek \cr … \cr}`. Jednotlivé sloupce (včetně šablony) se oddělují `&`. (Existuje i `\valign`, které vyrobí tabulku s prohozenými řádky a sloupci). Pozor to/spread funguje pouze na glue (`\tabskip`) mezi buňkami, ne na sloupce samotné.

Šablona musí vypadat `a1#b1&a2#b2&…\cr` (ai a bi jsou sekvence tokenů, neexpandují se až na výjimky: makro `\span`, které expanduje následující makro, a `\tabskip`, kde se najde přiřazení do registru a použije se na následující mezisloupcové mezery, po `\cr` ztrácí význam). Periodickou šablonu lze vyrobit `X&…&Y&&Z&U\cr`, kde X až Y je předperioda a Z až U je perioda. Bez předperiody: `&Z&…&U`.

Vysází se tak, že se nejdřív vygenerují všechny řádky, pak se podívá na jednotlivé sloupce a vezme maximum z jejich přirozených velikostí. Při expanzi řádku se vždy nakoukne na první token datové položky a pokud není speciální (`&/\cr`) vloží se konkrétní `a`. Následně se vloží datová položka a konkrétní `b`. Kompletuje se to jako vnitřní horizontální mód, ale nejdřív se jen spočítá přirozená velikost. Po zjištění šířek se každá buňka zabalí do `\hbox` šířky sloupce a pak se zkompletuje celý řádek jako `\hbox` obsahující `\hboxy` a `\tabskipy`.

`…\cr\noalign{vertikální materiál}` vysází za aktuálním řádkem vertikální materiál (případná `\hrule` se vysází na šířku tabulky a ne zrcadla). `\omit` zase nahradí šablonu v aktuální buňce pouhým `#` (vysází se pouze obsah bez daného `a` a `b`). `\span` k této buňce přilepí následující sloupec, většinou se ještě obaluje: `\omit\span\omit` (plain má makro`\multispan X`, které slepí (i s `\omity`) X sloupců k sobě), na toto slepení se prvně vůbec nebere ohled při výpočtu šířky sloupců, pokud se to nepovede bez ohledu, tak se nějak „hloupě“ (většinou ne tak, jak bychom chtěli) roztáhnou sloupce, aby to vyšlo, takže to chce vložit do tabulky podpěry, aby se tam sloučené buňky opravdu vešli.

Ohraničení se všechny dělají linkami.

Definice 6.2 (Leaders)

`\leaders <box/rule> <rozměr>` (máme předdefinováno `\hrulefill`, `\vrulefill`, `\dotfill`), leaders má ve skutečnosti 3 varianty: `\leaders` (zaokrouhlí materiál, aby byl násobkem velikostně boxu, který má leaders vysázet), `\cleaders` (chybu (pokud není místo celočíselný násobek boxu) dá půl na začátek, půl na konec), `\xleaders` (rovnoměrně rozloží chybu do mezer mezi boxy).

7 Práce se soubory

Definice 7.1 (Streamy)

Zápisové streamy 0…15. Zapisuje se `\newwrite\soubor` následovaný `\openout\soubor=jméno`, následně lze používat `\write\soubor{tokeny}` a ukončuje se `\closeout\soubor`. Tyto příkazy se neexpandují, pouze se ukládají a až při `\shipout` a tam se expandují (včetně ukládaných tokenů). Pokud chceme tyto příkazy vykonat okamžitě, připíšeme před ně

`\immediate`. Zápis do zavřeného streamu píše do logu a pokud je číslo ≥ 0 tak se dokonce vypisuje i na output, tedy speciálně `\write16` se používá pro zápis do logu a terminálu.

Čtecí streamy 0…15 se používají podobně (jen se vyhodnocují okamžitě) `\newread\soubor`, `\openin\soubor=jméno`, `\read\soubor to \xyz` (spíše se nepoužívá), `\ifeof\soubor` (pomocí toho se zjišťuje, zda dokument existuje), `\closein\soubor`.

