1 PDF

Poznámka (Historie)
Portable Document Format (Adobe, 90 léta) měl nahradit PostScript (Ten byl příliš komplikovaný. Dokonce i s doplněním o konvence DSC.). Na rozdíl od PostSciptu PDF není míněno jako programovací popis.

1.0 ani Medvěd neviděl, 1.7 už bylo v ISO standardu (ale lepší je číst PDF Reference přímo od Adobe než ISO standard).

Postupem vznikly profily (sady pravidel) pro různé použití: PDF / X = pro tiskárnu (zakazuje JS apod.), PDF / A = pro archivaci (univerzálně srozumitelná téměř jistě i za 30 let).

1.1 Lexikální struktura

Definice 1.1 (Obsah PDF)

PDF je key sensitive.

Znaky jsou nadmnožina ASCII. Dělí se na mezery (mezera, TAB, FF, NUL, CR, LF, řádek končí LF / CR LF), oddělovače (`()[]{}<>/%`), ostatní (u těch není specifikováno, jaké je kódování).

`%` značí komentář do konce řádku (formálně mezera).

Klíčová slova (začínají písmenem a jdou až do mezery): null, true, false, f*.

Čísla jsou buď celá nebo floaty a píší se standardně.

Stringy se píší do kulatých závorek (ne uvozovek). Mohou být víceřádkové, mohou obsahovat i znak 0, mohou obsahovat vnořené kulaté závorky (správně uzávorkované), escapes (`\( \) \n \t \newline` se ignoruje `\ooo` kód znaku v 8 soustavě). Nebo může být uzavřen do menšítek, pak je ale celý v hexadecimálním formátu.

Jména jsou znaky kromě mezer a oddělovačů předcházené /, hexadecimální znak se dá zadat `#xx` (ale nelze znak s kódem 0), konvence utf-8.

Pole jsou `[prvky …]`, tedy třeba `0 false (str) [1 2]`. Slovník je `<< /jméno hodnota … >>` (slušnost je, aby se slova neopakovala). Konvencí je, že pokud slovník obsahuje `/Type`, tak je správně, např. `/Page`. Stream je tvaru `<< slovnik + /Length bytes >> stream <eol> …data stream` slovník navíc může obsahovat `/Filter /jméno` nebo `/Filter [/jméno …]` (např. /ASCIIHexDecode, /ASCII85Decode, /FlatDecode, /LZWDecode?).

Nepřímé objekty: definice = `{číslo objektu} {číslo generace} obj … endobj` odkazování se = `{číslo} {generace} R`

Definice 1.2 (Struktura)

První řádek verze, druhý řádek nesmyslné znaky (aby se neinterpretovalo jako textový soubor), obojí zakomentované.

Mělo by (musí tam někde být) končit `%%EOF` (i když zatím občas bývá nějaký balast), před tím jsou (v pořadí odshora): xref (pro rychlé přečtení dat o objektech, 2 čísla a `f-` nebo `n-`?), trailer (metadata o soboru, slovník) a startxref (kde začíná xref, číslo).

Mezi tím jsou objekty.

Definice 1.3 (xref)

Začínají xref a končí (už tam nepatří) trailer. Má sekce (začínají 2 čísly, pořadím sekce? a počtem znaků). Navíc každý řádek musí mít 20 bytů (takže TeX doplňuje mezeru kvůli CR LF (TeX používá jen LF)).

Položka (tj. 1 řádek) může být n, pak obsahuje pozici (10 číslic), generaci (5 číslic) a n. Nebo může být f (později, souvisí s generacemi a aktualizacemi PDF).

Definice 1.4 (trailer)

Slovník obsahující /Size + počet objektů, /Prev + odkaz (pozice) na předchozí xref, /Root nepřímý odkaz, /Info nepřímý odkaz, občas obsahuje /Encrypt, nepovinné, ale silně doporučené /ID + číslo verze při vytvoření + číslo aktuální verze.

Definice 1.5 (Updatování)

PDF je stavěné na update připsáním na konec. (Zajímavé třeba při podepisování.) Navíc xref má položku f (smazáno, nebo také free = volný obsahující odkaz (číselný) na další f objekt (resp. u 0 na 0) jako spojový seznam a číslo poslední generace (a f)). Navíc objekty mají generace, aby se daly recyklovat jejich čísla.

