Narzędzia

†

Z CEP-em na EPS-y

Piotr Strzelczyk

Opisany poniżej pakiet CEP umożliwia kompresowanie PostScript-owych plików graficznych. Został on udostępniony jako oprogramowanie swobodne na konferencji BachoTEX '97.

Osoby zajmujące się obróbką danych graficznych od zawsze mają kłopoty z ogromnymi i wciąż rosnącymi plikami. Na przykład grafika formatu A4 przeskanowana z rozdzielczością 300 dpi składa się z ok. 8 700 000 pikseli. Jeśli jest to kolorowa mapa bitowa, w której każdy piksel opisany jest czterema bajtami (CMYK), to zajmuje ona na dysku w przybliżeniu 35 MB.

Niestety, to dopiero początek. Użytkownik TEX-a nie może używać danych w postaci binarnej (nie pozwala na to niestety DVIPS). Nasz biedny TEX-nik konwertuje więc grafikę do postaci szesnastkowego EPS, co dwukrotnie zwiększa zajmowane przez nią miejsce. Pamiętajmy również, że trzeba zarezerwować sobie co najmniej jeszcze raz tyle miejsca na dysku, aby miał gdzie powstać po przetworzeniu dokumentu TEX-em i DVIPS-em wynikowy plik *.ps zawierający całą informację graficzną. Okazuje się więc, że na każdą stronę A4 potrzebujemy 140 MB na dysku. A dyski niestety kosztują...

Problem ten jest znany już od dawna. Firma Adobe, tworząc wiele lat temu PostScript, najpopularniejszy język opisu strony, zawarła w specyfikacji poziomu drugiego (Level 2) obiekty zwane filtrami, umożliwiające m.in. kompresję danych. Możemy więc zamiast kodowania szesnastkowego zastosować kodowanie ASCII85 (objaśnienia skrótów i pojęć znajdują się na końcu artykułu), podobne do tego z unixowego uuencode. Mamy też dostęp do kompresji algorytmem RLE, algorytmem LZW, stosowanym w plikach JPEG algorytmem DCT, a także kilku innych.

Nasuwa się pytanie, dlaczego w codziennej praktyce nie stosuje się tych metod kompresji? Otóż niewiele programów – nie wiedzieć czemu – potrafi zapisać grafikę w postaci skompresowanych plików EPS, a do tego są to programy mało popularne.

Pakiet opisany w tym artykule, rozwiązuje w pewnym zakresie problem objętości plików graficznych. Umożliwia on kompresowanie najczęściej spotykanych, nieskompresowanych PostScript-owych plików graficznych. Niestety, w trakcie pracy nad programem okazało się, że problem jest bardziej skomplikowany, niż się początkowo wydawało. Na przykład nie da się wybrać jednego, najlepszego algorytmu kompresji i zawsze go stosować. Wybór ten musi zależeć od rodzaju kompresowanych danych, ich postaci, a także spodziewanego zastosowania. Z tego właśnie powodu pakiet składa się z aż dwóch zestawów programów, z których każdy posiada kilka opcji, umożliwiających pewne sterowanie procesem kompresji.

O programach

Pakiet składa się z dwóch par programów napisanych w języku AWK (CEP.AWK–UNCEP.AWK oraz COP.AWK–UNCOP.AWK), plików wsadowych dla systemu MS-DOS oraz informacyjnych plików tekstowych. Na podstawie analizy danych wejściowych skrypty CEP.AWK i COP.AWK tworzą program PostScript-owy, który przetwarzany za pomocą Ghostscripta dokonuje właściwej kompresji danych i tworzy plik wynikowy. UNCEP.AWK i UNCOP.AWK postępując podobnie realizują proces odwrotny, to znaczy dekompresję danych do postaci wejściowej.

CEP został stworzony do kompresji często spotykanych plików EPS zawierających pojedynczą mapę bitową zakodowaną szesnastkowo; COP potrafi skompresować dowolne dane PostScript-owe.