Definice 7.2 (Special)

S tím souvisí `\special`, který je také takzvané whatsit (nejsou součátstí TeXu, ale je to jen naše dohoda s konvertorem, že je použije). Jelikož TEXneumí barvičky, obrázky a spol, tak se takové věci dají vložit `\special`. Dnes jsou ale spíše zastaralé, protože se dnes spíše používají přímo pdftexy, které se o to starají už sami.

8 Matematický mód

Definice 8.1 (abovewithdelims)

`\abovewithdelims()xpt` vysází něco jako zlomek se zlomkovou čarou tloušťky x a s delimitry (závorkami) „(“„)“, které nemusí být prázdné, mohou být dokonce „žádné“ (pouze mezera): „.“ (tam se použije mezera `\nulldelimetter`).

Speciální případy jsou `citatel \over jmenovatel`, `n \choose 2`, `n\atop k`.

Automaticky se vycentrovává pomocí `\hfill`, tedy lze ho přebít `\hfilll`.

Definice 8.2 (Delimeters)

Delimiters se dají upravit pomocí `\left(…\right)`, což si změří obsah a podle toho natáhne závorky (těch existuje několik typů).

Existují i explicitní varianty (v plainu): `\big( \Big( \bigg( \Bigg(\`, které jsou zkonstruované pomocí podpěr. Respektive `\bcosil\bcosir\bcosim` kvůli mezerám vlevo, vpravo a ve středu.

Definice 8.3 (Odmocnina)

Odmocnina (`\sqrt{…}`) je speciální konstrukce, n-tou odmocninu (`\sqrt[n]{…}`) už řeší plain.

Definice 8.4 (Akcenty)

`\tilde…` (resp. `\widetilde{……}`), `\overline…`, `\underline…`, `\overbrace…`, `\underbrace…` (za to se dají napsat meze, tedy `\underbrace{x+(x+1)+\ldots}_\hbox{$k$ členů}$`)

Definice 8.5 (Styly)

4 (resp. 8): D = display (`$$…$$`), T = text (`$…$`), S = script (index, exponent, čitatel, jmenovatel), SS = scriptscript (vnořené S). `\atop` způsobuje (nahoře / dole) D− >T/T', T− >S/S', S− >SS/SS', SS− >SS/SS'. Indexy D− >S, T− >S, S− >SS, SS− >SS.

Každý ze 4 stylů existuje ještě čárkovaná varianta, která říká: „nade mnou ještě něco je“ (tedy dolní index přesouvá do čárkovaného a z čárkovaného se jde do čárkovaného).

Lze je explicitně přepínat pomocí `\displaystyle` apod. Naopak v makru mohu použít `\mathchoice{D}{T}{S}{SS}`, které definuje, jak se má co vysázet (jednodušeji to nejde, jelikož styl se propaguje doprava i doleva). Existuje i plainové makro `\mathpalette\t#1`, které umožní propagovat styl přes nematematický mód.

Definice 8.6 (Matematický seznam)

Obsahuje:

- atomy
- dočasné značky (např. `\left`)
- glue / math. glue (jednotky mu) / nonscript (glue, jež se vloží jen mimo S a SS).
- kern / math. kern (jednotky mu)
- změny stylu
- mathchoice
- horizontální materiál: linky / penalty / discretionary (nobreak musí být prázdný a parametry nejsou matematické listy, ale horizontální listy)
- vertikální materiál: mark / insert / vadjust
- whatsit
- zlomek (hned po doparsování se z něj stává atom)

Definice 8.7 (Atomy)

0 Ord (ordinaly)

1 Op (velký operátor)

2 Bin (binární operátor)

3 Rel (relace, lze za ním zlomit)

4 Open (otevírací delimetter)

5 Close (zavírací delimetter)

6 Punct (interpunkce, nelze za ní zlomit)

7 Inner (mezi `\left` a `\right`)

8 Over (`\overline`)

9 Under (`\underline`)

10 Acc (akcenty)

11 Rad (`\sqrt`)

12 VCent (`\vcenter{…}`)

Prvních 8 lze vyrobit z jiných atomů zavoláním `\mathrel` apod.