Definice 1.6

V novějších verzích se objekty ukládají do object streamů (aby se zmenšil text okolo). Ten vypadá: všechno generace 0, žádné další streamy, výjimky (nesmí obsahovat velikost jiného objektu atd.).

`Číslo gen obj << /Type /objStm /N #objektů /First pozice 1. objektu [/Extends] [/Fi`
kde stream obsahuje N párů čísel (čísla objekth a pozice), cokoliv, N objektů bez obj a endobj.

Následně také obsahuje Xref Stream (místo xref a traileru): `číslo gen obj << /Type /xref  polo`

Poznámka
Pro lepší práci s pdf se hodí program qpdf. Speciálně `qpdf --qdf --object-stream=disable` (výsledek se ukládá do souboru a pak ho lze po upravení zase 'zkomprimovat').

Definice 1.7 (Stranky)

Root objekt (viz trailer) obsahuje seznam dětí = stránek.

Definice 1.8 (High-level struktura)

Z traileru získáme info, ale získáme i odkaz na catalog (root), který obsahuje všechno. Obsahuje odkazy (page tree, page labels, …), viewer prefs (nějaké nastavení prohlížeče, např. zazoomování), jazyk (může být až na úrovni slov, když je dokument vícejazyčný, udává se např. kvůli ligaturám), version override (při updatu můžeme chtít použít novější verzi PDF, tak ta se uvádí zde).

page tree (počítá se klidně i s tisícemi stránek a málem paměti zařízení) se skládá z jednotlivých objektů, které mají typ pages, parent …, kids [. . .], count počet (aby se dalo rychle listovat). Listy pak mají typ page, parent …, resources (slovník), contents (stream), mediabox [velikost stránky[a]], cropbox [rozměry, na které ořezáváme veškeré kresby], bleedbox [kam může barva prosakovat] + trimbox [na co se bude ořezávat papír po opuštění tiskárny], rotate 90 (ne všechny prohlížeče respektují). (Boxy jdou definovat už v Pages, stejně tak další vlastnosti, jako přechody v prezentaci, …).

resources a contents: v resources jsou odkazy na objekty, protože nemohou být contents (protože je to stream a nemusel by být čitelný všem programům). resources můžou být nepřímý objekt, tedy je mohou stránky sdílet.

Content stream (vypadá jak postscript) – částečně zásobníkový jazyk (ale po provedení operátoru musí být zásobník prázdný), obshuje čísla a operátory (vždy seznam argumentů a operátor). Operátory jsou např (m = move, l = line (nekreslí, jen ji vytvoří), S = stroke (vykreslí)).

[a] Souřadnice jsou v bp (big pointech). Papír je popsán obdélníkem $[x_1 y_1 x_2 y_2]$.

Definice 1.9 (Operátory)

Nastavení parametrů kreslení (grafický stav = gstate)[a]: q / Q (save / restore – zásobník grafických stavů), $abcdef$ cm (definuje transformační matici (násobí se s předchozí?)), x w (line width = šířka obdélníku kolem úsečky, při $x = 0$ se nastaví 1 pixel),

Konce a napojení úseček: (butt ending = konec kolmý na úsečku, round ending = konec oblý se středem v konci, square ending = konec čtverec se středem v konci, zároveň existují 3 typy napojení (jedné lomené čáry) round = vyplní se kruhovou úsečí, tj. v podstatě round, bevel = spojí se vrcholy obdélníků, miter = protáhne se do špičky (pokud je moc daleko (tzn. víc jak miter limit), udělá se bevel)): typ J (line cap: 0 = butt, 1 = round, 2 = square), typ j (line join: 0 = miter, 1 = round, 2 = bevel), limit M (miter limit).

Dále čárkování: [pole délek] fáze d (dash pattern, pole délek = délky čáry, mezery, čáry,

mezery, …, fáze = kde v seznamu má začít, pole délek může mít i lichou délku, prostě se nekonečněkrát zopakuje za sebou, čárkovací pattern pokračuje i za zlomem, pokud je to jedna čára, line cap a line join se aplikují na každou čárku zvlášť).

Další věci se dají reprezentovat ExtGState objektem, na který se pak odkazujeme operátorem /jméno gs.

