УДК 004.424.4:025.135(043.2)

[1]Мелешко М.А., к.т.н.,
[2]Соломін А.В., к.ф.-м.н.,
[1]Таран В.М.,
[1]Ракицький В.А.

ВИКОРИСТАННЯ КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ В СУЧАСНИХ ДОДРУКАРСЬКИХ ПРОЦЕСАХ ВИДАВНИЦТВ

[1]Національний авіаційний університет
[2]НТУУкраїни «Київський політехнічний інститут»

mma.nau@ukr.net
andr-sol@i.ua
vi_taran@mail.ru
Rvadim4835@gmail.com

Показана актуальність вивчення та використання сучасних програмних продуктів у видавничо-поліграфічній справі як напрям подальшого розвитку галузі на основі стандартизації технологічних видавничих процесів з використанням комп'ютерних засобів

Ключові слова: PostScript, PDF, JDF, формати для поліграфії: PDF/X-1, PDF/X-3, стандарт ISO 32000

Вступ

Успішна реалізація проектів комп'ютеризації, уніфікації та стандартизації технологічних процесів у видавничо-поліграфічній справі та приєднання до вирішення цієї проблеми провідних виробників комп'ютерного обладнання та програмного забезпечення склались всі умови для автоматизованої інтеграції всіх процесів, що повинні враховуватись в індустрії видавництв та поліграфії. Забезпечення інтеграції і автоматизації технологічних процесів в поліграфії та комп'ютерне керування робочим потоком (Workflow) здійснюється на основі JDF (Job Definition Format).

Постановка задачі

Слід звернути увагу на те, що JDF – є надбудовою над форматом PDF, побудованого на базі PostScript, виконує роль стандарту у видавництві. Важливо свідомо коректне впровадження в навчальний процес та наукові дослідження і застосування PostScript і PDF, як бази для подальшого розвитку галузі [1].

PostScript є і форматом файлів і повноцінною мовою програмування, що оптимізована для виконання функцій відображення зображень і текстів на папері, плівці, друкарських формах чи дисплеї. Віднині роль PostScript і PDF значно розширились. Тепер не тільки додрукарська підготовка з використанням комп'ютерних засобів, а і всі стадії видавничого процесу пов'язані з JDF, а отже з PDF і PostScript [1-3]. Тому і уваги до цієї тематики повинно приділятись значно більше.

Метою даної публікації є розкриття ідеології створення і використання PostScript і PDF, стисло, наскільки дозволяє обсяг статті, розкрити основні ідеї, закладені в них.

Основна частина

Спочатку трохи історії.

PostScript був розроблений Джоном Уорноком (John Warnock) і Чаком Гешко (Chuck Geschke) з Adobe Systems на початку 80-х рр. Спочатку їх метою було створення спеціалізованої робочої станції для друку, яка мала називатися PostScript, але незабаром з'ясувалось, що краще сконцентрувати зусилля на розробці засобів для керування принтерами сторонніх виробників.

У 1984 році побачив світ PostScript (пізніше до назви було додано Level 1, щоб відрізняти його від наступних версій). PostScript (Level 1) мав ряд переваг перед іншими системами того часу:

- платформонезалежність. Один і той же файл міг друкуватися як на лазерному принтері, який видавав тоді 300 dpi, так і на фотонабірному пристрої з 2400 dpi з найкращою якістю в кожному випадку;

- будь-який виробник міг ліцензувати інтерпретатор PostScript та використовувати PostScript зі своїм пристроєм;

- специфікації PostScript були загальнодоступні, таким чином, будь-який розробник міг писати програми, що підтримують PostScript.

