Akademia

Co nieco o akapicie*

Marek Ryćko

Wstęp

Celem niniejszej pracy jest inne niż w książce The TEXbook uporządkowanie i zgromadzenie w jednym miejscu pewnej części wiedzy na temat akapitu w systemie TEX. Ideą pracy jest umieszczenie wszystkich informacji na temat niektórych *aspektów prezentowanej tu tematyki.*

Wybrane punkty zawierają komplet informacji na prezentowany tam temat, a pozostałe punkty mają charakter streszczenia lub zarysu i służą umiejscowieniu opisywanych szczegółów w całości spraw TEX-owych.

Praca przeznaczona jest dla osób, które dobrze znają TEX-a i chciałyby poznać najintymniejsze szczegóły działania pewnych jego mechanizmów. Autor sądzi, że praca ta może być wykorzystywana jako podręcznik (ang. reference manual) w sytuacjach, gdy podczas pracy z TEX-em trzeba dokładnie przewidzieć zachowanie tego systemu. Jest to szczególnie ważne dla „programistów TEX-owych", czyli osób piszących własne zestawy makr.

Użyteczność tej pracy dla osób początkujących jest ograniczona koniecznością rozumienia wielu nie opisanych tu fragmentów TEX-a, które są ściśle powiązane z prezentowaną tu tematyką. Aby jednak choć trochę ułatwić początkującym korzystanie z tych tekstów, prezentujemy w punkcie „Zakładana wiedza" zarys podstawowej wiedzy o TEX-u, której nie opisujemy w szczegółach, ponieważ znacznie wykraczałoby to poza ramy niniejszej pracy

Podstawowym źródłem informacji, które posłużyło do opracowania tego tekstu, była książka D.E. Knutha The TEXbook, (Addison-Wesley Publishing Company 1990). Wydania tej książki od pierwszego do szesnastego (opisujące TEX-a < 3.0) miały ten sam układ stron i różniły się tylko drobnymi szczegółami. Począwszy od wydania siedemnastego (opis TEX-a ≥ 3.0), układ stron został nieznacznie zmieniony. Odwołania do książki The TEXbook dotyczą wydania osiemnastego i zapisywane są następująco: $[283^4]$, *oznacza stronę 283, akapit 4 od góry.*

Autor będzie wdzięczny czytelnikom za wszelkie uwagi dotyczące treści i formy tej pracy, w szczególności sposobu prezentacji materiału, jego użyteczności, a także ewentualnych niedociągnięć czy błędów.

$*$: *Tekst ten i następujący po nim* Główna lista pionowa w systemie TEX *były przedstawione jako szkolenia na konferencji w Bachotku odpowiednio w 1993 i 1995 r. Dziekujemy Autorowi za zgodę na przedruk na łamach BG [Red.]*

Zakładana wiedza

TEX jest bardzo złożonym systemem i wszystkie jego fragmenty są ściśle ze sobą związane. Bardzo trudno jest opisać dokładnie jeden z nich, nie opisując równie dokładnie pozostałych lub nie zakładając dokładnej wiedzy czytelnika na temat pozostałych.

Poniżej przedstawimy skrót tego, co zakładamy, że czytelnik wie o TEX-u. Służy to m.in. uporządkowaniu informacji oraz ustaleniu stosowanego dalej słownictwa.

TEX jako system przetwarzania plików. TEX zajmuje się czytaniem i zapisywaniem plików. Pliki przetwarzane przez system TEX mogą być: **tekstowe** (pliki ASCII) oraz **binarne**.

TEX w czasie pracy **czyta**:
- pliki tekstowe typu `TEX`,
- pliki binarne `TFM` z informacjami o fontach,
- pliki tekstowe czytane komendą `\read`.

TEX **zapisuje**:
- plik binarny `DVI`,
- informacje tekstowe na ekranie,
- plik tekstowy `LOG` (tzw. plik transkrypcyjny),
- pliki tekstowe pisane komendą `\write`.

W niniejszej pracy opisujemy fragment drogi od wejściowego pliku `TEX` do wyjściowego pliku `DVI`.

Przegląd podstawowych mechanizmów TEX-a. System TEX przetwarzając wejściowy plik lub zestaw plików typu `TEX` na wyjściowy plik `DVI` oraz czytając i zapisując pliki pomocnicze pracuje równocześnie na kilku poziomach:
- czytanie znaków pliku wejściowego i zamiana ich na żetony (analiza leksykalna),
- rozwijanie żetonów,
- wykonywanie komend,
- tworzenie list (poziomych, pionowych i matematycznych),
- tworzenie pudełek,
- zapisywanie pudełek do pliku `DVI` (za pomocą komendy `\shipout`).

Żetony, rozwijanie i komendy. Każdemu żetonowi utworzonemu na poziomie analizy leksykalnej przypisane jest jego bieżące znaczenie. Biorąc pod uwagę to znaczenie, żetony można podzielić na kilka rodzajów.

Ze względu na **rozwijalność** żetony możemy podzielić na: **rozwijalne** (ang. *expandable*) i **nierozwijalne** (ang. *unexpandable*).

Ze względu na „podstawowość” żetony dzielą się na: **podstawowe** (ang. *primitive*), czyli wbudowane w system, i **zdefinowane przez użytkownika**, czyli makra (ang. *macros*).

Wszystkie nierozwijalne żetony są podstawowe (użytkownik nie może zdefiniować żetonu nierozwijalnego).

Wśród żetonów rozwijalnych są zarówno żetony podstawowe (np. `\the`, `\number`, `\csname`, `\if`), jak również żetony zdefiniowane przez użytkownika.

Analizator leksykalny TEX-a (najwyższy poziom jego działania) zamienia ciągi znaków z pliku wejściowego na żetony i ustawia je w kolejce żetonów wejściowych (ang. *TEX's input*), przeznaczonych do dalszej obróbki.

Wszystkimi poziomami TEX-a steruje mechanizm wykonywania komend (poziom trzeci na liście z punktu *Przegląd podstawowych mechanizmów TEX-a*). Praca TEX-a zaczyna się zawsze od uruchomienia tego poziomu. To ten poziom na początku pracy systemu, aby wykonać pierwszą komendę, musi odczytać jej nazwę z pierwszego żetonu wejściowej kolejki żetonów. Wtedy właśnie przystępuje do akcji najwyższy poziom zamieniając początkowy ciąg znaków na pierwszy w kolejce żeton.

Nie każdy jednak żeton jest interpretowany jako komenda. Komendą jest jedynie żeton *nierozwijalny* znajdujący się na początku kolejki. (Taki sam żeton – jak opisano poniżej – nie znajdujący się na brzegu kolejki, może nie być komendą – może m. in. pełnić rolę argumentu komendy).

Jeśli w momencie, gdy mechanizm wykonywania komend przystępuje do realizacji kolejnej komendy, na początku kolejki żetonów wejściowych znajduje się żeton **rozwijalny** (podstawowy lub nie), to najpierw uruchamiany jest mechanizm rozwijania (poziom drugi), który rozwija żetony, czyli zastępuje w ściśle określony sposób początkowy fragment kolejki żetonów innym ciągiem żetonów. Czynność rozwijania powtarzana jest tak długo, aż na początku kolejki znajdzie się żeton nierozwijalny. Dopiero wówczas mechanizm wykonywania komend przystępuje do działania.