Konstrukce cest: xy m (move to, začíná úsek), xy l (line to), $x_1y_1x_2y_2x_3y_3$ c (curve to = bézierova křivka 3. řádu (x_3y_3 je konec, další dva body jsou směry odchodu a příchodu do koncových bodů)), h (close = nakreslí úsečku do počátku úseku), $xywh$ re (založí nový úsek a zkonstruuje obdélník), S (stroke, aktuálními parametry gstate se obtáhne cesta a zruší se), s (close & stroke = uzavře a obtáhne), f/f* (close? & fill = vyplní cestu, f* počítá počet průsečíků a podle parity určí, zda vyplnit, f počítá počet orientovaných průsečíků a porovnává s 0), b/b* (close & fill & stroke), n (new = discard).

[a] Máme user space, grafický stav pak musí definovat tzv. CTM (current transform matrix?) a ta zobrazuje user space na device space. Obecně jsou PDF vektory $(x, y, 1)^T$ a CTM je 3×3

Definice 1.10 (Barvy)

Součást gstatu. Existují různé (přesně nedefinované) prostory: g G/g (device gray, $g \in [0, 1]$, větší hodnota jasnější), rgb RG/rg (device RGB), $cmyk$ (device CMYK), malé jsou výplň, velké hranice (stroke).

Poznámka (TeX)
TeX nastaví počátek userspace na aktuální referenci, pokud použijeme samotné `\pdfliteral{…}`. Pokud použijeme `\pdfliteral{…}` začne na začátku stránky.

Pozor
Transformace je lepší dělat přímo TeXem, jinak se rozsypou například odkazy.

Definice 1.11 (Další operátory)

Cestu můžeme zakončit příkazem w (w*). Další kreslení pak probíhá ořízнuté na vnitřek této cesty. (Po w může ještě následovat S, aby se ještě vykreslila. Nebo následuje n, aby se cesta, kterou se ořezává zahodila.)

Barvy mají ještě tzv. barevné prostory, viz přednáška…

BX (begin extension) … EX (end extension) = pokud se mezi nimi objeví neznámí operátor, má se ignorovat (a nehlásit chybu).

Definice 1.12 (XObject)

Obrázky, vložené stránky atd. Lze ho pojmenovat a pak volat jinde.

Definice 1.13 (Text)

Text začíná operátorem BT a končí ET. Musí obsahovat: /font velikost Tf (výběr fontu), x y Td (posun na konkrétní pozici), (řetězec) Tj (vykreslí řetězec). Dále může obsahovat: x Tc (character spacing = roztahování písmen), x Tw (word spacing = roztahuje slova), x Tz (horizontal scaling (v %)), x Ts (rise = na indexy a exponenty), x Tr (rendering mode: 0 = fill, 1 = stroke, 2 = obojí, 3 = nic, +4 je přidání do ořezávání).

K tomu ještě: x y Td (nový řádek na (x, y), relativně oproti začátku aktuálnímu řádku), x y TD (navíc nastaví leading (= rozpětí řádku) na -y, od té doby lze použít:), T* (= 0 -leading Td), a b c d e f Tm (set text matrix).

Vykreslení: (str) ' (= T* ∧ (str) Tj), [(str)kern(str)kern…] TJ (sází text s mezerami -kern/1000 aktuálních textových jednotek).

1.2 Ukládání dat

Definice 1.14

Slovníky můžou být moc velké = pomalé. Tedy se zavedl tzv. Name Tree, což je strom, který má v listech (uloženy ve vnitřních vrcholech nad nimi) stringy seřazené podle abecedy, vnitřní vrcholy mají intervaly pro vyhledávání (musíme se ale podívat do všech synů, abychom se dozvěděli, kam jít).

`<</Kids [references…] /limits [min max]>>` nebo `/Names [(key1) val1 (key2) val_2 …] /`

Obdobně funguje number tree.

1.3 Interakce

Definice 1.15 (Destinations = odkazy)

`page-obj /XYZ left top zoom`, `page-obj /Fit (stránka) nebo /FitV (šířka) nebo /FitH (výš`

Destinace jsou uložené v root katalogu v /Dests (odkaz na slovník) nebo /Names a /Dest (name dict = slovník odkazů na name tree).

V pdfTeXu se vytváří `\pdfdest name{jméno} (xyz | fitr | fitv | …)`.