Компанія Adobe ризикувала, випускаючи PostScript, і, можливо, їй не вдалося б переконати ринок у необхідності такого ресурсу, якби не Apple Computer. У 1985 році продажі комп'ютерів Macintosh почали падати, і компанії Apple потрібно було щось, що міг би тільки її комп'ютер. Apple Computer інвестував 2,5 мільйона доларів в Adobe, яка створила PostScript-контролер для принтера Apple LaserWriter, і в Aldus, що створила програму PageMaker, використавши всі можливості Macintosh і LaserWriter. Революційна тоді можливість якісної та зручної додрукарської підготовки на комп'ютері врятувала Apple і перетворила Adobe та Aldus у великі компанії. Інші виробники фотонабірної апаратури, починаючи з Linotype, оцінили PostScript і незабаром оснастили свою фотонабірну апаратуру інтерпретаторами PostScript. Згодом PostScript став стандартом в галузі додрукарської підготовки.

У 1991 Adobe випустила наступну версію PostScript – PostScript Level 2. Це була значна переробка, давно очікувана видавничим співтовариством.

Основними поліпшеннями були:

- збільшена швидкість і надійність (в основному це торкнулося управління пам'яттю);

- підтримка кольороподілу в самому контролері;

- розпакування компресованих зображень в самому контролері (JPEG та CCITT group 4);

- підтримка великих наборів шрифтів (для азіатських країн);

- кешування шрифтів і зображень;

- поліпшені драйвери;

- поліпшені алгоритми друку зображення (Accurate Screening).

У 1998 році Adobe ввела новий стандарт PostScript 3. Зміни в порівнянні з Level 2 не такі суттєві.

Основні переваги PostScript 3:

- підтримка 4096 рівнів на один колір (12-біт на колір, до того було 256 відтінків – 8 біт на колір), повна підтримка пантонних (spot) кольорів та прозоростей в зображеннях;

- підтримка PDF (контролери PS Level 3 можуть виводити PDF поряд з файлами PS Level 2);

- розширена підтримка кольороподілу (контролери PS 3 можуть здійснювати кольороподіл зображень, заданих у відтінках сірого або в шести кольорах);

- добавлена інтернетфункціональність (Web-ready printing).

PostScript – більше, ніж типова мова управління принтером, він є повнофункціональною мовою програмування. Хоча програми на PostScript і створюються в основному не людьми, а іншими програмами, в принципі ніщо не заважає писати на ньому програми для обрахунку графіки, реалізації чисельних методів розв'язування математичних задач тощо.

Багато прикладних програм можуть перетворити документ в PostScript-програму, при виконанні якої буде отриманий початковий документ. Ця програма може бути надіслана безпосередньо на принтер з підтримкою PostScript або перетворена інтерпретатором PostScript у інший формат (для принтерів без підтримки PostScript), або результат її виконання інтерпретатором може бути показаний на екрані. Оскільки вхідна PostScript-програма одна і та ж, PostScript називається незалежним від пристрою. Якщо прискіпливо придивитись до таких програмних продуктів, як Illustrator або QuarkXPress, то можна помітити, що по суті вони являють собою всього лише зручні графічні інтерфейси до PostScript, тобто, наприклад, коли дизайнер малює

графічними інструментами, Illustrator в цей час автоматично пише програми PostScript у фоновому режимі. В цьому можна переконатись, якщо відкрити в текстовому редакторі файл, створений в програмі Illustrator або QuarkXPress. Хто знає команди PostScript, побачить, що навіть пункти меню цих пакетів співзвучні з командами PostScript.

Більшість високопродуктивних принтерів і плотерів мають вбудований інтерпретатор мови PostScript. У той же час, прості принтери домашнього класу підтримують тільки елементарні графічні операції, тому завдання створення растрового зображення покладається на центральний процесор. Існують інтерпретатори мови PostScript для різних операційних систем.

PostScript-описи сторінок (програми) можуть генеруватися, переміщатися і інтерпретуватися у двох виявах: у вигляді ASCII тексту, тобто за допомогою звичайних друкованих символів, і у вигляді binary (двійкового) коду, тобто у вигляді потоку бітів. У першому випадку програми наочні і зрозумілі, більш універсальні по відношенню до різних комп'ютерних платформ, операційних систем і каналів передачі. У другому випадку – вище швидкість передачі та інтерпретації.