Jeśli komenda może lub musi mieć argumenty (np. w komendzie `\hskip6pt` argumentem jest `6pt`), to za komendę uważa się tylko pierwszy żeton (tu: `\hskip`) [rozdz. 24, 268^1]. Częścią procesu wykonywania komendy jest przeczytanie argumentów z kolejki żetonów wejściowych (z rozwijaniem lub bez).

Żetony opatruje się różnymi przymiotnikami, zależnie od sposobu ich powstawania ze znaków pliku wejściowego, i zależnie od bieżącego znaczenia. Zakładamy, że czytelnik zna pojęcie żetonu znakowego (ang. *character token*), jawnego (ang. *explicit character token*) i niejawnego (ang. *implicit character token*), żetonu kontrolnego (ang. *control sequence*) oraz pojęcie kategorii (ang. *category code*).

Tryby i listy. Wskutek wykonywania przez TEX-a komend powstają m. in. tzw. listy (poziome, pionowe i matematyczne).

W każdym momencie pracy TEX znajduje się w jednym z sześciu następujących trybów (ang. *modes*) [85], z których każdy określa rodzaj tworzonej przez system listy:

- **tryb pionowy** (ang. *vertical mode*) – tworzenie głównej listy pionowej (ang. *main vertical list*), z której powstają strony tekstu; (lista pionowa podzielona jest zawsze na dwie części: „bieżącą stronę” (ang. *current page*) i „ostatnie przyczynki” (ang. *recent contributions*));
- **tryb wewnętrzny pionowy** (ang. *internal vertical mode*) – tworzenie listy pionowej za pomocą jednej z komend: `\vbox`, `\vtop`, `\vcenter`, `\valign`, `vadjust` lub `\insert` [87_3]; lista ta służy na ogół do skonstruowania pudełka pionowego;
- **tryb poziomy** (ang. *horizontal mode*) – tworzenie listy poziomej służącej do skonstruowania akapitu (ang. *paragraph*); w niniejszej pracy zajmujemy się właśnie fragmentami procesu tworzenia takiej listy;
- **ograniczony tryb poziomy** (ang. *restricted horizontal mode*) – tworzenie listy poziomej za pomocą komendy `\hbox` lub `\halign` [87_3]; w wyniku realizacji każdej z tych komend powstaje jedno lub więcej pudełek poziomych (ang. *horizontal box*);
- **tryb matematyczny** (ang. *math mode*) – tworzenie formuły matematycznej do umieszczenia na liście poziomej;
- **wyróżniony tryb matematyczny** (ang. *display math mode*) – tworzenie formuły matematycznej do umieszczenia w osobnej linijce.

Na początku pracy TEX-a wszystkie listy są puste i system znajduje się w trybie pionowym. Tworzenie każdej listy polega na dopisywaniu do niej kolejnych elementów.

Listy matematyczne zamieniane są po ich utworzeniu na listy poziome. Lista pozioma powstała w trybie poziomym (akapit) jest zamieniana na listę pionową. Lista pionowa powstała w trybie pionowym jest zamieniana na pudełka przez mechanizm tworzenia stron (ang. *page builder*).

Skonstruowane pudełka, do których zastosowano komendę `\shipout`, są zapisywane do pliku `DVI`.

Niniejsza praca opisuje szczegóły pracy TEX-a na początku i na końcu tworzenia listy poziomej:

- od chwili rozpoczęcia tworzenia listy poziomej
- do chwili dołączenia linijek akapitu (powstałych z listy poziomej) do zewnętrznej listy pionowej.

Przegląd tworzenia akapitu

W procesie tworzenia przez system TEX akapitu można wyodrębnić następujące trzy fazy:

1. **rozpoczęcie tworzenia listy poziomej:** przełączenie TEX-a z trybu pionowego lub wewnętrznego pionowego na tryb poziomy, czasowe zawieszenie tworzenia listy pionowej, rozpoczęcie tworzenia listy poziomej (szczegóły – patrz punkt *Rozpoczęcie tworzenia listy poziomej*),
2. **tworzenie listy poziomej:** realizacja komend TEX-a (podstawowych, nierozwijalnych żetonów) polegająca m. in. na dołączaniu elementów do listy poziomej (tej fazy nie opisujemy w niniejszej pracy, patrz punkt *Tworzenie listy poziomej*),
3. **zakończenie tworzenia listy poziomej** obejmujące m. in.:
   3.1. zamianę listy poziomej na materiał pionowy (linijki, odstępy między nimi i inne elementy),
   3.2. umieszczenie materiału na liście pionowej, (szczegóły – patrz punkt *Zakończenie tworzenia listy poziomej*).

Rozpoczęcie tworzenia listy poziomej

W niniejszym rozdziale opisujemy, w jaki sposób realizowana jest zmiana trybu TEX-a znajdującego się w jednym z trybów pionowych na tryb (zwykły) poziomy.

Zmianę tę realizuje poziom (mechanizm) TEX-a wykonujący komendy. Jednym z ważniejszych skutków takiej zmiany trybu jest rozpoczęcie tworzenia listy poziomej [85–86]. Nie opisujemy tu szczegółów wykonania poszczególnych komend, tylko wspólne dla nich zachowanie TEX-a przy zmianie trybu.

Pewną grupę komend (czyli żetonów podstawowych i nierozwijalnych, które w odpowiednim momencie znalazły się na początku kolejki) będziemy określać mianem *przełączników* (nie ma odpowiednika w angielskiej terminologii Knutha), ponieważ służą do przełączania trybu pionowego lub wewnętrznego pionowego na tryb poziomy.

Wśród przełączników wyróżnimy dwie grupy komend:

- `\indent` i `\noindent` [282–283], służące do wykonywania jawnej zmiany trybu,
- *komendy poziome* (ang. *horizontal commands*) wymienione poniżej [283[4]], które napotkane w trybie pionowym również powodują zmianę trybu na (zwykły) poziomy.

Lista komend poziomych:

- litera (ang. *letter*): żeton znakowy, kategorii 11, również niejawny, np. `\d` po wykonaniu przypisania `\let\d A`; [283[4]];
- inny znak (ang. *other character*): żeton znakowy, kategorii 12, również niejawny, np. `\colon` po wykonaniu przypisania `\let\colon:` [283[4]]; do tej kategorii w PLAIN TEX-u należą m. in. wszystkie cyfry,
- `\char`,
- żeton typu „chardef” (ang. *chardef token*), czyli żeton kontrolny, któremu przypisano znaczenie komendą `\chardef` (np. żeton kontrolny `\ae` po wykonaniu przypisania `\chardef\ae="1A`; taki żeton użyty w roli komendy powoduje złożenie znaku z reprezentowanej przezeń pozycji bieżącego fontu, w przypadku symbolu `\ae` – jeśli bieżący font ma układ PL lub CM – jest to znak „æ” znajdujący się w foncie w pozycji `"1A`; żeton typu „chardef” może być też użyty jako parametr innej komendy i wówczas oznacza liczbę, np. `\box\ae` oznacza to samo co `\box26`),
- `\noboundary` (nowy żeton podstawowy, który pojawił się w TEX-u 3.0),
- `\unhbox`, `\unhcopy` (niezależnie od tego czy i co jest w środku pudełka będącego parametrem),
- `\valign`,
- `\vrule`,
- `\hskip`,
- `\hfil`, `\hfill`, `\hss`, `\hfilneg` (te żetony są podstawowe w TEX-u – nie są makrami – mimo że można by efekty ich działania uzyskać za pomocą `\hskip` z odpowiednimi parametrami),
- `\accent`,
- `\discretionary`, `\-` (komenda `\-` jest podstawowa, choć w swoim działaniu jest równoważna

`\discretionary{-}{}{}`, a ściślej `\discretionary{\char h}{}{}`, gdzie h jest wartością parametru `\hyphenchar` bieżącego fontu, pod warunkiem, że $0 \leqslant h \leqslant 255$; jeśli zaś warunek ten nie jest spełniony, to `\-` jest równoważna `\discretionary{}{}{}` [455^3]),
- `\␣` (odstęp kontrolny `\␣` jest komendą podstawową i użyty w trybie pionowym zmienia tryb na poziomy, choć zwykły odstęp ␣, ogólnie każdy żeton znakowy, jawny i niejawny, kategorii 10 (odstęp) (ang. *space token*), użyty w trybie pionowym, jest pomijalny i nie zmienia trybu),
- `$` (również pierwszy z pary `$$` rozpoczynającej wyróżnioną formułę matematyczną).