Při kliknutí se neskočí, ale provede se tzv. akce (Action) `<< /Type /Action /S typakce a ještě ob`
Typy akcí jsou `/S /GoTo`, `/S /URI /URI (uri)`, `/S /Named /N /NextPage`, …

Definice 1.16 (Outline)

Další zvrhlý stromeček, kde se ukládá např. odkaz. Vrcholy se zde odkazují na prvního a posledního syna, na vedlejší sourozence (max 2) a na otce. Navíc obsahují informaci o tom, kolik je otevřených položek v podstromu, title a co se stane, když se zmáčkne (/A

action, /C [r g b], /F flags (bit 0 je kurzíva, bit 1 je tučné)).

V TeXu: `\pdfoutline attr{…}, akce (např. goto page N), user {…}, count N`, kde N je velikost podstromu (rozbalený) nebo - velikost podstromu (sbalený)

Definice 1.17 (Anotace)

Stránka může mít ve svém slovníku /Annots annotation nebo [annotations]. Ty mají /Type /Annot, /SubType …, /Rect [oblast, k čemu anotace patří], /Contents string, /P page, a dále /Border [horizontal Corner radius, vertical Corner radius, width of line, někdy opt. dash array], /C [gray / r g b / c m y k] (pozná se podle počtu prvků v seznamu).

Používají se na všechno možné. Subtypy jsou např. /Text, /FreeText (kreslí se přímo na stránku) /Line (úsečky / kóty), /Highlight (zvýraznění), /Link (/A action /Dest dest).(Existuje i např. anotace na kótování technických výkresů).

pdfTeX umí `\pdfannot, \pdfstartlink … \pdfendlink`

Definice 1.18 (PageLabels)

Root katalog může obsahovat /PageLabels a odkaz na nametree. Page labels pak obsahuje `<</Type /PageLabel /S style>>`, kde style je (/D decimal, /R ROMAN, /r roman, /A uppercase A-Z, /a lowercase a-z).

TODO

TODO Shading & Patterns

2 MetaFont

Poznámka
Původně bylo myšleno hlavně na parametrické fonty (parametry byly zvlášť (roman, italic, …), písmo zvlášť).

Definice 2.1 (Typy proměnných)

boolean, string, path, pen, picture, transform, pair, numeric (defaultní)

+ MetaPost: color

Deklaruje se `typ název`.

Definice 2.2

Znaky se dělí na: písmena + mezera, (`<=>:|`), uvozovky, + -, `\/*`, ?!, `^~`, [,], `{}`, ., loners (,;()), další (”0-9).

Druhy tokenů: řetězcové ("…"), numerické (číslice a nejvýše 1 tečka $< 2^{12}$, rozlišení 2^{-16}), symbolické (dokud se opakuje stejná skupina znaků (např. uvozovky, písmena, …) krom loners, tvoří se 1 symbolický token, může obsahovat i .).

Spark = symbolický token s definovaným výrazem (existuje něco jako let v TeXu pro změnu), tag = vše ostatní.

Proměnná = `<tag><suffix>` nebo vnitřní hodnota (suffix může být prázdný, nebo suffix tag nebo suffix subscript (numerický token nebo numerický výraz)).

Definice 2.3

$x++y$ je $\sqrt{x^2+y^2}$, $x+-+y = \sqrt{x^2-y^2}$.

$z1 = (x1, y1)$ (tj. $zn = (xn, yn)$).

Definice 2.4 (Řídící konstrukce)

if bool.výraz: … elseif bool.výraz:…else:…fi

for x=A,B,C,…: tělo(x) endfor

for x = A step B until C: tělo(x) endfor

forsuffixes s=A,B,…: tělo(s) endfor

forever: tělo endfor

Pozor
Cykly se expandují, nevykonávají!

Definice 2.5 (Cesty)

Operátor length vrací počet úseků, ze kterých se daná cesta skládá. Úsek jsou krajní body + 2 body (bézierova křivka 3. řádu).

Definice 2.6 (numeric)

Může být unknown, dependent (závislá na jiné), known. Výrazy jsou pouze lineární.

Můžeme napsat např. `z5 = whatever[z1, z2]` a `z5 = whatever[z3, z4]`, kde `x[a, b]` je lineární kombinace $(1-x)\cdot a + x \cdot b$. Tyto dva výrazy 'uloží' do z5 průsečík dvou úseček (přímek?), jelikož whatever je proměnná, která je při každém použití nová (unknown) a jinak se počítá soustava lineárních rovnic.

