Sprawozdanie

z projektu z podstaw programowania

Patrycja Stasiak
Milena Baranowska
Inżynieria danych

Zad 3

Zamienia dane z pliku tekstowego, które są **w jednej kolumnie (czyli po jednej linii)**, na dane **w trzech kolumnach w jednym wierszu**. Co trzy linie robi nowy wiersz. Wynik zapisuje do pliku `dane_w_kolumnach.txt`.

```
patry@Patrycja MSYS ~
$ awk '{ printf "%s\t", $0; if (NR % 3 == 0) printf "\n" }' "/c/Users/patry/OneDrive/Pulpit/dane.txt
" > dane_w_kolumnach.txt

patry@Patrycja MSYS ~
$ cat dane_w_kolumnach.txt
0       0       0
0,001   0,000999999833333342    0,00015923558917198 1
0,002   0,00199999866666693     0,00031847070063715 5
0,003   0,00299999550000203     0,00047770485668951
0,004   0,00399998933334187     0,00063693757962462 8
0,005   0,00499997916669271     0,00079616839174047 9
0,006   0,0059999640000648      0,00095539681533821 7
0,007   0,00699994283347339     0,00111462237272298
0,008   0,00799991466693973     0,00127384458620467
0,009   0,00899987850049207     0,00143306297809878
0,01    0,00999983333416666     0,00159227707072717
0,011   0,0109997781680088      0,00175148638641886
0,012   0,0119997120020736      0,00191069044751082
0,013   0,0129996338364274      0,0020698887763488
0,014   0,0139995426711485      0,00222908089528809
0,015   0,0149994375063281      0,00238826632669433
0,016   0,0159993173420714      0,00254744459294431
0,017   0,0169991811784987      0,00270661521642677
0,018   0,0179990280157463      0,00286577771954318
0,019   0,0189988568539673      0,00302493162470853
0,02    0,0199986666933331      0,00318407645435216
0,021   0,0209984565340338      0,00334321173091853
```

Zad 4

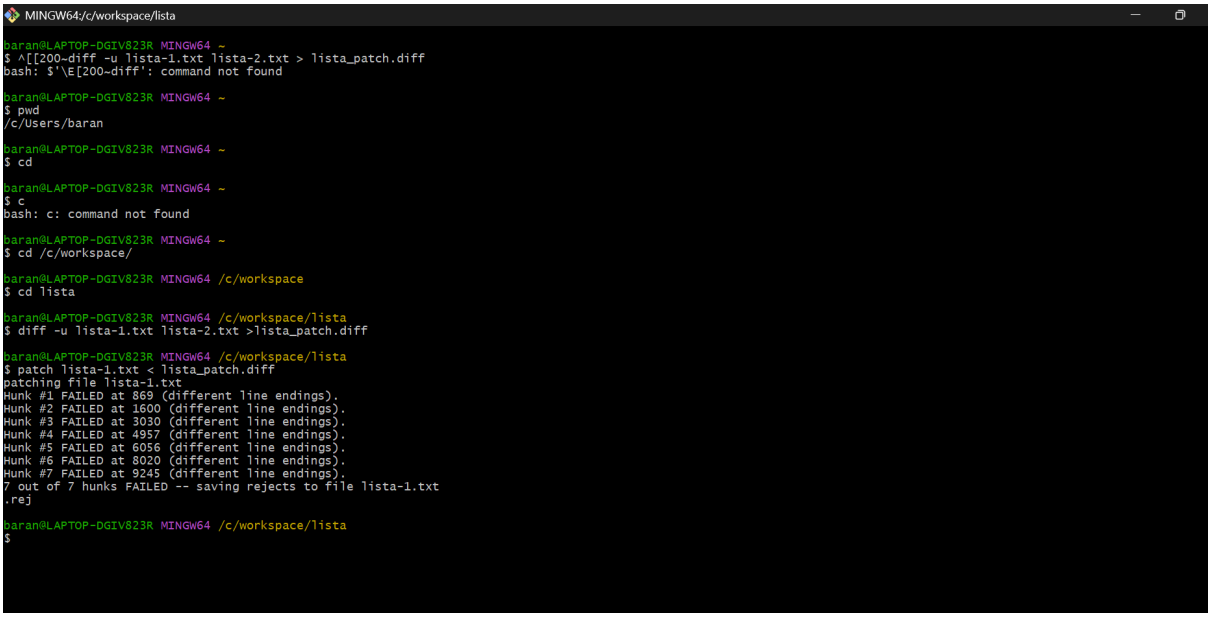
Porównuje pliki `lista-1.txt` i `lista-2.txt`.

Wynik porównania (czyli różnice) zapisuje do pliku `lista_pop.txt.txt`.

Ten plik `.diff` zawiera instrukcje, jak zmienić `lista-1.txt`, żeby wyglądał jak `lista-2.txt`.

Przekazuje plik `.diff` do programu `patch`.

`patch` próbuje **zmodyfikować `lista-1.txt`** na podstawie różnic zawartych w `lista_pop.txt.txt`.



Zad 5

`BEGIN { print "dateTime;steps;synced" }`

➤ Na początku wypisuje nagłówek pliku CSV.