Należy zwrócić uwagę, że realizacja komendy oznaczającej pudełko (m. in.: `\hbox`, `\vbox`, `\vtop`) w trybie pionowym lub wewnętrznym pionowym nie powoduje zmiany trybu, tylko powoduje umieszczenie tego pudełka na liście pionowej.

TEX realizujący komendy w trybie pionowym lub wewnętrznym pionowym po napotkaniu w kolejce żetonów wejściowych *przełącznika* (komendy `\indent`, `\noindent` lub jednej z wymienionych wyżej komend poziomych) realizuje kolejno następujące czynności [85–86], [282–283]:

1. Warunkowe dołożenie pionowego kleju (ang. *vertical glue*) wielkości `\parskip` (w PLAIN TEX-u `\parskip=0pt plus 1pt`, [348_1]) do tworzonej listy pionowej.
   1.1. Jeśli TEX jest w trybie *wewnętrznym* pionowym, tzn. jest w trakcie konstruowania pudełka typu `\vbox` lub `\vtop`, i lista pionowa jest pusta, to klej nie jest dokładany [282_1].
   1.2. Jeśli TEX jest w trybie (zwykłym) pionowym, to klej jest dokładany do części „ostatnie przyczynki" (ang. *recent contributions*) listy pionowej, a następnie uruchamiany jest algorytm przenoszący elementy z „ostatnich przyczynków" do „bieżącej strony" (ang. *current page*) [122^2]. Może to spowodować w szczególności znalezienie właściwego miejsca podziału strony i uruchomienie procedury wyjściowej `\output` (patrz [rozdz. 23]). (Działanie TEX-a w trybie pionowym nie wchodzi w zakres tematyki opisywanej w niniejszej pracy.)
2. Przejście z trybu pionowego lub wewnętrznego pionowego do (nieograniczonego) trybu poziomego.
3. Ustalenie parametru `\spacefactor=1000` [285^3].
4. Zainicjowanie zapisywania *listy poziomej*.
   4.1. Jeśli przełącznikiem był `\noindent`, to na liście poziomej nie jest w tym momencie umieszczany żaden element.
   4.2. Jeśli przełącznikiem było cokolwiek innego (`\indent` lub dowolna komenda pozioma), to na liście poziomej umieszczane jest: puste pudełko poziome, szerokości `\parindent` (na podstawie obecnej wartości tego parametru; PLAIN TEX ustala `\parindent=20pt` [$348_{\text{wiersz } 10}$]).
5. Ustawienie na początku kolejki żetonów wejściowych (ang. *TEX's input*) następujących elementów:
   5.1. listy żetonów (ang. *token list*) `\everypar` (umieszczana jest bieżąca zawartość rejestru `\everypar`, PLAIN TEX ustala `\everypar={}`, czyli normalnie ta lista jest pusta),
   5.2. żetonu będącego przełącznikiem, jeśli jest to komenda pozioma (czyli jeśli przełączenie nastąpiło w wyniku wykonania komendy `\indent` lub `\noindent`, to żeton ten nie jest ponownie umieszczany w kolejce żetonów wejściowych);

   pozostała część kolejki nie ulega zmianie.
6. Uruchomienie algorytmu podziału na strony (ang. *page builder*), żeby sprawdzić, czy nie nazbierało się „dostatecznie dużo materiału" w trybie pionowym, żeby odciąć jedną stronę i ewentualnie zapisać ją do pliku DVI.
7. Rozpoczęcie wykonywania komend (dostarczanych przez mechanizm rozwijający) z kolejki żetonów wejściowych (patrz wyżej punkt 5).

Uwagi:

1. Lista `\everypar` może zawierać m. in. komendę `\par`, wielokrotne zmiany trybów, zmianę wartości `\everypar`, komendę `\end` itp. Komenda pozioma, która spowodowała rozpoczęcie składania akapitu może być w związku z tym przeczytana „bardzo późno" lub wcale.
2. Przykład. Jeśli przełącznikiem jest `\hskip` oraz `\everypar` zawiera `\let\hskip B`, to akapit zacznie się od „B".
3. Przykład. Jeśli

```
\everypar={Pierwszy
akapit.\par\everypar={}Drugi akapit.}
```

to rozpoczęcie akapitu od słów:

```
Ciąg dalszy...
```

spowoduje złożenie następujących dwóch akapitów:

Pierwszy akapit.
Drugi akapit. Ciąg dalszy...

4. Jeśli $$ włącza tryb poziomy, to tworzy się akapit zawierający jedynie \hbox to\parindent{}, którego składanie zawieszane jest na czas składania wyróżnionego równania.
5. Z punktu 1.2 i 6 wynika, że ewentualne błędy spowodowane reakcją TEX-a na zakończoną stronę, np. błędy wykryte w czasie analizy listy żetonów \output sygnalizowane są w pozornie niezrozumiałym miejscu (np. po pierwszej literze pewnego akapitu). Jeśli więc nie wiesz co błędnego jest w pierwszym znaku akapitu, to sprawdź zawartość rejestru \output.
6. Po zakończeniu etapu „rozpoczęcie tworzenia listy poziomej" lista pozioma jest pusta lub zawiera jedno pudełko (puste poziome pudełko szerokości \parindent).

Tworzenie listy poziomej

Opis sposobu tworzenia listy poziomej, tj. umieszczania na niej elementów, wykracza poza zakres niniejszej pracy.

Definicję listy poziomej i wykaz rodzajów elementów, jakie mogą się na niej znaleźć zawiera The TEXbook [94_1–95^1].

Systematyczny opis tego jak każda komenda TEX-a wpływa na stan listy poziomej (oraz na zmianę innych elementów wewnętrznego stanu TEX-a) zawierają rozdziały 24 i 25 TEXbook-a [267–287].

Zakończenie tworzenia listy poziomej

Poniżej opisujemy reakcję TEX-a (części TEX-a realizującej komendy) na nierozwijalne żetony powodujące zakończenie trybu poziomego, a ściślej: wspólną dla tych komend reakcję TEX-a na zmianę trybu.