Definice 2.7 (Další užitečné funkce)

known, numeric, > (booleany, poslední porovnává případně lexikograficky), min, floor, ceiling, sind, cosd (ve stupních), angle (vrací úhel daného vektoru od (1, 0), inverze (na jednotkovou kružnici) dir), mlog, mexp.

normal deviate, uniform deviate (náhodné generátory)

Definice 2.8 (Transformace (vstupem dvojice čísel = komplexní číslo vlevo))

scaled, xscaled, yscaled, rotated (vstupem jedno číslo vpravo), shifted (dvě čísla), zscaled (násobení komplexních čísel (tj. dvojic)), dotprod

Definice 2.9 (Priorita operátorů)

Primary = funkce, Secondary = násobení dělení, Tertiary = sčítání apod, Výraz = vše ostatní.

Definice 2.10 (Definice)

`def symb.token = replacement enddef;` nebo s parametry `def symb.token (typ nazev, …) = rep` kde typ může být suffix, text a expr.

3 Křivky

Definice 3.1 (Bézierova křivka)

Bézierova křivka řádu (stupně) d je parametrická ($t \in [0,1]$) křivka určená body $a_0^d, \ldots, a_d^d$. Definujeme $a_i^j = a_i^{j+1} \cdot (1-t) + a_{i+1}^{j+1} \cdot t$ a výsledný bod je a_0^0. Tedy výsledný polynom je $\sum_{i=0}^{d} \binom{d}{i}(1-t)^{d-i}t^i \cdot a_i^d$. To jsou tzv. Bernsteinovy polynomy.

Pozorování (Vlastnosti B. křivek)
Je (koeficienty l. kombinace řídících bodů, která dává bod křivky, jsou) invariantní vůči lineárním transformacím. Leží v konvexním obalu řídících bodů.

Jakákoliv B. křivka řádu d je taktéž B. křivka řádu $> d$. $d \to d+1$: $a_0 \to a_0$, $a_d \to a_{d+1}$, $\frac{i}{d+i}a_{i-1} + \left(1 - \frac{i}{d+i}\right) a_i \to a_i$.

Křivku lze rozdělit v bodě t na dvě B. křivky stejného řádu. Řídící body jsou pak $a_0^d, a_0^{d-1}, \ldots, a_0^0$ a $a_d^d, a_{d-1}^{d-1}, \ldots, a_0^0$.

Popis maticí: $a(t) = (t_0, \ldots, t^d) \cdot M \cdot (z_0, \ldots, z_d)^T$, kde M je matice koeficientů.

Poznámka
Jelikož po křivce se „cestuje“ různě rychle, tak se špatně počítá správě hustá aproximace. Proto se rekurzivně křivka dělí, dokud spojnice koncových bodů není dobrá aproximace. Obdobně se hledá průnik (větve, kde jsou disjunktní obaly řídících bodů, můžeme zahodit). Také se tak počítá aproximace délky křivky, bod v dané vzdálenosti, inverze, …

Hledání bodů, které jsou nejvýše δ vzdáleny, zvyšuje řád!

Definice 3.2 (Splines)

Narozdíl od B. křivek prochází všemi řídícími body. Fungují tak, že se křivka rozdělí na části mezi 2 následujícími řídícími body, tyto části se pak vyjádří jako kubické (nejčastěji) křivky tak, aby celá křivka měla spojité derivace do 2. řádu (nejčastěji) (stačí v řídících bodech). Pokud je uzavřená, tak je to plně definovaná, jinak se většinou chce, aby derivace v krajních bodech byly nulové.

4 Asymptote

Definice 4.1 (Asymptote)

Inspirován MetaFontem a MetaPost. ALe je to C-like jazyk (pozor, `++` a `--` jsou vždy prefixové, aby se nepletli se spojováním linek, dále nemáme bitové operátory, složené typy jsou referenční a používají se pouze 1st class funkce, navíc lze přetěžovat funkce a operátory).

Křivkový model MF generalizovaný do 3D a výstup v PS nebo PDF. Naopak neřeší rovnice (na takové věci se většinou používají knihovny). Navíc můžeme generovat TeXové popisky (jednoduše, MF nebo PS to umí také).