Власне PostScript – типова мова інтерпретатора, що базується на стеках аналогічно звичайному калькулятору. Стек, як відомо, – це всього лише область пам'яті, організована за принципом «останнім увійшов –першим вийшов». Основна перевага полягає у високій швидкості запису та зчитування даних, що пояснюється фактично відсутністю явної адресації комірок пам'яті, тобто явне вказування адреси замінюється порядком вкладання даних у стек і, відповідно, порядком вилучення зі стека.

PostScript оперує з 4-ма основними стеками, що втілюють в кожен момент стан середовища програми.

Стек операндів – це стек, куди вкладаються аргументи (операнди) PS-операторів до їх виконання і звідки беруться результати після виконання операторів.

Наприклад, припустимо, ми хочемо перемножити 14 і 135. Ми повинні використовувати наступний PS-код:

14 135 mul

Перші два слова «14» і «135» поміщають числа 14 і 135 в стек операндів. «Mul» викликає оператор множення, який витягує два верхніх, тобто поміщених в стек останніми числа зі стека операндів (числа 14 і 135), перемножує їх і поміщає результат в той же стек знову ж таки на його вершину. Результат може залишатися там для використання іншими операторами пізніше в програмі. Таким чином, у загальному вигляді синтаксис операторів наступний:

arg-1 arg-2. . . operator result

Ця конструкція означає, що перш, ніж викликати оператор "operator", необхідно помістити аргументи "arg-1", "arg-2" і т.д. в стек операндів.

Якщо результат не повертається, використовується риска "-".

Стек словників (dictionary). Перш за все про словники. Словник – це набір пар "ім'я (key)-значення". Всі іменовані змінні зберігаються в словниках разом зі своїми значеннями. Також всі допустимі оператори мови записані в словниках разом зі своїми кодами, що мають виконуватись. Слід зазначити, що в мові немає зарезервованих слів, зокрема, і імен вбудованих операторів. Тобто, PS-програма в принципі може змінити імена будь-яких операторів або додати нові.

Стек словників представляє собою стек всіх відкритих у поточний момент словників. Коли в програмі зустрічається будь-яке ім'я (key), інтерпретатор шукає першу появу такого імені у всіх згаданих в цьому стеку словниках, починаючи з вершини стека.

Таким чином імена асоціюються зі змінними, операторами і процедурами. Словник, що знаходиться на вершині стека словників, називається поточним. Словнику також може бути присвоєно ім'я, і

така пара може бути поміщена в інший словник.

Відзначимо, що в стеку словників постійно знаходяться наступні словники:

- Systemdict – системний словник з атрибутом "тільки читання", в якому містяться імена всіх PS-операторів та їх значення (коди), тобто їх дії;

- Userdict – словник користувача з атрибутом "доступний для запису", що містить зазвичай значення за замовчуванням змінних, які використовують PS-програми.

Суттєво, що userdict знаходиться на вершині всіх постійно розташованих в стеку словників. Створюючий нові процедури оператор def поміщає визначення процедур саме сюди. Розташування userdict на вершині стека забезпечує зазначену раніше можливість перевизначити при бажанні будь-який оператор PostScript, хоча словник systemdict, що містить цей оператор, має атрибут "тільки читання". Ця можливість пояснюється тим, що інтерпретатор шукає ім'я, починаючи з вершини стека словників, і якщо ім'я оператора знайдеться в словнику userdict, то далі пошук не проводиться.

Різні програми можуть створювати свої словники, які будуть розташовуватися в стеку словників вище словника userdict. Ці програми повинні стежити за порядком, тобто по завершенні видаляти їх із стека, інакше може порушитися порядок роботи PS-програм, які вважають, що userdict знаходиться на вершині.

Постійні словники видалити з стека неможливо.