Komendy kończące tryb poziomy. Tryb poziomy kończy się:

1. wskutek wykonania komendy podstawowej \par – jest to tzw. jawne zakończenie akapitu (należy zwrócić uwagę, że mówiąc o „komendzie podstawowej \par" mamy na myśli dowolny żeton, którego bieżące znaczenie jest takie, jak znaczenie żetonu \par w momencie rozpoczynania pracy przez system TEX; na przykład PLAIN TEX przypisuje żetonowi o nazwie \endgraf postawowe znaczenie komendy \par i w związku z tym komenda \engraf również służy do jawnego kończenia akapitu; z drugiej strony komenda o nazwie \par może mieć w danym momencie niestandardowe znaczenie i wówczas nie jest komendą kończącą akapit)
2. lub wskutek wykonania jednej z komend pionowych, należących do kategorii gramatycznej ⟨vertical command⟩ (w przypadku tych komend TEX postępuje tak jakby przeczytał komendę o nazwie \par, a następnie przystępuje ponownie do czytania komendy pionowej; jeśli żeton \par ma w tym momencie niestandardowe znaczenie, to właśnie ono jest tu realizowane).

A oto lista wszystkich **komend pionowych** [286_3]:

– \unvbox, \unvcopy,
– \halign,
– \hrule,
– \vskip,
– \vfil, \vfill,
– \vss,
– \vfilneg,
– \end,
– \dump.

Realizacja zakończenia składania akapitu. Wykonanie w trybie poziomym komendy \par (jej podstawowego znaczenia) polega na zrealizowaniu następujących czynności [99–100]:

Jeśli lista pozioma jest niepusta:

1. **Usunięcie z listy poziomej ostatniego elementu,** jeśli tym elementem jest klej (w szczególności dotyczy to kleju, który może być widoczny, powstałego w wyniku wykonania komendy \leaders; jeśli chcemy, by klej typu \leaders nie znikał na końcu akapitu, musimy po nim dodać albo coś nie będącego klejem, np. puste pudełko, albo dowolny klej, który zostanie usunięty).
2. **Umieszczenie na końcu listy poziomej:**
   2.1. kary \penalty10000 (zwanej w PLAIN TEX-u \nobreak),
   2.2. kleju o wielkości równej bieżącej wartości parametru \parfillskip, (podczas łamania listy poziomej na linijki TEX nie złamie linii na tym kleju z powodu poprzedzającej go kary),
   2.3. kary \penalty-10000 nakazującej złamanie linijki w tym miejscu.
3. **Ustalenie zasad łamania akapitu na linie:**

3.1. określenie nieskończonego ciągu (być może różnych) długości linii-pudełek,
3.2. określenie nieskończonego ciągu wielkości przesunięć tych linijek w poziomie,
3.3. określenie wielkości klejów poziomych (\leftskip i \rightskip) umieszczanych na początku i na końcu każdej linijki-pudełka,
3.4. określenie parametrów wpływających na wybór optymalnego łamania.

4. **Dokonanie łamania** na podstawie określonych wyżej parametrów i zamiana listy poziomej na skończony, niepusty, ciąg pudełek poziomych. Każde pudełko zaczyna się od kleju o wartości \leftskip, kończy się klejem \rightskip i zawiera skończony (być może pusty) ciąg elementów pochodzących z listy poziomej.
5. **Umieszczenie na liście pionowej** (tej, która była tworzona w momencie rozpoczynania akapitu) ciągu pudełek (wraz z odpowiednimi klejami, karami oraz innymi elementami) – patrz opis umieszczania akapitu na liście pionowej.

Dotychczasowe punkty są realizowane tylko, jeśli lista pionowa jest niepusta. Ponadto niezależnie od tego, czy lista jest pusta, czy nie realizowane są następujące czynności:

6. **Ustalenie (przywrócenie) podstawowych wartości parametrów** określających nietypowy kształt akapitu: \parshape=0, \hangindent=0pt, \hangafter=1 (specyficzny kształt akapitu może być określony tylko na przeciąg jednego akapitu).
7. **Uruchomienie mechanizmu przenoszenia elementów** z listy ostatnich przyczynków (ang. *recent contributions*) do bieżącej strony (ang. *current page*).

Wnioski:

1. \noindent\endgraf tworzy akapit 0-pudełkowy,
2. \noindent\hskip0pt\endgraf tworzy akapit 1-pudełkowy mimo, że klej \hskip0pt zostaje usunięty z listy poziomej przed rozpoczęciem łamania,
3. oczywiście \indent\endgraf tworzy akapit 1-pudełkowy.

Łamanie na linie. Opis algorytmu łamania na linie wykracza poza zakres niniejszej pracy.

Umieszczanie akapitu na liście pionowej. Podczas trzech faz tworzenia akapitu TEX umieszcza na liście pionowej pewne elementy:

1. **W momencie rozpoczęcia tworzenia akapitu** TEX umieszcza na liście pionowej klej wielkości \parskip. Klej ten jest umieszczany niezależnie od jego wielkości, tzn. również wtedy, gdy \parskip=0pt. Klej nie jest umieszczany, jeśli trybem, w którym następuje rozpoczęcie tworzenia akapitu jest tryb wewnętrzny pionowy i lista pionowa jest pusta.
2. **W trakcie tworzenia listy poziomej** żadne elementy nie są umieszczane na liście pionowej.
3. **Po zakończeniu tworzenia listy poziomej i złamaniu akapitu na linijki** [104_1] TEX umieszcza na liście pionowej:

3.1. ciąg pudełek (linijek) poprzedzonych klejami międzylinijkowymi i oddzielonych karami (kleje i kary obliczone według zasad określonych poniżej),
3.2. elementy migrujące z pudełek-linijek (elementy typu \vadjust, wstawki, znaczniki).

Po zakończeniu tworzenia listy poziomej i złamaniu akapitu na linijki powstaje ciąg linijek, które umieszczane są na liście pionowej wraz z dodatkowymi elementami według następujących zasad:

1. Jeżeli po zakończeniu tworzenia listy poziomej i złamaniu akapitu akapit jest 0-linijkowy, to żadne elementy nie są umieszczane. (Taki akapit powoduje jedynie umieszczenie kleju \parskip.)
2. Umieszczenie każdej linijki na liście pionowej polega na umieszczeniu na liście pionowej następujących elementów:

2.1. kleju międzylinijkowego; wielkość kleju obliczona jest na podstawie wartości parametrów \baselineskip, \lineskiplimit, \lineskip, \prevdepth obowiązujących w momencie wykonania komendy \par kończącej akapit; dokładne reguły obliczania kleju międzylinijkowego opisane są w [80^2], intuicyjne wyjaśnienia zawiera [78^2–80^1]; pierwsze trzy z wymienionych parametrów są stałe dla całego akapitu, parametr \prevdepth zmienia się po umieszczeniu każdej linijki (pudełka) [80^2];
2.2. tej linijki (pudełka);
2.3. elementów migrujących z tej linijki (elementy typu \vadjust, wstawki, znaczniki) w takiej kolejności, w jakiej znajdują się w linijce (jeśli takie elementy istnieją); między pudełkiem, a elementami migrującymi nie są umieszczane żadne dodatkowe elementy, w szczególności

żadne kary i kleje [105_5]; umieszczanie elementów migrujących nie zmienia wartości parametru `\prevdepth` (ćwiczenie 14.28 [105_4] oraz rozwiązanie [316_5]);

2.4. kary międzylinijkowej o wartości obliczonej według zasad opisanych w [104_1–105^1] kara nie jest umieszczana po ostatniej linijce akapitu oraz jeśli jej wartość jest równa 0 (rozwiązanie ćwiczenia 14.26, [316^4]).