Powstaje pytanie, po co używać CEP-a, jeżeli COP potrafi skompresować dowolny plik PostScript-owy. Odpowiedź jest bardzo prosta – CEP wie, jakich danych się spodziewać, i w związku z tym potrafi je spakować dużo efektywniej. Poprawa efektywności wynika z dwóch czynników. Po pierwsze CEP spodziewa się danych szesnastkowych, czyli przyjmuje, że dwa znaki kodu opisują jeden bajt mapy bitowej (co już powoduje zwiększenie upakowania informacji). Po drugie mapy bitowe mają najczęściej zdecydowanie bardziej regularną, a tym samym redundantną strukturę niż przeciętny kod PostScript-owy. Wynika z tego, że w przypadku map bitowych nawet prostsze algorytmy kompresji mogą dać bardzo dobre rezultaty.

Z naszych eksperymentów wynika, że przy dobrych danych wejściowych (zrzuty grafiki z ekranu) łatwo jest osiągnąć nawet dwudziestokrotną kompresję. Niestety, w niektórych przypadkach żadna metoda kompresji nie daje efektów. W takiej sytuacji zawsze możemy zastosować filtr ASCII85, który zmniejszy objętość mapy bitowej zakodowanej szesnastkowo o około 35%.

Poniżej krótko opisujemy działanie CEP i COP. Jak na razie, pakiet ten działa tylko w systemie MS-DOS, choć mamy nadzieję, że uda się go łatwo przenieść na inne platformy, tam gdzie są dostępne GAWK i Ghostscript. W tej wersji zastosowaliśmy implementację AWK-a stworzoną w ramach inicjatywy GNU: GAWK-EMX.EXE oraz wersję Ghostscripta dla systemu MS-DOS: GS386.EXE.

Do testów pakietu używaliśmy kilku wersji Ghostscripta i GAWK-a. Najlepsze wyniki otrzymaliśmy stosując Ghostscript 4.03 oraz GAWK 3.0.0. Aktualnie używamy już Ghostscripta w wersji 5.10 oraz GAWK 3.0.3. (w wersji 32-bitowej – GAWK32.EXE) i nie zauważyliśmy żadnych problemów.

CEP

W skład podpakietu CEP wchodzą DOS-owe pliki wsadowe CEP.BAT i UNCEP.BAT oraz programy AWK-owe CEP.AWK i UNCEP.AWK. Po wywołaniu pliku wsadowego najpierw uruchamiany jest AWK, który przegląda dane wejściowe, stara się znaleźć w nich mapę bitową zakodowaną szesnastkowo i tworzy odpowiedni program PostScript-owy. Następnie zostaje uruchomiony Ghostscript, który wykonując przygotowany w poprzednim kroku program, dokonuje właściwego przetwarzania danych. Przepisuje on też oryginalną preambułę PostScript-ową dokonując w niej tylko kilku zmian; żadne komentarze DSC nie ulegają zmianie.

Jeżeli program nie znajdzie w pliku mapy bitowej albo w trakcie analizy struktury pliku powstają jakieś problemy, CEP poddaje się i nie tworzy wynikowego pliku EPS.

Przed skasowaniem oryginalnej wersji zalecamy sprawdzenie, czy powstały plik jest prawidłowy. W niektórych (rzadkich) przypadkach CEP może błędnie rozpoznać mapę bitową. Czasami także błędy Ghostscripta mogą powodować powstanie uszkodzonego pliku, *lepiej więc zawsze obejrzeć go przed usunięciem źródła*.

CEP nie tworzy binarnych plików wyjściowych, zawsze stosuje kodowanie ASCII85 lub szesnastkowe. Wynika to z naszego założenia, że kompresowane CEP-em pliki będą używane głównie w środowisku TEX-owym z DVIPS-em jako sterownikiem PostScript-owym. Niezależnie od tego założenia pliki CEP-owe można używać w dowolnych systemach DTP, które wczytują tak zwane *placeable* EPS. Niestety takie systemy najczęściej wymagają plików EPS zawierających podgląd w postaci binarnego TIFF-a, który może zostać źle przetworzony przez (G)AWK-a.

