Materiały **Ćwiczenia**

Materiały do laboratorium znajdują się tutaj. Po rozpakowaniu powinniśmy zobaczyć dwa katalogi `drop` i `japan`, zwierające obrazy.

Obróbka obrazków

Program convert

Podstawowym programem, który będziemy używać na zajęciach jest program `convert` z biblioteki ImageMagick. Program ten służy do konwertowania i zmiany właściwości obrazów. Potrafi także dodawać elementy do obrazu, a nawet tworzyć obrazy od zera. Najłatwiej zobaczyć jak go używać analizując przykłady.

Uwaga:: Zanim rozpoczniesz pracę, skopiuj zdjęcia do tymczasowego katalogu!

Wykonaj poniższe operacje i sprawdź efekty.

- `convert plik.gif plik.jpg` – konwertuje plik w formacie `.gif` na format `.jpg`,
- `convert plik1.jpg -resize 50% plik2.jpg` – zmniejsza obrazek dwukrotnie,
- `convert plik1.jpg -resize 100 plik2.jpg` – zmniejsza obrazek, tak aby krótszy wymiar był równy 100 pikseli,
- `convert plik1.jpg -resize 100x100 plik2.jpg` – zmniejsza obrazek tak, aby mieścił się w kwadracie o wymiarze 100 na 100 pikseli,
- `convert plik1.jpg -resize 100x100\! plik2.jpg` – zmniejsza obrazek dokładnie do rozmiaru 100 na 100 pikseli,
- polecenie

```
convert -size 320x85 canvas:none -pointsize 72 -fill red \
-draw "text 20, 55 'Magic'" magic.jpg
```

– stworzy obrazek `magic.jpg`, z naniesionym tekstem "Magic" (znak \ na końcu linii oznacza, że ciąg dalszy komendy nastąpi w kolejnej linii).

Napisz skrypt, który:

- Zmniejszy wszystkie pliki `.jpg` zawarte w tym samym katalogu, w którym znajduje się skrypt.
- Zmniejszy wszystkie pliki `.jpg` zawarte w tym samym katalogu, w którym znajduje się skrypt i umieści je w innym katalogu.
- Dokona konwersji wszystkich plików `.jpg` na `.gif`, dodając końcówkę (`plik.jpg -> plik.jpg.gif`).
- Dokona konwersji wszystkich plików `.jpg` na `.gif`, zmieniając końcówkę (`plik.jpg -> plik.gif`).
- Na każde zdjęcie naniesie tekst wykorzystując argument

```
-pointsize rozmiar -draw "text x, y 'Tekst'"
```

- Na każde zdjęcie naniesie ramkę (argument `-border 20x20`) z aktualną datą (komenda `date`).
- Na każde zdjęcie naniesie datę utworzenia tego zdjęcia (można ją wyciągnąć za pomocą komendy `stat -c %y plik`).
- Zmniejszy wszystkie obrazki z katalogu "drop" i połączy je w animację za pomocą komendy `convert *.jpg animacja.gif`.

Automatyzacja

Spróbuj napisać skrypty wykonujące następujące zadania:

- Doda do obrazka ramkę i wypisze na niej wybraną informację `EXIF`. Informację tę można wyciągnąć za pomocą polecenia `identify -format "%[EXIF:*]" plik.jpg`. Może to być np. modelu aparatu, którym wykonano zdjęcie.
- Wypisze na ekran liczby od 0 do 10 za pomocą pętli.
- Wypisze na ekran liczby od 0 do pewnej liczby podanej jako argument skryptu.
- Połączy wszystkie obrazki w danym katalogu, zmniejszone do rozmiaru 10 na 10 pikseli, w jeden duży obraz `JPG`. Argument `-append` łączy obrazy w pionie, a argument `+append` w poziomie. Przykładowo, komenda `convert 1.jpg 2.jpg 3.jpg +append 4.jpg` złączy poziomo trzy obrazki w jeden.

- W sposób analogiczny, połączy obrazy z katalogu `drop` w jeden duży obraz. Obrazy powinny być połączone tak aby tworzyły szachownicę 10 na 10 obrazów. Podpowiedź: połącz obrazki `drop-00*.jpg` w poziomie, później `drop-01*.jpg`, itd. Następnie wszystkie te podłużne obrazki połącz w całość w pionie.
- Rozłoży animację GIF na pojedyncze obrazy JPG. Do każdego z otrzymanych obrazów dopisze tekst, a następnie złoży je w nową animację GIF. Skrypt powinien skasować pliki tymczasowe (tzn. pojedyncze obrazy otrzymane przez rozłożenie pierwotnej animacji).