◇ Marek Ryćko
marek@do.com.pl

Główna lista pionowa w systemie TEX

Marek Ryćko

Główna lista pionowa

Niniejszy tekst opisuje tę część mechanizmów poziomu trzeciego TEX-a (poziomu wykonywania komend), która związana jest z główną listą pionową. (Patrz również: [rozdz. 15, 109–115^2] oraz [125_1]).

Główna lista pionowa tworzona jest przez system TEX, a ściślej przez mechanizm realizujący komendy, w czasie jego pracy w (zwykłym) trybie pionowym. Elementy tej listy mogą być na nią wprowadzane bezpośrednio przez użytkownika za pomocą komend takich jak `\hbox`, `\vskip` czy `\penalty`. Również sam system, bez komendy służącej bezpośrednio do tego celu, może umieszczać na tej liście pewne elementy. Na przykład po zakończeniu składania akapitu i zamianie listy poziomej na ciąg linijek-pudełek TEX dodaje do listy pionowej te pudełka wraz z odpowiednimi klejami, karami i innymi elementami.

Nasz opis utrzymany jest na określonym poziomie abstrakcji. Nie zajmujemy się tu takimi aspektami pracy TEX-a, jak: zajętość pamięci i czas działania.

W poniższym opisie nie są opisane wszystkie komendy wpływające na postać listy pionowej i wszystkie ich skutki, ponieważ – ze względu na możliwości rekursywne TEX-a – musiałoby to oznaczać konieczność opisania praktycznie całego systemu. Nie opisujemy zatem w szczególności:

1. komend służących do konstruowania pudełka (pudełko jest jednym z elementów, które mogą się znaleźć na liście pionowej),
2. struktury wewnętrznej pudełka (czyli pojęcia pudełka jako typu danych).

Główna lista pionowa jako typ danych

[112^1]. Główna lista pionowa (będziemy mówić w skrócie „lista pionowa” lub nawet czasem „lista”) jest podzielonym na dwie części, skończonym ciągiem elementów:

$$x_1 \ldots x_k \mid x_{k+1} \ldots x_n.$$

Każda z części (lub obie) może być pusta ($k \geq 0$, $n \geq k$). Część pierwsza tego ciągu $x_1 \ldots x_k$ nazywa się *bieżącą stroną*, (ang. *current page*), a część druga $x_{k+1} \ldots x_n$ nazywa się *ostatnimi przyczynkami* (ang. *recent contributions*). Intuicyjnie ostatnie przyczynki, to te elementy, które już zostały dołożone do listy pionowej, ale nie zostały jeszcze umieszczone na bieżącej stronie.

Każdy z elementów x_i listy pionowej może być jednego z siedmiu możliwych typów [110^2]:

1. pudełko (ang. *box*) (poziome, pionowe lub linia (ang. *rule*)); każde pudełko może być dodatkowo przesunięte poziomo komendą `\moveleft` lub `\moveright`, co nie wpływa na jego wymiar pionowy),
2. wihajster (coś) (ang. *whatsit*),
3. znacznik (ang. *mark*),
4. wstawka (ang. *insertion*),
5. klej (ang. *glob of glue*) lub doprowadzenie (smycz, wykropkowanie?) (ang. *leaders*),
6. podcięcie (ang. *kern*),
7. kara (ang. *penalty*).

Uwagi:

1. Pudełko umieszczone na liście pionowej może być skonstruowane bezpośrednio jedną z komend: `\hbox`, `\vbox`, `\vtop`, `\box`, `\copy` lub `\hrule` lub pośrednio jako część listy pionowej pochodzącej bądź z rozpakowanego pudełka pionowego (komendy `\unvbox`, `\unvcopy`), bądź z połamanego na linijki akapitu. Z pudełkiem nie będącym linią (ang. *rule*) skojarzona jest informacja o jego wymiarach i zawartości, z pudełkiem typu `\hrule` – informacja o wymiarach.
2. klej lub smycz powstają w wyniku wykonania jednej z następujących dwu postaci komend:
   2.1. ⟨skip⟩
   2.2. `\leaders` ⟨box or rule⟩ ⟨skip⟩

   taki element niesie w sobie informację o typie wypełnienia (klej lub smycz), o wielkości kleju (⟨skip⟩) oraz w przypadku smyczy o pudełku lub linii, które podlega rozciąganiu lub powielaniu.
3. podcięcie niesie w sobie informację o związanej z nim odległości (⟨dimen⟩).

4. kara zawiera informację o liczbowej wielkości tej kary (dowolna liczba w sensie TEX-a)

Każdy z elementów niesie w sobie pewne dodatkowe informacje, np.

Klasyfikacja elementów. Warto zwrócić uwagę, że spośród wyżej wymienionych elementów niektóre nie mają wpływu na „wypełnienie strony" (ich wymiar pionowy jest – jakby – równy zeru). Te, które mają wymiar pionowy (być może, ale niekoniecznie, zerowy) to: pudełka, wstawki, kleje i podcięcia.

Elementy, które mogą pojawiać się na liście pionowej, są podzielone na dwa rodzaje [110^2]:

1. *elementy usuwalne* (ang. *discardable items*) (kleje, podcięcia i kary),
2. *elementy nieusuwalne* (ang. *nondiscardable items*) (wszystkie pozostałe).

Podział ten jest niezależny od miejsca zajmowanego przez element na liście.

Niektóre z elementów usuwalnych znajdujących się na liście pionowej określa się mianem *dozwolonych miejsc podziału na strony* (ang. *permissible breakpoints*) [110_2]. Uznanie elementu za dozwolone miejsce podziału na strony na ogół zależy od jego położenia na liście. Element x_i jest dozwolonym miejscem podziału jeśli jest:

a) klejem poprzedzonym bezpośrednio elementem nieusuwalnym,
b) podcięciem, po którym bezpośrednio następuje klej,
c) karą.

Warto zwrócić uwagę na różnice w traktowaniu klejów i podcięć znajdujących się na liście pionowej w zakresie ich wpływu na podział strony.

Aby łatwiej było zapamiętać reguły dotyczące dozwolonych miejsc podziału, sformułujmy je nieco inaczej.

1. Pojedyncze podcięcia ani ciągi podcięć oddzielające elementy nieusuwalne nie są dozwolonymi miejscami podziału. Należy pamiętać, że oprócz jawnie umieszczonych podcięć na liście pionowej mogą znaleźć się kleje umieszczane tam automatycznie, między innymi przed pudełkiem (ang. *interline skip*) oraz przed akapitem (klej typu \parskip). W szczególności klej wielkości 0pt nie jest równoważny brakowi kleju; pojawienie się takiego kleju po podcięciu powoduje zezwolenie na łamanie na tym podcięciu.
2. Klej jest dozwolonym miejscem podziału wtedy i tylko wtedy, gdy jest pierwszym z serii elementów usuwalnych.
3. Kara jest dozwolonym miejscem podziału niezależnie od jej położenia na liście względem innych elementów. Zgodnie z definicją kara nazywa się dozwolonym miejscem podziału nawet, jeśli wartość liczbowa z nią związana jest bardzo wysoka – większa od 10 000).

A oto jeszcze inny sposób myślenia o elementach nieusuwalnych.