Znaky v matematice mají krom `\catcode` ještě `\mathcode`znak="TFXY` TFXY je hexadecimální (T = typ atomu 0…7, F = typ fontu, XY = pozice znaku ve fontu). Pokud kód TFXY je 8000, tak se v matematice (a pouze tam) chová jako aktivní (makro se musí definovat tak, že se přenastaví catcode a pak se vrátí). Navíc T=7 způsobí, že se použije rodina podle `\fam`$\in \{0, \ldots, 15\}$, nebo F, pokud není v intervalu a vygeneruje se atom Ord (toho využívá napr. `\bf`).

Znaky z daným kódem lze explicitně (bez žádného znaku) dají vygenerovat jako `\mathchar"TFXY`. Nebo si můžeme vytvořit pojmenovaný znak `\mathchardef\jmeno="TFXY`.

S delimetry je vše jinak, `\delcode'(=F_1X_1Y_1F_2X_2Y_2` (1 = základní velikost a 2 = první větší). Explicitní je `\delimiter"TF_1X_1Y_1F_2X_2Y_2`, kde T je typ (open / close / rel).

Stejně tak radical jde vyrobit `\radical"F_1X_1Y_1F_2X_2Y_2`.

Definice 8.8 (Mezery)

Běžná se ignoruje, s lomítkem podle fontu, `\hskip 1em` funguje normálně, `\hskip 1mu`, kde zpravidla $1mu = \frac{1}{18}em$ fontu pro daný styl.

Navíc existují rozměry `\thinmuskip` (`\,` nebo záporná `\!`), `\medmuskip` (`\>`) a `\thickmuskip` (`\;`). Např. `\int x^2 \,{\rm d}x` nebo `1\,{\rm km}` nebo `\int\!\int`.

Poznámka (Lámání T modu)
Vkládají se automatické penalty `\binoppenalty` a `\relpenalty`. Jinak se láme na explicitních penaltách a discr.

Poznámka (Úplně mimo)
umožňuje vložit # do expanze makra.

Definice 8.9 (Mezery mezi atomy)

V TeXbooku naruby máme tabulku. Velké mezery mizí (a malá se zmenšuje, protože je v mu) v S a SS.

Navíc Bin se ve skoro všech případech mění na Ord.

Definice 8.10 (Mathsurround)

Dimen, který se vkládá před a po matematickém módu (NEPOUŽÍVAT, rozbíjí to neplainová (ten si to automaticky přenastavuje) makra). Změnou v matematice se změní jen pro aktuální matematiku.

Definice 8.11 (Displayed Math)

`$$formule[\[l]eqno formule v textové matematice]$$` Formule se centruje na šířku zrcadla - z obou stran 1em a `\[l]eqno` (`\[l]eqno` = číslo rovnice sázené v[levo]pravo). Nevejde-li se: 1. smršťujeme formuli, 2. centrujeme do zbylého místa (= zahodí se místo pro nesázené eqno (buď levé nebo pravé, podle toho, jestli máme pravé, nebo levé)).

`\left/rightskip` se neuplatňuje, `\hang/parshape` ano. Formule se počítá za 3 řádky odstavce (mezera, formule, mezera). Při spuštění (první `$$`) se nastaví, v průběhu ho lze tedy měnit, při ukončení (druhých `$$`) se použije.

Před se vloží `\predisplaypenalty + abovedisplay[short]skip` a po se vloží `\postdisplaypenalt` (short se použije, pokud předchozí řádek „končí“ (vizuálně + 2em) dřív než začíná matematika, konec odstavce = `\predisplaysize`, eqno se ignoruje, když je šířky 0, nebo pokud eqno začíná `\rlap`, čímž se vysází číslo místo mezery za displaymath). Následuje odstavec bez `\indent`.

Definice 8.12

`$$ \halign{…} $$` se vysází (hlavně odsadí) jako display math, do odstavce se počítá zase jako 3 řádky.

Lze dokonce i vycentrovat: `$$ \vbox{\halign{…}} $$`, pak se ale neláme stránka uvnitř (bez `\vbox` ano).

Definice 8.13

`\eqalign{…&…\cr…&…\cr…}` sází rovnice se zarovnanou (1, jen 2 sloupce) věcí pod sebe. Sází se jako `\vcenter` = nedělitelné, lze použít „kdekoliv“.

`\[l]eqalignno{…&…&n_1\cr…}` sází podobně jako výše, ale i s číslem. Dá se v něm lámat.