Czasem napisanie skryptu może być żmudne. Dotyczy to zwłaszcza sytuacji, w których skrypt musi być rozbudowany a jednocześnie zostanie wykorzystany tylko raz. Przykładowo, poniższy skrypt generuje obrazek złożony z trzech nakładających się obrazów: czarno-białego tła oraz dwóch plików `1.jpg` i `2.jpg`.

```
#!/bin/bash

convert -size 90x60 canvas:white \
-size 90x30 gradient: -append -rotate 90 \
\( 1.jpg -resize 90x90\! -clone 0 -compose CopyOpacity \
+matte -composite -repage +60+0 \) \
\( 2.jpg -resize 90x90\! -clone 0 -compose CopyOpacity \
+matte -composite -repage +120+0 \) \
-compose Over -mosaic overlap_series.jpg
```

Przeanalizuj powyższy przykład i napisz skrypt, który stworzy obrazek ze wszystkich plików zawartych w określonym katalogu. W tym celu skrypt powinien:

- generować skrypt `tmp.sh` zawierający komendę analogiczną do powyższej, tylko dla większej liczby obrazów (użyj `echo linia > tmp` dla pierwszej linii i `echo kolejna linia » tmp` dla każdej następnej),
- nadawać uprawnienia do wykonywania dla skryptu `tmp.sh`,
- uruchamiać go.

Wczytywanie danych binarnych

Program `convert` może wczytać dane binarne interpretując je jako obrazek. Przykładowo, może to być tablica zawierająca wartości typu `char`. Napisz program `obrazek.c` zawierający poniższy kod:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
  const int N = 100;
  const int M = 100;
  char* tab   = (char*)malloc(N * M * sizeof(char));

  for (int i = 0; i < N * M; ++i)
    tab[i] = 255 * i / (N * M - 1);

  fwrite(tab, sizeof(char), N * M, stdout);
  free(tab);
}
```

Następnie skompiluj go za pomocą komendy

```
gcc obrazek.c -o obrazek.out
```

i wykonaj

```
./obrazek.out > obrazek
```

Mamy teraz plik binarny zawierający liczby typu `char` (1 bajt) od 0 do 255. Możemy dokonać jego konwersji na obrazek `JPG` wpisując:

```
convert -size 100x100 -depth 8 gray:obrazek obrazek.jpg
```

Uwaga: Zamiast typu `char` moglibyśmy użyć tablicy typu `float` i liczb z przedziału $[0, 1]$. Wtedy jednak musielibyśmy skonwertować obrazek za pomocą polecenia:

```
convert -size 100x100 -depth 32 \
-define quantum:format=floating-point gray:obrazek obrazek.jpg
```

Ćwiczenia

- Zmniejsz dowolne zdjęcie do rozmiarów (dokładnie) 100 na 100 pixeli.

- Zmodyfikuj utworzoną komendę, tak aby konwertować to zdjęcie na plik binarny.
- Na podstawie programu `obrazek.c` napisz program `filtr.c`, który:
  - wczyta ze standardowego wejścia tablicę (służy do tego funkcja "fread"),

```
size_t fread(void* ptr, size_t size, size_t count, FILE* stream);
```

    gdzie: `ptr` to wskaźnik na tablicę, `size` to rozmiar elementu tablicy, `count` to liczba elementów do wczytania, a `stream` to wskaźnik do strumienia, na którym wykonywana jest operacja (w naszym przypadku będzie to `stdin`),
  - dla każdej wczytanej liczby `x` znajdzie wartość, która pozwoli na odwrócenie odpowiadającego jej koloru (zauważ, że dla liczb typu `char` kolor czarny to 0, a kolor biały to 255),
  - wynik przekształcenia wyśle do standardowego wyjścia.
- Spróbuj przepuścić wybrany obrazek przez taki "filtr" i sprawdź wynik. Pamiętaj, że informacje możemy wysłać na standardowe wejście programu za pomocą `<`.
- Napisz skrypt, który wszystkie pliki z aktualnego katalogu zmniejszy do rozmiaru 100 na 100 pikseli, dokona konwersji na pliki binarne, przepuści przez filtr i zapisze wyniki w postaci plików `.jpg`.