Lista pionowa składa się z serii elementów nieusuwalnych na zmianę z seriami elementów usuwalnych. Wśród elementów usuwalnych niektóre mogą być dozwolonymi miejscami podziału:

1. Każda kara jest dozwolonym miejscem podziału.
2. Klej jest dozwolonym miejscem podziału, jeśli jest pierwszym elementem w serii elementów usuwalnych.
3. Podcięcie jest dozwolonym miejscem podziału jeśli bezpośrednio po nim następuje klej.

Wnioski

1. Podcięcie, będące ostatnim elementem w serii elementów usuwalnych nie jest dozwolonym miejscem podziału.
2. Jeśli seria (elementów usuwalnych) składa się wyłącznie z podcięć (jednego lub więcej), to żaden element tej serii nie jest dozwolonym miejscem podziału.
3. Jeśli seria składa się wyłącznie z podcięć i kar, to żadne podcięcie nie jest dozwolonym miejscem podziału (i każda kara jest).
4. W ciągu podcięć (będącym częścią serii elementów usuwalnych) tylko ostatnie ma szansę być dozwolonym miejscem podziału.
5. W ciągu klejów (będącym częścią serii elementów usuwalnych) tylko pierwszy ma szansę być dozwolonym miejscem podziału.
6. W ciągu kar (będącym częścią serii elementów usuwalnych) każda kara jest dozwolonym miejscem podziału.

Na zakończenie jeszcze inny sposób myślenia o elementach nieusuwalnych.

1. O kleju wiadomo, że jest dozwolonym miejscem podziału strony, jeżeli jest tuż po elemencie nieusuwalnym.
2. O karze wiadomo, że zawsze jest dozwolonym miejscem podziału strony.
3. O podcięciu nie wiadomo czy jest dozwolonym miejscem podziału strony, dopóki nie znamy

następnego elementu. Dopiero jeśli się okaże, że jest nim klej, to takie podcięcie staje się dozwolonym miejscem podziału strony.

Uwagi, wątpliwości.

1. Czy klej może być pierwszym elementem listy pionowej? Chyba tak – w części „ostatnich przyczynków", jeśli „bieżąca strona" jest pusta.
2. W jaki sposób TEX stwierdza, czy podcięcie jest dozwolonym miejscem podziału, jeśli na liście pionowej po nim nic nie następuje? Według powyższej definicji takie podcięcie **nie** jest dozwolonym miejscem podziału, ale może TEX czeka z ostateczną decyzją aż coś się pojawi.

Kary i koszty. Oprócz wspomnianych wyżej informacji zawartych w każdym elemencie listy pionowej, TEX w każdym stanie listy przypisuje niektórym jej elementom pewne wartości liczbowe. Te wartości to *kary* i *koszty*.

Dla każdego elementu x_i listy pionowej ($1 \leqslant i \leqslant n$), który jest dozwolonym miejscem podziału, określona jest wartość kary p_i. Jeśli x_i jest karą, to p_i jest równe wartości tej kary. Jeśli x_i jest klejem lub podcięciem, to $p_i = 0$.

Każdemu z elementów x_i listy pionowej leżących w części „bieżąca strona" ($1 \leqslant i \leqslant k$) przypisana jest liczba c_i zwana *kosztem* złamania w tym miejscu (ang. *cost*). Wartość c_i jest określona tylko dla elementów x_i będących dozwolonymi miejscami podziału. Jest to liczba całkowita z przedziału $[111_1]$:

$$c_i \in \{-\infty\} \cup [-100\,000, 100\,000] \cup \{+\infty\}.$$

Sposób obliczania c_i jest opisany w *TEXbook-u* na stronie 111_1. Wartość c_i zależy m. in. od elementów umieszczonych wcześniej na liście pionowej. Wartość kary $= \infty$ jest oznaczana w pliku transkrypcyjnym (pliku typu `log`, (ang. *log file*)) symbolem $*$.

Operacje wykonywane na głównej liście pionowej

W punkcie *Główna lista pionowa jako typ danych* opisaliśmy listę pionową jako typ danych, czyli jej strukturę wewnętrzną. W niniejszym rozdziale opisujemy operacje (czynności) służące do modyfikowania listy pionowej. Niektóre z tych operacji dostępne są bezpośrednio użytkownikowi, inne wykonuje TEX „samodzielnie" w ściśle określonych momentach.

Omawiając operacje modyfikujące listę pionową określimy w szczególności jakie elementy stanu wewnętrznego TEX-a wpływają na wykonywanie tych operacji oraz jakie elementy stanu wewnętrznego są zmieniane poprzez te operacje poza samą listą pionową.

System TEX wykonuje podczas pracy cztery rodzaje operacji na liście:

1. określenie (ustalenie) stanu początkowego,
2. dołączanie nowych elementów do części „ostatnie przyczynki",
3. przenoszenie elementów z „ostatnich przyczynków" do „bieżącej strony",
4. podział strony, uruchamiany w pewnych przypadkach po przeniesieniu elementu; w wyniku podziału strony pewna liczba elementów z początkowej cześci bieżącej strony może zostać usunięta.

W czasie wykonywania operacji przenoszenia TEX może uruchomić m. in. mechanizm podziału strony.

Stan początkowy listy. Na początku pracy TEX-a lista pionowa jest pusta, tzn. puste są: „bieżąca strona" i „ostatnie przyczynki" ($k = 0$, $n = 0$).

Dołączanie nowych elementów do listy „ostatnich przyczynków". Nowe elementy dołączane mogą być do listy „ostatnich przyczynków" w każdym momencie pracy TEX-a w trybie pionowym. Większość komend wykonywanych w tym trybie powoduje dołączenie pewnych elementów do tej listy. Na przykład napisanie w trybie pionowym komendy `\hbox{A}` powoduje (na ogół) umieszczenie na liście dwóch elementów: międzylinijkowego kleju (w ilości starannie obliczonej według odpowiednich zasad) oraz pudełka zawierającego literę „`A`". Przypomnijmy:

1. Klej międzylinijkowy (ang. *interline glue*) pojawia się zwykle automatycznie przed każdym pudełkiem umieszczanym na liście pionowej, w tym również przed każdą linijką utworzonego w trybie poziomym akapitu – patrz opis akapitu w poprzednim tekście. Ilość umieszczanego kleju jest sterowana ustalanymi przez użytkownika parametrami `\baselineskip`, `\lineskiplimit` i `\lineskip` na podstawie bieżącej wartości parametru `\prevdepth`.
2. Klej międzyakapitowy (ang. `\parskip` *glue*) umieszczany jest przez system TEX na liście pionowej automatycznie w każdym momencie przejścia TEX-a z trybu pionowego do trybu poziomego (patrz [MR '93]).

3. Kary mogą być umieszczane automatycznie lub ręcznie, np. TEX automatycznie wstawia po pierwszej linijce akapitu karę \widowpenalty, a przed ostatnią linijką akapitu karę \clubpenalty. Nazwy pochodzą od angielskich określeń *widow line* i *club line*. Ponadto przed każdą linijką akapitu, z wyjątkiem pierwszej, umieszczana jest kara \interlinepenalty.
4. Podcięcia (ang. *kerns*) mogą być umieszczane na liście pionowej tylko poprzez jawne zastosowanie komendy \kern przez użytkownika. Zwróćmy uwagę, że podcięcia na liście poziomej mogą pojawić się automatycznie.

Przenoszenie elementów z „ostatnich przyczynków" do „bieżącej strony". Żadne operacje TEX-a nie zmieniają bezpośrednio kolejności elementów wcześniej umieszczonych na głównej liście pionowej. Można to zrobić jedynie pośrednio umieszczając podczas działania procedury wyjściowej \output zawartość lub część zawartości pudełka 255 na końcu listy.