UNCEP dekompresuje wszystkie pliki stworzone CEP-em pod warunkiem, że nie zostały one w żaden sposób zmienione. W skompresowanym pliku występuje specjalny komentarz %UNCEPInfo:, który zawiera informacje potrzebne do dekompresji. Jest on jednak bardzo czuły na budowę pliku. Nawet tak nieznaczna zmiana jak dodanie (lub usunięcie) linii komentarza powoduje, że pliku nie daje się zdekompresować. Trzeba też zwrócić uwagę na to, że podstawą kompresji jest tu przetwarzanie *strumienia danych*, czyli *ciągu liczb*, a nie *ciągu znaków*, co implikuje, że informacja o sposobie łamania wierszy podczas czytania danych szesnastkowych jest tracona. W związku z tym UNCEP nie może wygenerować pliku identycznego z wejściowym. Nie ma to żadnego znaczenia dla PostScript-u, gdyż jest on całkowicie odporny na różne metody łamania linii. Niestety niektóre programy „czytające" pliki EPS z niezrozumiałych przyczyn wymagają łamania linii według własnych zasad. Tego typu programy (np. Aldus PhotoStyler) nie odczytają zdekompresowanego pliku.

Polecenie CEP ma następującą postać (nazwa pliku wyjściowego musi być różna od nazwy pliku wejściowego):

```
cep.bat plik_we plik_wy [opcje]
```

Program rozpoznaje następujące opcje:

8 – kodowanie ASCII85 (domyślnie);
h lub H – kodowanie HEX (szesnastkowe);
r lub R – kompresja RLE (domyślnie);
l lub L – kompresja LZW;
f lub F – kompresja Flate (PDF oraz Level 3);
n lub N – bez kompresji.

Polecenie UNCEP ma następującą postać (jak poprzednio nazwa pliku wyjściowego musi być różna od nazwy pliku wejściowego):

```
uncep.bat plik_we plik_wy
```

Jak zostało wspomniane, metoda dekompresji jest ustalana na podstawie pliku wejściowego.

COP

W skład podpakietu COP wchodzą DOS-owe pliki wsadowe COP.BAT i UNCOP.BAT oraz programy AWK-owe COP.AWK i UNCOP.AWK. COP nie przeprowadza właściwie żadnej analizy danych POSTSCRIPT-owych, jedynym wyjątkiem jest próba odnalezienia komentarza strukturalnego %%BoundingBox:. Jeśli próba się powiedzie, jest on dołączany do generowanej preambuły, w przeciwnym wypadku preambuła nie zawiera komentarza %%BoundingBox:. Reszta pliku wejściowego jest traktowana jak dane binarne i kodowana zgodnie z parametrami wywołania.

Pliki wygenerowane przez COP są czytane przez każdy interpreter POSTSCRIPT-u Level 2, a dokładniej przez każdy interpreter, który poprawnie realizuje polecenia związane z obiektami typu filter.

Przy dekompresji nie trzeba podawać żadnych opcji, UNCOP ustala bowiem – podobnie jak UNCEP – metodę dekompresji na podstawie nagłówka pliku wejściowego, natomiast – w przeciwieństwie do UNCEP – odtwarza dokładnie każdy bit oryginalnego pliku. Jest jeszcze jedno, niestety przykre, podobieństwo obu podpakietów: z powodu błędów Ghostscripta usunięcie pliku źródłowego przed sprawdzeniem, czy plik wynikowy jest prawidłowy, może się skończyć niemiłą niespodzianką. Dlatego przed skasowaniem źródeł *zalecamy obejrzenie zdekompresowanego pliku za pomocą programu Ghostscript.*

COP może być używany do kompresowania danych POSTSCRIPT-owych dla różnych aplikacji, tak więc zdecydowaliśmy się w tym wypadku dopuścić możliwość kodowania binarnego. Kompresowane EPS-y niestety nie mogą zawierać podglądu TIFF-owego, wykorzystywanego przez wiele programów przetwarzających grafikę POSTSCRIPT-ową. Plik wynikowy może być używany jako tak zwany *placeable* EPS bez podglądu (*preview*), można też wczytywać skompresowane pliki jako tak zwany *POSTSCRIPT interpreted.* Niekiedy się to udaje, ale nie należy się dziwić, jeśli się to nie powiedzie – interpretacja POSTSCRIPT-u to naprawdę trudne zadanie.