Definice 4.2 (Typy)

void, bool (false, true), bool3(false, true, default), int, real (float, tj. menší přesnost než MF, kde to byly „inty“), pair (komplexní číslo), triple (trojrozměrné matice), string (bytestring, zadávají se `"…"`, kde se překládá jen `\"`, nebo `'...'`, kde se interpretují i jiné escapované znaky (např. `\n`)), path (s měřítkem), guide (bez měřítka), pen, transform, transform3, frame (s měřítkem), picture (bez měřítka), file, struktury, pole (homogenní).

Definice 4.3 (Zadávání křivek (syntax guides))

`..` je spojení dle Hobbyho, `--` spojení úsečkou (funguje `..cycle`), `(0, 0)..controls(a, b)and(c, d)..` specifikuje bezierku, `..tension x..` a `..tension x and y` specifikuje napětí (parametr pro křivky), `..tension atleast x..` (magie, která nějak mění křivku, `:: = ..tension atleast 1..` zařídí, že je křivka v trojúhelníku určeném krajními body a směry, `__ = ..tension atleast ϖ..` zařídí spojitost křivosti v uzlech, jinak vypadá jako úsečka), `p&q` je slepení `p` a `q` za koncové body, `p^q` je disjunktní sjednocení (např. při výrobě mezikruží).

Dále se dá říct `(0, 0){směr}`, nebo $(0,0)curl2$.

Definice 4.4 (Operace na cestách)

length (počet částí), size (počet vrcholů), cyclic(path) (je cyklická?), straight(path, int), piecewisestraight(path), point(path, int/real) (pro reálné vrací bod na křivce), dir(path, real, bool (normalize=true)), … (`https://asymptote.sourceforge.io/asymptote.pdf`)

Definice 4.5 (příkazy)

`int x; int x=5; int x[]; int x[]={1, 2, 3}; int x[5]; x=new int[]{1,2,3};` (Při dosazení do indexu > velikost se automaticky rozšíří). `for(int i=0; i < 10; ++i)` nebo `for(int i:x)`.

`x` pole, pak `x.length, x.cyclic, x.push(…), x.pop(…), array(n, x)` (n kopií x), `copy, concat, max, min, map(f, pole)`, `x+y` (po složkách), `x[array], x[2:6], x[:]` (kopie nekopírují cykličnost) (takto lze i přiřazovat). `Sequence(n) = {0, …, n-1}, Sequence(m, n) =`

Funkce: `int f(real x, real y) {…}` neboli `int f(real x, real y) = new int(real x, real y` (všechny funkce jsou přiřazené anonymní funkce), `int f(real x=0, real y=x)` (běžné volání s defaulty), `int f(keyword int x)` (musí se volat `f(x = 10)`), `void f(real x … real y[])` (zbylé args se předají jako pole y), stejně tak se dá volat `f(1…pole)`.

`typ operator znak(typ x, typ y){…}` (znak můžeme použít standardní operátory, nebo operátory bez významu (`$, $$, @, @@`)).

Definice 4.6 (Automatická konverze)

int → real → pair → (→ guide) → path

real → path

Manuální se dělají normálně (`(string) 5`) (v definici funkce se dá říct `explicit`, čímž se v parametrech zakážou automatické konverze).

Automatická konverze se definuje jako `typ operator cast (typ x){…}`.

Definice 4.7 (Transformace)

Aplikování: `transformace*(pair|guide|path|…)`, `transformace*transformace`. Máme `identity(), shift(x, y)/shif(pair), xscale(x)/yscale(y)/scale(…), stant(s)` (zkosení), `rotate(deg, [pair]), reflect(pair a, b)` (osová symetrie)

Definice 4.8 (Příkazy na párech)

`length(p), angle(p), degrees(p), unit(p), xpart(p), ypart(p), dot(p, q), *` (komplexní násobení), `dir(deg), expi(rad), interp(x, y, t) = (1-t)x + ty`.

Definice 4.9 (Pera)

A+B je kombinace (A má přednost). currentpen. a*pero (saturace barvy). Pera: `gray(a), 0<=a<=1, rgb` (nebo rovnou pen + bp), `linetype(array, offset)` (čárkované), `(square/roundcap/extends)cap`, `(miter/round/bevel)join`.

K textu lze nastavit peru `(no)basealign`, `fontsize(pt [, lineskip_pt])`.