Stronami tekstu zapisywanymi w pliku dvi, a następnie drukowanymi, są pudełka posyłane tam komendą \shipout. Można jej użyć w dowolnym momencie pracy TEX-a. Zwykle strony-pudełka konstruowane są i posyłane w wyniku wykonania komend zapisanych w treści procedury wyjściowej (\output). Do utworzenia takiego pudełka procedura wyjściowa może wykorzystać tylko pewną część „bieżącej strony" będącej fragmentem głównej listy pionowej.

Jak opisaliśmy w punkcie *Dołączanie nowych elementów do listy „ostatnich przyczynków"* nowe elementy dokładane są do głównej listy pionowej tylko w jej części „ostatnie przyczynki". A zatem musi być inny sposób umieszczania elementów w części „bieżąca strona" głównej listy pionowej. Jedynym sposobem umieszczenia tam elementu jest wykonanie na tej liście operacji *przenoszenia elementów*. Można to sobie wyobrażać jako przesuwanie granicy pomiędzy „ostatnimi przyczynkami" i „bieżącą stroną". Użytkownik TEX-a nie ma bezpośredniego dostępu do operacji przenoszenia. TEX wykonuje tę operację jedynie automatycznie i jedynie w ściśle określonych sytuacjach. Dlatego niezbędna jest staranna wiedza o tym kiedy to następuje. Jednym ze skutków przenoszenia może być „odcięcie" z „bieżącej strony" pewnej liczby elementów przeznaczonych do zapisania w pliku dvi i uruchomienie procedury wyjściowej (\output). W szczególności z poniższych informacji wynika kiedy może być uruchomiona procedura wyjściowa.

TEX rozpoczyna przenoszenie elementów z listy „ostatnich przyczynków" do „bieżącej strony" dokładnie w następujących momentach [122²]:

a) na początku i końcu akapitu umieszczanego na liście pionowej głównej (patrz opis zmiany trybu na poziomy w *Co nieco o akapcie*),
b) na początku i końcu wyróżnionego równania (ang. *displayed equation*) w obrębie akapitu umieszczanego na liście pionowej głównej,
c) po zakończeniu wykonywania komendy \halign w trybie zwykłym pionowym,
d) po (jawnym) dołożeniu do głównej listy pionowej (do części „ostatnich przyczynków") pudełka, wstawki lub kary, tj. użycie w trybie pionowym jednej z komend \hbox, \vbox, \vtop, (również \box0, \copy17 itp.), \hrule \insert, \penalty. (również chyba po wymigrowanym \insert [281¹])
e) po zakończeniu działania procedury wyjściowej (ang. \output *routine*).

Wynika stąd w szczególności, że dołożenie do głównej listy pionowej kleju lub podcięcia **nie** powoduje „przenoszenia", a zatem nie może uruchomić procedury wyjściowej.

Przenoszenie pojedynczych elementów Poniżej opisujemy wykonywanie przez system TEX operacji przeniesienia pojedynczego elementu z listy „ostatnich przyczynków" do „bieżącej strony". Przypomnijmy, że elementy tej listy oznaczamy przez:

$$x_1 \ldots x_k \mid x_{k+1} \ldots x_n, \quad (k \geq 0,\ n \geq k).$$

Każdemu usuwalnemu elementowi x_i tej listy przypisana jest kara p_i. Każdemu dozwolonemu miejscu podziału na strony x_i (w części „bieżąca strona", tj. dla $1 \leqslant i \leqslant k$) przypisany jest koszt c_i. Przeniesienie elementu x_{k+1} do „bieżącej strony" przebiega w sposób następujący:

1. **Przeniesienie elementu nieusuwalnego.** Jeżeli x_{k+1} jest elementem *nieusuwalnym* (a zatem nie może być dozwolonym miejscem podziału), to po prostu dołączany jest na końcu bieżącej strony i usuwany z „ostatnich przyczynków". Po przeniesieniu stan listy pionowej jest następujący: $x_1 \ldots x_k x_{k+1} \mid x_{k+2} \ldots x_n$.

2. **Postępowanie z elementem usuwalnym.** Jeżeli x_{k+1} jest elementem *usuwalnym* (klejem, podcięciem lub karą), to:
   2.1. **Warunkowe usunięcie elementu usuwalnego.** Jeśli wśród elementów $x_1 \ldots x_k$ nie ma żadnego pudełka, to x_{k+1} jest **usuwany** (stąd nazwa *usuwalny*) z części „ostatnie przyczynki" i **nie jest** umieszczany na „bieżącej stronie". Po tej operacji stan listy pionowej jest następujący: $x_1 \ldots x_k \mid x_{k+2} \ldots x_n$.
   2.2. **Warunkowe przeniesienie elementu usuwalnego.** Jeśli wśród elementów $x_1 \ldots x_k$ jest co najmniej jedno pudełko, to element x_{k+1} jest dołączany na końcu „bieżącej strony" i usuwany z „ostatnich przyczynków". Po wykonaniu tej operacji bieżąca strona składa się z elementów $x_1 \ldots x_{k+1}$, a ostatnie przyczynki z elementów $x_{k+2} \ldots x_n$ (w szczególności ten ostatni ciąg może być pusty).
   2.2.1. Jeżeli nowo dołączony do bieżącej strony element x_{k+1} **nie** jest dozwolonym miejscem podziału (tj. jest klejem po elemencie usuwalnym lub podcięciem po którym nie ma kleju), to przenoszenie elementu na tym się kończy.
   2.2.2. **Obliczenie kosztu i ewentualne rozpoczęcie podziału strony.** Jeśli element x_{k+1} jest dozwolonym miejscem podziału (przypomnijmy, że ta cecha elementu zależy od położenia elementu na liście), to obliczany jest dla niego koszt c_{k+1} złamania w tym punkcie Następnie TEX rozpatruje wartości c_{k+1} i p_{k+1} (koszt i karę). Jeśli $c_{k+1} = \infty$ lub $p_{k+1} \leqslant -10\,000$ (nieskończenie duży koszt lub ogromna nagroda), to TEX przystępuje do **podziału strony** „na elemencie x_i" $(1 \leqslant i \leqslant k+1)$ takim, że c_i jest minimalny spośród $c_1 \ldots c_{k+1}$. Jeśli jest więcej niż jeden element o minimalnym koszcie, to wybiera się najpóźniejszy. Algorytm podziału strony jest opisany poniżej.

Po zakończeniu przenoszenia (z ew. wielokrotnym uruchomieniem procedury wyjściowej i ponownym uruchomieniem procesu przenoszenia) TEX powraca do wykonywanej uprzednio czynności.

Pytanie. Czy może nastąpić podział strony na takim elemencie x_{k+1}, dla którego koszt podziału jest nieskończony, tj. $c_{k+1} = \infty$.

Odpowiedź. Tak, jeżeli wśród $c_1, \ldots, c_{k+1}$ nie ma mniejszego kosztu, w szczególności jeśli x_{k+1} jest pierwszym dopuszczalnym miejscem podziału.

Pytanie. Czy użycie PLAIN TEX-owego makra `\medskip` może spowodować uruchomienie mechanizmu podziału strony?