Polecenie COP ma następującą postać (nazwa pliku wyjściowego musi być różna od nazwy pliku wejściowego):

```
cop.bat plik_we plik_wy [opcje]
```

Program rozpoznaje następujące opcje:
- 8 – kodowanie ASCII85 (domyślnie);
- b lub B – kodowanie binarne;
- h lub H – kodowanie HEX (szesnastkowe);
- r lub R – kompresja RLE (domyślnie);
- l lub L – kompresja LZW;
- f lub F – kompresja Flate (PDF oraz Level 3);
- n lub N – bez kompresji.

Kodowanie binarne polega w istocie na wyłączeniu kodowania – jest to „naturalny” wynik korzystania z filtrów.

Polecenie UNCOP ma następującą postać:

```
uncop.bat plik_we plik_wy
```

Podobnie jak poprzednio nazwa pliku wyjściowego musi być różna od nazwy pliku wejściowego, i tak jak w przypadku dekompresji UNCEP-em metoda dekompresji jest ustalana na podstawie pliku wejściowego.

Sterta uwag o pakiecie

Poniżej podajemy rozwiązania niektórych problemów napotkanych podczas tworzenia pakietu:

1. Nie ma prostej metody znajdowania zakodowanej szesnastkowo mapy bitowej w pliku EPS. Analiza semantyczna (do zrealizowania przez przedefiniowanie instrukcji image, imagemask i colorimage) jest bardzo trudna do pełnej realizacji (parametry w postaci słowników). Zdecydowaliśmy się więc na wyszukiwanie mapy bitowej oparte na analizie składni; to z kolei wywołało kłopoty z analizą napisów add oraz def, które mogą być zarówno częścią mapy bitowej, jak i instrukcją POSTSCRIPT-ową.
2. Nie da się także podać dokładnych wytycznych, która metoda kompresji będzie najlepsza dla konkretnych danych. Zwykle najlepszą metodą kodowania jest ASCII85. Dla map bitowych (kompresowanych CEP-em) najczęściej wystarczy algorytm RLE, choć dla kolorowych rysunków lepszy jest LZW. Najlepsze wyniki prawie zawsze daje metoda Flate. Jednak Flate, jak również LZW mają ograniczone zastosowania:

   (a) kompresja LZW została w wersjach 4.x Ghostscripta wyłączona z powodu nieszczęsnego patentu (p. „Słowniczek” zamieszczony na końcu artykułu). Firma Aladdin (twórca Ghostscripta) zastosowała zastępczo filtr kodujący zgodny z LZW, który niestety nie kompresuje danych, a wręcz je zwiększa o 10%.

Można używać starszych wersji Ghostscripta lub skompilować Ghostscript z filtrem LZW na własną odpowiedzialność...

(b) kompresja Flate (oparta na tym samym algorytmie co GZIP) na razie niestety nie jest dostępna w fotonaświetlarkach. Firma Adobe włączyła ten algorytm najpierw do specyfikacji formatu PDF 1.2, a następnie do ogłoszonego w roku 1997 PostScript-u Level 3. W chwili obecnej filtry FlateEncode and FlateDecode są dostępne w Polsce prawdopodobnie tylko dla użytkowników Ghostscripta, choć np. firma Agfa już sprzedaje naświetlarki z interpreterem PostScript Level 3.

Z naszych doświadczeń wynika, że w wypadku zdjęć optymalne efekty daje wyłączenie kompresji i stosowanie tylko kodowania ASCII85. Żadna bezstratna metoda kompresji nie jest w stanie tu wiele poprawić, gdyż zdjęcia zawierają dużo informacji szumowej. Nawet takie programy jak ARJ, ZIP czy LHARC nie dadzą istotnie lepszych wyników. Jedyną możliwością jest kompresja fotografii algorytmami stratnymi, takimi jak stosowany w plikach JPEG algorytm DCT.