9 Fonty

Definice 9.1 (Základní font TeXu)

cmr10 [at 12pt / scaled 1100] = (cm = computer modern) + (r = roman, jiné možnosti (ti = text italic), (mi = math italic), (b = bold), (bx = bold extended, `\bf`), (tt = typewriter), (sl = slanted)) + (10 = design size 10pt = jak velká byla destička s daným písmenem (konvence tiskařů)) + (at 12pt = velikosti 12pt, pozor nerovná se cmr12, jiné velikosti (cmr12) písmen vypadají jinak) + (scaled 1100 = naškáluj na 1100 promile).

Nastavuje se `\font\cs = cmr10 [at 12pt / scaled 1100]` a pak se dá použít (např. mimo matematiku pouhé `\cs` v matematice `\textfont family = \cs`).

Název aktuálního fontu získáme `\fontname\font`.

Definice 9.2 (fontdimen)

`\fontdimen číslo \cs` je registr pro daný fontdimen, kde číslo =

1. stant (sklon, jaká je šířka na 1pt výšky)
2. Mezislovní mezery
3. -||-
4. -||-
5. ex
6. em
7. extra mezera (viz `\xspaceskip`)

Poznámka
Ještě se lze ptát na `\hyphenchar\font` = kód rozdělítka a `\skewchar\font` = matematické cosi.

Poznámka
`\magnification promile` znásobí velikost od zavolání do konce jobu (hlavně v dvi), 12truept je 12pt při ignorování magnification. Navíc existuje `\magstep n` = `\magnification 1000 * 1,2^n` a `\maghalf` = `\magnification 0.5`.

Definice 9.3 (Soubory)

Začíná se souborem .mf, ten METAFONT převede do .tfm (metriky) a .gf („obrázky“). .gf závisí na rozlišení a podobných věcech (celkově třeba tiskárně) se dále převede pomocí

GFTOPK .pk („obrázky2“). TeX použije metriky (.tfm) a vyrobí .dvi. DVIPS potom vezme .dvi a .pk a vyrobí z toho .ps.

TeX už většinou umí volat METAFONT, kdyby nenašel chtěné soubory.

Pro .gf, .pk a .dvi existují programy na prohlížení (GFTYPE, PKTYPE, DVITYPE). .tfm se dá překládat tam a zpět pomocí PLTOTF a TFTOPL.

Co je TFM: základní hlavička (jméno, design size) + rozměry znaků (w/h/d + italické korekce, kódované přes omezeně velkou tabulku (pouze 64 různých šířek písmen → METAFONT dělá ztrátovou kompresy)) + program pro kerny a ligatury + seznam variant pro velké operátory a delimitery + fontdimens.

Definice 9.4 (Postscriptové fonty)

Type1 (= bezierovské segmenty), type3 (= PS programy), type32 (= bitmapy), type42 (= TrueType zabalený do PS), type0 (= composite fonty).

Jsou 2 soubory .pfa (častěji komprimovaná verze .pfb) a afm (adobe font metric, komprimovaně .pfm). Do TeXu: .pfm PFMTOAFM .afm AFM2TFM .tfm TEX .dvi DVIPS(+.pfa/.pfb) .ps.

Nebo (tzv. systém virtuálního fontu) AFM2TFM může ještě dostat .enc, čímž se ještě vyrobí .vpl, ze kterého se VPTOVF vytvoří .tfm pro TEX a .vf pro DVIPS. (Existuje i inverzní VFTOVP).

DVIPS navíc ještě dostává font map (jeden řádek = `rptmr Times-Roman < ptur_8a.pfb` = jméno v TeXu + jméno v PostScript světě + „vlož“ (=<) + soubor).

Definice 9.5 (TrueType fonty)

Vznikly ve Windows, mají jen beziérky druhého řádu (= jsou omezené). Lze je převést .ttf TTF2AFM .afm, nebo (.tff TTF2TFM .tfm + .tff TTF2PK .pk).

Definice 9.6 (OpenType)

Kontejnerový obal, do kterého lze uzavřít skoro cokoliv (většinu PS a všechny TT). Existují převáděcí programy OTF2…