Odpowiedź. Makro `\medskip` rozwija się do `\vskip\medskipamount`. W przypadku użycia tej komendy w trybie poziomym TEX rozpoczyna jej wykonywanie od zakończenia bieżącego akapitu, a ściślej mówiąc od realizacji bieżącego znaczenia komendy `\par`, co zwykle prowadzi do zakończenia składania akapitu. Powoduje to m.in. uruchomienie mechanizmu przenoszenia elementów, co może spowodować podział strony. Użycie komendy `\medskip` w trybie pionowym powoduje tylko dopisanie kleju do listy ostatnich przyczynków i nie może powodować podziału strony.

Pytanie. Czy możliwa jest taka sytuacja, w której użycie komendy `\penalty10000` (kara zakazująca podziału) spowoduje przystąpienie TEX-a do podziału strony. Ściślej mówiąc, czy możliwe jest, że w sutuacji, kiedy pionowa lista składa się z elementów $x_1 x_2 \ldots x_n$, jawne użycie kary `\penalty10000` doprowadzi do uruchomienia mechanizmu podziału strony.

Odpowiedź. Tak. Przykładem takiej sytuacji może być lista pionowa $x_1 \ldots x_{n-1} \mid x_n$, gdzie x_{n-1} jest pudełkiem, x_n jest klejem (dołożenie kleju nie uruchamia przenoszenia), koszt złamania na x_n jest ∞ (pudełko x_{n-1} „wystaje" poza stronę). Dołożenie kary powoduje przeniesienie kleju x_n i złamanie strony na najpóźniejszym z miejsc o minimalnym koszcie, być może na x_n.

Podział strony. TEX inicjuje akcję tzw. *podziału strony* w pewnych przypadkach po przeniesieniu elementu z listy ostatnich przyczynków do listy zwanej „bieżącą stroną", jak opisano powyżej. Sytuacja, w której następuje podział strony jest następująca. Lista pionowa składa się z bieżącej strony $x_1 \ldots x_{k+1}$, która jest niepusta (ostatni element jest dozwolonym miejscem podziału) oraz z – być może pustej – listy ostatnich przyczynków. Wśród kosztów $c_1 \ldots c_{k+1}$ przypisanych dozwolonym miejscom podziału $x_1 \ldots x_{k+1}$ koszt c_i $(1 \leqslant i \leqslant k+1)$ jest jedynym minimalnym lub najpóźniejszym z minimalnych.

Podziału dokonuje się **na** elemencie x_i w następujący sposób:

1. Elementy $x_1 \ldots x_{i-1}$ umieszczane są w pionowym pudełku o numerze 255 – `\box255` (z odpowiednim zamrożeniem klejów, ale z możliwością

ich rozmrożenia w przypadku rozpakowania pudełka). Nie dotyczy to wstawek (ang. *insertions*) znajdujących się w ciągu $x_1 \dots x_{i-1}$, których zawartość umieszczana jest w osobnych pudełkach [125₁].

2. Ciąg elementów $x_i \dots x_{k+1}$ znajdujących się dotychczas w części „bieżąca strona" przenoszony jest na początek ciągu $x_{k+2} \dots x_n$ ostatnich przyczynków [125_1]. Obliczone wcześniej koszty $c_i \dots c_{k+1}$ są zapominane; będą musiały być obliczone ponownie w przypadku powrotu tych elementów do bieżącej strony. Wśród przyczyn konieczności ich ponownego obliczenia należy wymienić możliwość dokonania przez użytkownika przed kolejną akcją podziału strony zmiany parametrów takich jak `\topskip` czy `\vsize`, które wpływają na obliczane koszty.
   - 2.1. Jeśli element x_i jest karą o wartości p_i, to parametrowi `\outputpenalty` przypisuje się wartość p_i, a karę x_i zamienia się na karę o wartości 10 000.
   - 2.2. Jeśli element x_i nie jest karą (jest klejem lub podcięciem), to parametr `\outputpenalty` otrzymuje wartość równą zeru.
3. Uruchamiana jest procedura wyjściowa. Mechanizmy rozwijania makr i wykonywania komend czytają i wykonują komendy umieszczone w ciągu żetonów `\output`. Automatycznie dodawane są na początku i na końcu symbole początku i końca grupy. Komendy wykonywane są w trybie wewnętrznym pionowym; tworzona więc jest dodatkowa lista pionowa. Komendy umieszczone przez użytkownika w procedurze wyjściowej (`\output`) muszą być tak skonstruowane, żeby po zakończeniu działania tej procedury pudełko o numerze 255 (`\box255`) było puste (ang. *void*). Po zakończeniu pracy procedury wyjściowej pozostaje pewna liczba nie wykorzystanych wstawek w pudełkach, w których były przechowywane oraz zbudowana lista pionowa. Elementy te są dołączane na początku listy ostatnich przyczynków. Po zakończeniu tego etapu lista przyczynków zawiera więc kolejno następujące elementy:
   - 3.1. niewykorzystane przez procedurę wyjściową wstawki,
   - 3.2. zbudowaną w czasie działania procedury wyjściowej listę pionową,
   - 3.3. dołączone do listy przyczynków elementy $x_i \dots x_{k+1}$, które przed rozpoczęciem podziału strony były na „bieżącej stronie" (pamiętamy, że jeśli x_i jest karą, to jej wartość mogła ulec zmianie),
   - 3.4. dotychczasowe elementy listy przyczynków $x_{k+2} \dots x_n$ (ten ciąg może być pusty).

   Po zakończeniu podziału strony lista pionowa zawiera co najmniej jeden element. Jest to usuwalny element x_i w części „ostatnich przyczynków".
4. [122^2] Uruchomienie akcji przenoszenia elementów z ostatnich przyczynków do bieżącej strony (wzajemna rekursja ogonowa), co może spowodować ponowne uruchomienie procedury wyjściowej.

Tworzenie dodatkowej listy pionowej w czasie działania procedury wyjściowej umożliwia m. in. umieszczenie na początku każdej strony dowolnych elementów, np. umieszczenie tam pustego pudełka spowoduje, że kleje nie będą znikały na górze strony. Takie proste! Ogólnie można umieścić co się chce – zostanie to potem przeniesione na bieżącą stronę i jako jej składnik umieszczone w pudełku o numerze 255 (`\box255`). Pozwala to na wpływanie na postać drukowanej strony inaczej niż poprzez dobudowywanie elementów do pudełka 255, tak jak to się dzieje np. w PLAIN TEX-u.

Opisany powyżej mechanizm podziału strony jest w TEX-u jedynym sposobem uruchamiania procedury wyjściowej. Nie jest to jednak jedyny sposób pisania do pliku `dvi`. Każde pudełko można zapisać w tym pliku przez zastosowanie w dowolnym momencie komendy `\shipout`⟨box⟩.

Możliwe jest takie zorganizowanie współpracy procedury wyjściowej i mechanizmów związanych z tworzeniem listy pionowej, że na liście pionowej (w jej części „ostatnie przyczynki") będzie znajdować się ilość materiału pionowego wystarczająca na kilka stron. Warto dobrze znać różne subtelności działania algorytmów obsługujących listę pionową i podział na strony, by móc w pełni korzystać z niezwykłej mocy TEX-a.

Prace cytowane

1. Donald E. Knuth, *The TEXbook*, Addison Wesley Publishing Company
2. Marek Ryćko, *Co nieco o akapicie*, Bachotek 1993

◇ Marek Ryćko
marek@do.com.pl