3. Zgodnie z tym co powiedzieliśmy wyżej, najczęściej polecane jest stosowanie kodowania ASCII85, niestety wywołuje ono pewne problemy. Otóż dane zakodowane filtrem ASCII85 mogą zawierać linie, zaczynające się od dwóch znaków procenta %% lub od procenta i wykrzyknika %!, które dość skutecznie podszywają się pod komentarze strukturalne. W tym kontekście wydaje się dziwne, że firma Adobe nie zdecydowała się na usunięcie procenta z zestawu znaków ASCII85. Niestety nie uczyniła tego, więc niektóre programy używające plików EPS, mogą próbować przetwarzać linie zawierające takie „niby-komentarze". DVIPS jest tego przykładem – standardowo usuwa linie komentarzy z wczytywanych plików. Można zablokować tę funkcję (opcja -K0), ale wtedy pozostają również prawdziwe komentarze DSC, które mogą powodować błędy w analizie dokumentu przez programy korzystające z konwencji DSC.
4. Przydatne mogłoby być włączenie do pakietu jeszcze kilku standardowych filtrów PostScript-u, na przykład DCT oraz CCITTFax. Jednak filtry te wymagają podania wielu parametrów wejściowych, więc korzystanie z nich byłoby dużo bardziej skomplikowane. Co więcej, właściwie nie wiadomo, jak można ustalać parametry kompresji DCT, nie mając możliwości interakcyjnego korygowania efektów. Na razie więc nie oprogramowaliśmy tych filtrów. W zamian przygotowujemy programik umożliwiający konwersję plików JPEG (czyli bardzo mocno skompresowanych bitmap) do postaci PostScript-owych EPS-ów (używających filtra DCT).
5. W pakiecie CEP staraliśmy się oszczędnie gospodarować przestrzenią dyskową. W czasie pracy nie są tworzone żadne duże pliki tymczasowe; właściwie do kompresji pliku potrzebne jest tyle wolnego miejsca na dysku, ile zajmuje plik wynikowy.
6. W czasie półrocznej eksploatacji pakietu okazało się, że w wypadku niektórych komercyjnych urządzeń PostScript-owych zgodność z standardem PostScript Level 2 kończy się na efektownych ulotkach i nalepce na obudowie. Część naświetlarek (zwłaszcza starszych) nie jest w stanie, mimo obietnic producentów, przetwarzać poprawnie plików *.ps zawierających operacje związane z obiektami typu filter.

Jedyną metodą sprawdzenia czy dana naświetlarka prawidłowo realizuje wymienione filtry jest niestety metoda prób i błędów. W wypadku korzystania z naświetlarki, która odmawia współpracy z pakietem CEP (czyli *nie* jest zgodna z PostScript-em Level 2), możemy: (a) zmienić naświetlarnię, (b) wymusić na naświetlarni zakup nowego urządzenia, lub ewentualnie uaktualnienia oprogramowania naświetlarki (w przypadku RIP-a programowego) lub (c) zrezygnować z używania CEP-a.

Pomocny może okazać się tu krótki plik PostScript-owy:

```
%!PS-Adobe-2.0 EPSF-1.2
%%Pages: 1
%%BoundingBox: 0 0 540 150
%%EndComments
/Helvetica 8 selectfont
90 rotate
1 2 moveto
(*)
{0 -10 rmoveto gsave show grestore}
255 string
```

```
/Filter
resourceforall
showpage
%%EOF
```

Program ten spowoduje wypisanie listy filtrów obsługiwanych przez dany interpreter. Jeśli w czasie przetwarzania pojawia się błąd (obsługa w studiu graficznym twierdzi, że „plik się nie RIP-uje"), to najprawdopodobniej mamy do czynienia ze zbyt ubogą wersją PostScript-u i musimy zrezygnować z używania CEP-a.

Powyższy plik można albo bezpośrednio wysłać na urządzenie PostScript-owe (naświetlarkę, drukarkę), albo umieścić w dokumencie TEX-owym jak zwykły plik EPS.

7. Błędy i pułapki:

(a) Poprzedzenie komendy `closefile` teoretycznie nadmiarową komendą `flushfile` pozwala ominąć pluskwę w Ghostscript 3.x (zjadanie końcówki pliku wyjściowego).