Стек виконання (execution) містить об'єкти, що виконуються, наприклад, процедури, в стадії виконання. Процедури, що потребують виконання, спочатку вкладаються у цей стек, потім поелементно виконуються, а потім видаляються зі стека. Як тільки інтерпретатор змушений відкласти виконання будь-якої процедури, щоб почати нову, він поміщає нову на вершину стека, виконує її, а після завершення видаляє з стека, таким чином відкладена процедура знову опиняється на вершині стека і виконується далі.

Стек станів графічної системи (graphics state). Цей стек містить копії значень спеціальної структури даних, званої «поточний стан графічної системи» (graphics state), в якій зберігається інформація про поточний колір заповнення графічних об'єктів, поточну товщину лінії малювання, поточний шрифт, поточний шлях, поточну трансформаційну матрицю тощо. Більшістю цих параметрів окремо керують відповідні оператори мови PostScript, а стек служить для тимчасового зберігання всієї сукупності таких параметрів за допомогою оператора gsave і наступного відновлення в якості поточного стану за допомогою оператора grestore.

Ідея функціонування PS-інтерпретатора полягає в наступному. На вхід інтерпретатора надходить потік символів (в ASCII-кодуванні) або потік біт (у binary-кодуванні), які представляють собою PS-опис сторінок. Інтерпретатор сканує цей потік, розділяє на об'єкти відповідно до синтаксису мови PostScript, визначає їх тип і залежно від типу реагує, а саме: об'єкти типу «дані» (числа, строкові дані і т.д.) інтерпретатор просто вкладає на вершину стека операндів; об'єкти типу «ім'я» (іменовані об'єкти) інтерпретатор шукає в словниках (dictionary), відкритих і згаданих в даний момент в стеку словників і у відповідному порядку згадування в стеку. Знайдене в словнику значення виконується, якщо це оператор, процедура, або потрапляє у стек операндів, якщо це значення нездійсненне.

Розглянемо ряд базових понять мови та їх використання.

Простір вивідного пристрою (device space). Це система координат, що «розуміється» вивідним пристроєм, одиниця вимірювання яких зазвичай відповідає роздільній здатності пристрою. PostScript-програми зазвичай ніколи цей простір не використовують, за винятком останнього етапу – виведення.

Простір користувача (user space). Це координатна система, яка використову-

ється PS-програмами для опису розташування точок і ліній. По суті, це те ж саме, що перший квадрант звичайної прямокутної системи координат з початком відліку (точка (0,0)) у лівому нижньому кутку. Координати виражаються дійсними числами, тому поняття роздільної здатності в цьому просторі не має сенсу. Інтерпретатор з використанням відповідного PS-драйвера автоматично конвертує простір користувача в простір вивідного пристрою.

Поточна трансформаційна матриця (current transformation matrix, CTM). Перетворення координат простору користувача в координати простору вивідного пристрою здійснюється через поточну трансформаційну матрицю. Це матриця розмірності три на три, що дозволяє користувачеві обертати, масштабувати та переміщувати весь простір користувача всередині простору вивідного пристрою.

Поточна трансформаційна матриця є одним з елементів структури даних «поточний стан графічної системи (graphics state)».

Шлях (path). Це сукупність відрізків прямих ліній і сегментів кривих, розташованих на сторінці. Можна сказати, що шлях уявний, тобто він не обумовлює наявність конкретної фарби на папері принтера, а лише описує уявні лінії і криві на сторінці. Для того, щоб шлях став видимою намальованою принтером на папері лінією, існує оператор stroke. При цьому товщина, колір, тип лінії буде визначатися відповідними параметрами, що містяться в структурі даних graphics state (поточний стан графічної системи) в конкретний момент. Якщо шлях замкнутий, то обмежену їм область сторінки можна залити (зафарбувати) певною фарбою за допомогою оператора fill або вирізати, тобто зробити невидимими виступаючі за цей шлях частини зображень, за допомогою оператора clip. Відзначимо, що при заливці області за допомогою оператора fill колір, щільність фарби визначається знову ж таки відповідними параметрами в структурі graphics state.