TODO

5 LuaTeX

Definice 5.1

LUA + API k TeXu, ε-TeX, pdfTeX, Aleph (unicode, OFM místo TFM, sázení všemi směry), sazba TrueType/OpenType fonty.

Volání Luy: `\directlua{expanded text}`, např. `\directlue{dofile("něco.lua")}` nebo `\directlua{tex.print(tex.count[0])}`. `\latelua{…}` se vykonává při `\shipout`.

`\luaescapestring{…}` odescapuje speciální znaky, např. pro házení textu do lua stringu.

Při zavolání `luatex --luaonly soubor` to soubor začne vykonávat jako Luu, z ní pak lze volat TeX.

Změny v TeXu: více catcode tables. TODO.

5.1 LUA (5.3)

Definice 5.2 (Typy)

Hodnotové: nil, boolean (false je pouze false z boolean a nil), number (v 5.3: double precision i int, v 5.2 jenom double precision), string (posloupnost bytů).

Referenční: function, table (jako Pythoní slovník, ukládá cokoliv kromě nilu, používá se i jako tzv. sequence: klíče 1, 2, 3, …, n), thread (Lua má coroutines), userdata (data z vnějšku: lightuserdata = Cčkový void*, fulluserdata spravovaná Luou (např. se na ně aplikuje garbage collector)).

Zkratka: `tab.xyz` je `tab["xyz"]`.

Meta-tabulky: používají se na definice „typů“. Garbage collector: weak reference nepočítá, takže objekt bez strong reference smaže (a weak nahradí nilem).

Definice 5.3 (Syntax)

Středníky jsou nepovinné. Bloky nejsou tvořeny závorkami, ale `do…end`. Přiřazuje se `variablelist = exp` (přebytečné expresion se zahodí, do přebytečných proměnných se přiřadí nil). Funkce vrací i více proměnných (pokud je na konci explistu (nebo samotná), tak se výsledky přidají do seznamu, jinak se bere jen 1).

`if cond then … [elseif cond then] … [else] … end`. `while cond do … end`. `repeat … until break, goto label`. `::label::`. `return explist` (!musí být na konci bloku). `for var=from,to[,step] for varlist in explist do … end`.

`function name(args …) … end` (ve skutečnosti `f = function (args …) … end`). Moduly se pak definují: `mod = {}; function mod.f(…) … end`. Také funguje `function obj:f(x, y) … en` což vytvoří `obj.f = function(self, x, y) … end`. Obdobně se dá volat: `obj:f(x, y)`. Též `function f(x, ...) … end`, kde `...` se uvnitř funkce expandují na zbytek parametrů.

Operátory: `//` je floor division. `^` je umocňování. `..` konkatenace (zprava asociativní, slovníky, stringy). `...` jsou varargs. `==` `~=` jsou rovná se a nerovná se. `and, or, not`. `f "string"` a `f {…}` jsou volání funkce. `#` je délka řetězce / sekvence (na obecné tabulce nefunguje). `{1, 3, 5}` je konstruktor sekvence, `{[1] = 5, ["xyz"]=7, abc = 9}` tabulek (prvkům bez klíče se přiřazuje klíč 1, 2, 3, …, bez ohledu na to, které byly explicitně použity).

`-- …` je komentář. `[===[ … ]===]` (tzv. long brackets, rovnítek může být libovolně, jen otevírací a zavírací se musí shodovat) se používá např. na long komentář: `--[=…=[…]=…=]` a fungují i přes více řádků.

Definice 5.4 (Knihovny)

basics: `assert(cond[, "msg"])` je standardní assert, `pairs(tab)` je iterátor přes klíč – hodnota, `ipairs(tab)` iteruje přes int klíče (aneb sekvence), `tonumber(…), tostring(…)` (pro explicitní konverzi, jinak se dělá automaticky), `type(expr)->string`.

strings: `s[1]…s[#s]`, `s[-1]`, `s:byte(i)`, `string.format`.

tables: `table.concat(tab[, sep])`, `table.insert(seq, pos, val)`, `table.remove(seq, pos)`, `table.sort(seq[, comp.])`.

5.2 Integrace s LuaTeXem

Poznámka
Z TeXu: `\directlua{…}`. Z Luy: knihovny pro sazbu (vidí vnitřní stav TeXu), callbacky.