(b) Dodatkowy, pusty filtr `NullEncode` pozwala nam ominąć pewną (wielokrotnie poprawianą) pluskwę Ghostscripta. Mianowicie filtr `ASCII85` z wyjściem danych skierowanym do procedury często zapisuje w strumieniu wynikowym dodatkowe znaczniki końca danych, czyli `~>` (przy pewnych parametrach może zapisać ich tysiące). Skierowanie danych wyjściowych do procedury, zamiast bezpośrednio do pliku uniemożliwia niestety stosowanie wcześniejszych niż 3.x wersji Ghostscripta przy kodowaniu `ASCII85`. Na szczęście Ghostscript w wersji od 2.6x może być stosowany przynajmniej do kodowania szesnastkowego (`HEX`) – ta wersja Ghostscripta zawiera również „legalną" kompresję `LZW`.

(c) Procedura docelowa, o której mowa w podpunkcie 7.b, została dodana, aby w sposób kontrolowany zmodyfikować wiersze, które mogłyby być błędnie zinterpretowane jako komentarze `DSC` (por. też podpunkt 3). Wiersze przełamywane są po znaku procenta. Zabezpiecza to przed konsekwencjami często stosowanej, a w tym wypadku bardzo groźnej opcji `DVIPS`-a „usuwaj komentarze" (`-K1`).

(d) Dziwaczna składnia komendy `{2 2 .quit}` zamiast formy `{2 .quit}` została zastosowana z powodu błędu Ghostscripta 3.5x, który wpadał w nieskończoną pętlę. Wewnętrzną instrukcję Ghostscripta `.quit` wybraliśmy po to, by umożliwić obsługę błędów na poziomie systemu operacyjnego.

(e) W starszych wersjach Ghostscripta występuje jeszcze jeden dziwny błąd, którego nie udało się nam ominąć. Mianowicie niektóre pliki `EPS` są poprawnie kompresowane przez Ghostscript 2.6, ale ta sama wersja Ghostscripta nie potrafi ich wyświetlić. Podobne objawy obserwowaliśmy w wypadku innego pliku w wersji 3.51 Ghostscript. Oczywiście takie pliki dają się bez problemu odczytać w wyższych wersjach programu. Nowsze wersje Ghostscripta zachowują się w sposób bardziej stabilny, jeśli chodzi o filtrowanie danych. Gdyby nie ów nieszczęsny patent na `LZW`...

(f) Podsumowując, stanowczo polecamy stosowanie Ghostscripta w wersji 4.x lub 5.x (ewentualnie z dokompilowanym algorytmem `LZW`) oraz `GAWK`-a w wersji 3.x. Ghostscript 4.x jest praktycznie pełną implementacją poziomu drugiego (Level 2) PostScript. `GAWK` 3.x umożliwia stosowanie wyrażeń regularnych jako separatorów rekordów, co pozwala z kolei na obsługiwanie znaków końca linii w sposób dokładnie zgodny z specyfikacją PostScript. Wersja 3.x `GAWK`-a jest także znacznie stabilniejsza od wersji poprzednich.

CEP dla wszystkich

Autorami pakietu CEP są Bogusław Jackowski, Piotr Pianowski i Piotr Strzelczyk. Powstał on w firmie BOP s.c wiosną 1997 roku. Podczas (jak zawsze wspaniałej) konferencji BachoTEX '97 dokonaliśmy publicznej prezentacji pakietu. Jednocześnie przekazaliśmy go do użytku społeczności TEX-owej. W związku z tym wszystkie prawa, lewa, góry, doły autorskie, czy jak tam to zwać, dotyczące wszystkich plików wchodzących w skład pakietu CEP, mają charakter oprogramowania swobodnego.

Udostępniając ten pakiet mamy nadzieję, że okaże się on przydatny, nie możemy jednak dać na niego żadnej gwarancji. Staraliśmy się znaleźć i usunąć z pakietu wszystkie błędy, ale nie ponosimy żadnej odpowiedzialności za uszkodzenia bądź straty wywołane jego właściwym lub niewłaściwym użyciem. Będziemy oczywiście starali się – w miarę naszych możliwości – pielęgnować pakiet, jednakże trudno nam obiecać, że dotrzymamy kroku częstości zmian Ghostscripta.