Тепер звернемося до синтаксису опису об'єктів мови і його основних концепцій.

Розгляд проводимо для ASCII-кодування, тому що binary-кодування використовується зазвичай лише у разі машинної генерації коду для зв'язку між частинами якого-небудь програмно-апаратного комплексу без втручання оператора.

Звичайний набір символів для PS-програм в ASCII-кодуванні складається з друкованої підмножини символів ASCII-набору плюс розділові коди: «пробіл» (SP), «табуляція» (tab), «переведення рядка» (LF), «переведення формату» (FF), «повернення каретки» (CR), «Null» (nul). Всі перераховані вище коди розділових знаків трактуються PS-інтерпретатором однаково, як «пробіл», за винятком випадків, коли вони використовуються в коментарях або в строкових даних. У таких випадках вони реалізують своє пряме призначення, тобто повернення каретки, переведення рядка і т.д.

Символ пробілу (у зазначеному вище широкому сенсі слова) розділяє в PS-програмах одну синтаксичну конструкцію, наприклад, ім'я або число, від іншої. Будь-яка кількість підряд один за одним символів пробілу (в широкому сенсі слова) трактується як один символ пробілу. Символи (,), <, >, [,], {, }, /, % – спеціальні. Вони позначають (обмежують) такі синтаксичні об'єкти, як рядки, процедури, імена, коментарі. Будь-який з цих символів примусово завершує попередній об'єкт, при цьому не включається до нього.

Коментар. Будь-яка поява символу % поза об'єктом типу «рядок» означає початок коментаря. Коментар складається з усіх будь-якого типу символів між символом % і одним з символів: «переведення рядка» (LF), «повернення каретки» (CR), «переведення формату» (FF). Звертаємо увагу, що у разі об'єкта типу «коментар» символи «пробіл» (SP) і «табуляція» (tab) не означають закінчення коментаря.

Інтерпретатор ігнорує коментарі, трактуючи кожен з них як один пробіл.

Відзначимо попутно, що у зв'язку з таким трактуванням в принципі коментар може використовуватися як роздільник між PS-об'єктами замість пробілу, табуляції, повернення каретки і т.д.

Основне призначення коментаря в будь-якій мові – це полегшити розуміння програми, тобто це короткі пояснення до конкретних об'єктів, програмних модулів, які один програміст пише, щоб легше було розібратися іншому. Саме у зв'язку з цим інтерпретатор ігнорує коментарі, – вони призначені людині, а не інтерпретатору.

Однак, у випадку PostScript коментарі грають ще одну, куди більш важливу роль. Через коментарі спеціального типу передається інформація програмним надбудовам, що працюють над PS. Наприклад, у вигляді коментаря міститься інформація про розташування сторінок у виданні. Це дуже гарна ідея – розділити сфери функціонування PS-інтерпретатора і надбудов над PS. Інтерпретатор ігнорує коментарі, а надбудови саме з ними і працюють.

З інформацією, що міститься в коментарях, тісно пов'язані такі сучасні формати та об'єкти, як job ticket, CIP4, pdf, на базі яких будуються всі сучасні системи work flow.

Слід відзначити ще одну важливу функцію коментаря. Спеціальним коментарем %! починається будь-яка PS-програма. Ці два символи можуть розглядатися як маркер, який позначає, що наступний файл є PS-кодом. Цей коментар розпізнається будь-яким PS-інтерпретатором або PS-принтером.

Числа (number) в PS-програмі використовуються в форматах, звичайних для будь-яких програм. PostScript працює з цілими та дійсними числами.

Рядок (string). Рядок – це саме рядок символів. Для позначення об'єктів типу «рядок» використовуються наступні два варіанти:

1) послідовність символів, укладена в дужки, наприклад, (це рядок);

2) послідовність шістнадцятирічних кодів символів, укладена в кутові дужки.

Ім'я (name). Ім'я – це будь-яка послідовність символів, яка не може інтерпретуватися як число. Ім'я є по суті посиланням до деякого значення у словнику з числа згаданих в стеку словників. Словник, як згадувалося вище, – це набір пар "ім'я-значення".

Символ / (slash прямий), що передує імені, визначає ім'я як літеральне (нездійсненне). Сам символ / не є частиною імені, а лише префіксом, що вказує на те, що наступне за ним ім'я є літеральне. Ім'я без символу / є здійсненним.

Всі об'єкти поділяються на літеральні або здійсненні. Це є одним з атрибутів об'єкта. Літеральні об'єкти інтерпретатор трактує просто як дані і поміщає в стек операндів для використання наступними операторами. Здійсненні об'єкти інтерпретатор намагається виконати. Манера виконання залежить від типу об'єкта. Наприклад, для цілих чисел виконання полягає в переміщенні об'єкта в стек операндів, тобто тут відмінності неістотні. Для здійсненного імені виконання полягає в пошуку в словниках відповідного імені значення. Якщо це значення – не оператор і не процедура, воно знову ж таки потрапляє у стек операндів. Для здійсненного оператора і здійсненної процедури виконання полягає у вчиненні деяких, зазначених у них, дій.

Числові і рядкові константи – завжди літеральні об'єкти.

Імена – літеральні, якщо їм передує символ /, і здійсненні в іншому випадку.

Масив (array). Масив у PostScript аналогічний масиву у будь-якій іншій мові програмування. Масиви можуть містити об'єкти різних типів і відображаються як послідовність об'єктів, розділених, як усюди в PS, пробілами, і укладена в квадратні дужки. Наприклад, [14 /Let 5.1E2] представляє собою 3-елементний масив, що містить число 14, літеральне ім'я Let і число 5.1E2. Елементи масиву доступні за допомогою індексу (від 0 до n-1, де n – розмірність масиву).

Слід зауважити, що ліва і права квадратні дужки ([ ]) насправді є іменами, які при виконанні викликають відповідні PS-оператори, що збирають відповідні об'єкти і створюють з них новий об'єкт – масив.

Звернемо також увагу на те, що будь-які здійснимі вирази всередині масиву при його створенні виконуються, тобто створений масив містить вже обчислені значення, а не вирази для їх обчислення.

Процедура (procedure). Процедура – це можливість для користувачів створювати нові оператори.

Процедура – це масив (array), який є здійсненним, і відповідно позначається фігурними дужками, а не квадратними. Тут дужки також є іменами PS-операторів, які створюють масив з розміщених між ними об'єктів, однак, у цьому випадку жодна процедура і жоден оператор, у т.ч. і вкладені, відразу не виконується. Вони називаються відкладеними (deferred). Інтерпретатор не виконує негайно ні самої процедури, ані елементів цієї процедури, якщо зустрічає процедуру безпосередньо. Замість виконання він трактує процедуру як дані (літеральний об'єкт) і поміщає в стек операндів. Виконання здійснюється лише при подальшому непрямому зверненні до цієї процедури через ім'я або з іншої процедури, оператора.

Наприклад, процедура зведення в квадрат верхнього елемента стеку операндів може бути записана наступним чином: {dup mul}. Тут оператор dup дублює, тобто створює копію верхнього елемента стеку, а оператор mul перемножує два верхніх елемента стеку, які в даному випадку, після виконання оператора dup, є тотожними. У підсумку результат міститься на вершині стеку.

Ми можемо далі визначити за допомогою цієї процедури оператор піднесення в квадрат наступним виразом:

/ Square {dup mul} def.