Prosimy o przesyłanie wszelkich uwag, wykrytych błędów oraz ewentualnych ulepszeń do autorów pakietu. Można się z nami skontaktować pod adresem B.Jackowski@gust.org.pl.

Słowniczek

Poniżej zamieszczamy krótki słowniczek nazw i pojęć występujących w tym artykule (w kolejności bynajmniej nie alfabetycznej).

- Ghostscript – wspaniały interpreter języka PostScript, stworzony przez Aladdin Enterprise, dostępny bezpłatnie na zasadach *free public license*; aktualna wersja (na grudzień 1997 – wersja 5.10) jest często lepsza i bardziej stabilna od sprzedawanych za tysiące dolarów interpreterów fotonaświetlarek (tzw. RIP-ów).
- AWK – program narzędziowy, a także język programowania umożliwiający skuteczne i łatwe do zaprogramowania przetwarzania wsadowe plików tekstowych, stworzony w 1977 r. przez Alfreda V. Aho, Petera J. Weinbergera i Briana W. Kernighana.
- GAWK – GNU AWK – dostępna bezpłatnie na zasadach „GNU Free Software Foundation" implementacja AWK-a, napisana w 1986 r. przez Paula Rubina oraz Jay Fenlasona z pomocą Richarda Stallmana.
- GNU – inicjatywa *The Free Software Foundation*, niekomercyjnej organizacji zajmującej się tworzeniem i rozpowszechnianiem darmowego oprogramowania. Fundację założył Richard M. Stallman.
- TeX – system komputerowego składu tekstu, stworzony przez prof. Donalda E. Knutha z Uniwersytetu Stanforda, rozpowszechniany jako dobro publiczne.
- DVIPS – program tworzący pliki PostScript-owe z TeX-owych plików DVI, autorstwa Tomasa Rokickiego z Uniwersytetu Stanforda.
- DSC – *Document Structuring Convention* – zdefiniowany przez firmę Adobe standard strukturyzacji plików PostScript-owych.
- ASCII85 – algorytm kodowania danych binarnych w postaci tekstowej. Koduje każde cztery bajty jako pięć znaków z zakresu od % do u; cztery bajty zerowe są traktowane inaczej i kodowane jako litera z (szczegóły w dokumentacji języka PostScript, strony 128–130).
- RLE – *run length encoding* (kodowanie powtarzających się bajtów) – standardowa metoda kompresji danych (szczegóły w dokumentacji języka PostScript, strony 133–134).
- LZW – algorytm kompresji wymyślony w 1978 roku przez J. Ziva i A. Lempela, poprawiony przez T. Welcha w 1984 r.; w 1985 roku opatentowany w Stanach Zjednoczonych przez firmę Unisys, zatrudniającą Welcha. Firma Unisys uznała, że implementacje algorytmu sprzed roku 1985 są wolne od opłaty autorskiej. (Informacja zaczerpnięta z opracowania Nelsona H. F. Beebe, email beebe@math.utah.edu.)
- DCT – *discrete cosine transform compression* (kompresja oparta o numeryczną transformatę kosinusową) – bardzo efektywna, stratna metoda kompresji danych (głównie graficznych).
- JPEG – *Joint Photographic Experts Group* – organizacja, która stworzyła będący standardem format zapisu plików oparty na kompresji algorytmem DCT. Filtr PostScript-owy DCTEncoding jest zgodny ze standardem JPEG.
- GZIP – program kompresujący stworzony przez GNU Free Software Foundation jako odpowiedź na opatentowanie algorytmu LZW, oparty na bardzo skutecznym i ogólnie dostępnym algorytmie (wariant algorytmu Lempela i Ziva). Jest on obecnie standardowym narzędziem kompresji danych w systemach unixowych.
- Flate – filtr PostScript-u Level 3 wykorzystujący kompresję opartą o ten sam algorytm, który używany jest przez program GZIP; nazwa pochodzi od angielskich słów *deflate-inflate*.

◇ Piotr Strzelczyk
PiotrS@telbank.pl