Тут square – це вигадане нами ім'я такого оператора. Звертаємо увагу на символ /, що передує імені і що означає (як зазначалося раніше), що ім'я літеральне, при цьому ім'я square переміщується в стек операндів, а не шукається в словниках і не заміщується своїм значенням. Згадаймо про відкладений характер процедур: при створенні процедур інтерпретатор не виконує їх негайно, а трактує як дані (літеральні) і переміщує в стек операндів. Оператор def витягує із стеку операндів поміщені туди ім'я і літеральну процедуру-масив і створює з них пару "ім'я - значення" в словнику userdict, що знаходиться на вершині стеку словників. Таким чином за допомогою вищенаведеного виразу в словнику створюється пара "ім'я" - square, "значення" – процедура {dup mul}, яка не виконується при її створенні, однак, буде виконуватися при подальшому непрямому зверненні до неї за допомогою імені square.

Коротко на конкретному прикладі торкнемося ще питання організації умовних конструкцій, без яких PostScript не був би повноцінною мовою програмування.

Приклад: nm lt {n} {m} ifelse.

За допомогою цього виразу визначається мінімальне значення з чисел, пов'язаних з іменами n і m. Як це працює? Інтерпретатор, виявивши здійсненні імена n і m, шукає їх у словниках, витягує відповідні значення і поміщає в стек операндів. Оператор lt (менше ніж) витягує два операнди з стека і порівнює їх. Якщо перший операнд менше другого, він поміщає в стек логічне значення true (логічна одиниця), інакше – значення false (логічний нуль). Далі інтерпретатор бачить процедури {n} і {m} і, оскільки зустрічає їх безпосередньо, то трактує як літеральні об'єкти, і поміщає в стек операндів. Оператор ifelse (якщо ... то, інакше ...) має три операнди (в стеку): логічне значення true або false і дві процедури. Якщо логічне значення є true, оператор ifelse змушує виконувати першу процедуру, інакше – другу. Всі три операнди беруться і видаляються з стека операндів до того, як вибрана процедура буде виконуватися. У

цьому прикладі кожна процедура складається з єдиного елемента, який є здійсненним ім'ям (n або m). Інтерпретатор шукає це ім'я в словнику і, оскільки воно пов'язане з числом, поміщає це число на вершину стека операндів. Таким чином, результатом виконання всього програмного фрагмента є розміщення на вершину стека операндів мінімального з чисел, пов'язаних з іменами n і m.

Далі слід сказати, що операторів PostScript досить багато, і якби метою нашої роботи був детальний виклад PostScript, то левову частину зайняло б висвітлення цих операторів. Але ж ми досягли поставленої мети розуміння ідеології функціонування і побудови PostScript, а інформація про конкретні оператори береться з довідників.

До речі, модуль AGM (Adobe Graphics Model), що використовується в InDesign, є фактично інтерпретатором PostScript: він реально виконує PostScript-інструкції і "відображає" результати на екрані, а не на папері. У порівнянні з програмами верстки від інших виробників, які користуються растровими прев'ю (попередній перегляд) сторінок, це набагато більш адекватний і точний спосіб попереднього перегляду і контролю. Це "реальне" WYSIWYG – те, що ви бачите на екрані, є максимально точним екранним відображенням того, як файл буде друкуватися.

Гаразд, – якщо PostScript є мовою програмування, а інтерпретатор є те, що виконує інструкції цієї мови, то що таке інкапсульований PostScript-файл, або EPS? Просто файл EPS є програма PostScript, збережена як окремий файл, який включає в себе "інкапсульоване" прев'ю низької роздільної здатності, що дозволяє деяким програмам відображати прев'ю на екрані. InDesign не потребує цього прев'ю, тому що він має вбудований інтерпретатор, що дозволяє відкрити PostScript- (і Illustrator-) файли природнім способом, тобто виконавши інструкції PostScript.

Тепер про PDF.

Portable Document Format (PDF) – кросплатформений формат електронних документів, створений фірмою Adobe Systems на базі PostScript. В першу чергу призначений для представлення в електронному вигляді та безпечного міжкомп'ютерного переміщення макетів видань. Значна кількість сучасного професійного комп'ютеризованого друкарського обладнання може обробляти PDF безпосередньо. Для перегляду можна використовувати офіційну безкоштовну програму Adobe Reader. Традиційним способом створення PDF-документів є віртуальний принтер, тобто документ, готується в своїй спеціалізованій програмі – графічній програмі або текстовому редакторі, САПР тощо, а потім експортується у формат PDF для розповсюдження в електронному вигляді, передачі в друкарню. PS-формат легко експортується в PDF за допомогою Adobe Acrobat Distiller.

PDF з 1 липня 2008 року є відкритим стандартом ISO 32000.

Формат PDF дозволяє вбудовувати необхідні шрифти, векторні та растрові зображення, форми і мультимедіавставки. Підтримує RGB, CMYK, Grayscale, Lab, Duotone, Bitmap, кілька типів стиснення інформації. Має власні технічні формати для поліграфії: PDF/X-1, PDF/X-3. Включає механізм електронних підписів для захисту і перевірки достовірності документів.

PDF в основному базується на PostScript, але спрощений. Оператори опису сторінок ідентичні операторам PostScript. Головна відмінність в тому, що PDF – не програмна мова і не містить складних програмних конструкцій: процедур, змінних величин і т.п. PDF-файл насправді файл PostScript, який вже було витлумачено інтерпретатором і представлено у вигляді чітко визначених об'єктів. Ці об'єкти можна переглянути на екрані нев коді, а у вигляді візуальних об'єктів. Оскільки ці файли вже пройшли через ін- терпретатор, вони можуть бути надійні- шими, ніж EPS або PS-файли при друку. Крім того, оскільки EPS файли і PS файли

можуть бути легко перетворені у PDF і переглядатись на екрані, виявляється додаткова перевага в тому, що можна побачити файл після інтерпретації, але перед відправкою його до пристроїв друку. Звідси можливість побачити помилки у файлі, перш ніж витрачати папір, плівку або пластини.

PDF має певну структуру, яка дозволяє програмам мати доступ до будь-якої частини документа. Структура PDF формату включає об'єкти, структуру файла, структуру документа, сторінковий опис. Більшість конструкцій, що описують структуру PDF формату, побудовано у вигляді словників (dictionary). Для прямого пошуку будь-якої сторінки файлу PDF-файл містить спеціальну таблицю посилань. Таблиця розміщується в кінці файлу і сприяє зменшенню часу пошуку і виведення сторінки в PDF-публікації, забезпечуючи незалежність часу пошуку від загальної кількості сторінок в документі.

Можна дискутувати про переваги та недоліки PostScript і PDF, стверджувати, що можна було б створити щось краще, але факт залишається фактом – тепер це міжнародний стандарт, яким треба керуватись і, який треба застосовувати професійно і коректно, а для цього його треба знати.

Висновки

Можна з великою мірою достовірності стверджувати, що використання PostScript і побудованого на його базі PDF з надбудовою JDF у видавничо-поліграфічній справі дає можливість в повній мірі реалізувати переваги єдиного стандартизованого підходу в технології, а спеціалісти, що вільно володіють цими напрямками знань, будуть конкурентно здатними в передових видавництвах, де вже й зараз ці технологічні досягнення активно впроваджуються.

Список літератури

1. Дурняк Б.В., Піх І.В., Сеньківський В.М. Системний аналіз та оптимізація параметрів книжкових видань: Монографія.- Львів: Українська академія друкарства, 2006. – 197 с.

2. PostScript Language Reference. Third Edition /Author Adobe Systems Incorporated /. – Addison Wesley, 1999. – 765 p.

3. PDF Reference Fifth Edition: Adobe® Portable Document Format Version 1.6. By: Adobe Systems Incorporated. – Adobe Press, 2004. – 1248 p.

4. Technical Note #5001, PostScript Language Document Structuring Conventions Specification, Version 3.0.

Статтю подано до редакції 25..05.2016