`/INSERT INTO/ { ... }`

➤ Przetwarza tylko linie zawierające `INSERT INTO`.

`match($0, /\(([^)]+)\)/, arr)`

➤ Wyciąga dane z pierwszego nawiasu `(dane1, dane2, dane3)` i zapisuje je do tablicy `arr`.

`split(arr[1], vals, ",")`

➤ Dzieli dane po przecinkach, np. `vals[1] = timestamp`, `vals[2] = steps`, `vals[3] = synced`.

`dateTime = int(vals[1] / 1000)`

➤ Konwertuje czas z milisekund na sekundy.

`steps = vals[2] + 0`

➤ Zapisuje liczbę kroków (jako liczba).

`synced = vals[3] + 0`

➤ Zapisuje informację, czy dane zostały zsynchronizowane (np. 0 = nie, 1 = tak).

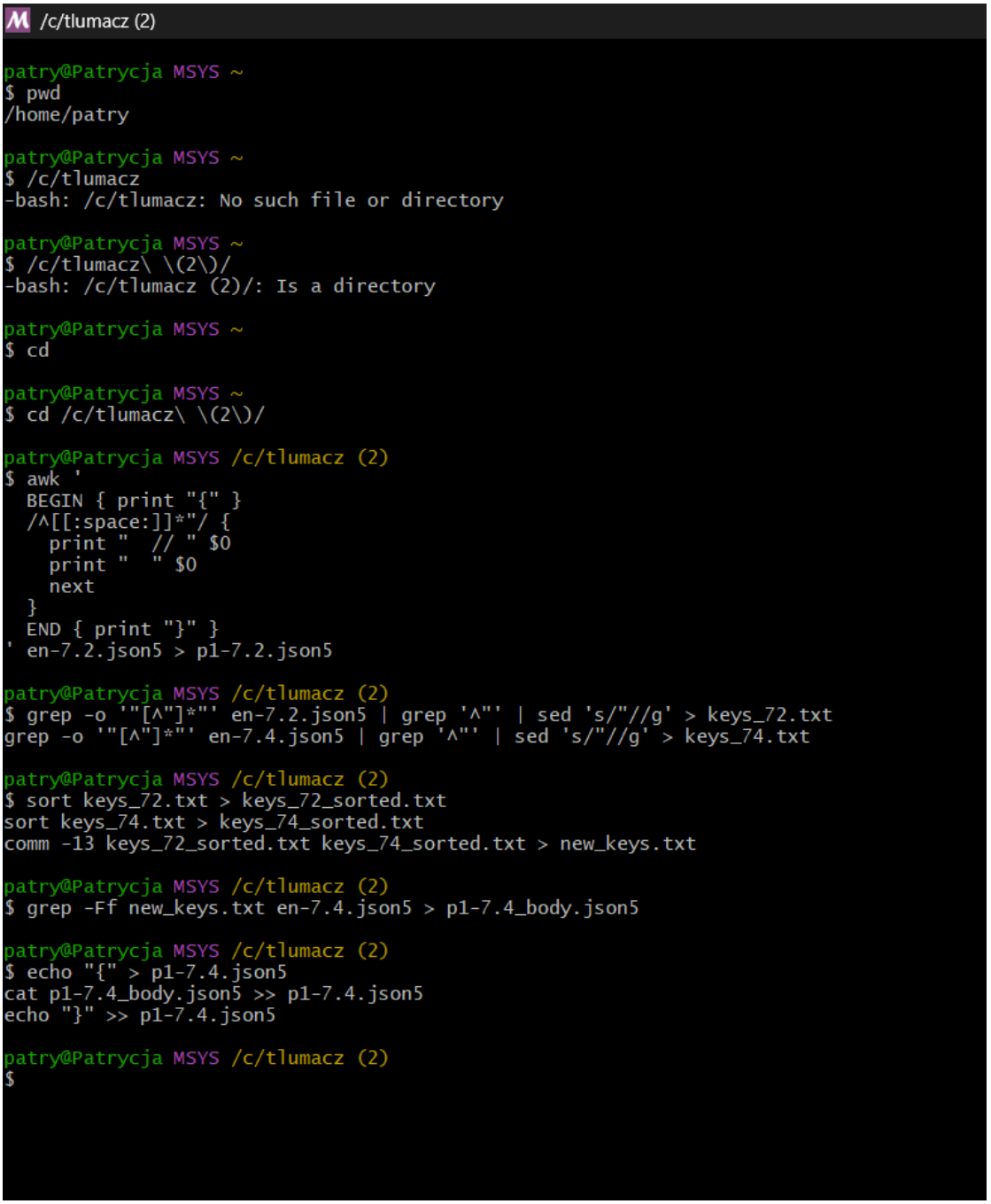
`print dateTime ";" steps ";" synced`

➤ Wypisuje przekształconą linię jako wiersz CSV.

```
0;0;0
0;0;0

patry@Patrycja MSYS ~
$ awk '
BEGIN { print "dateTime;steps;synced" }
/INSERT INTO/ {
    match($0, /\(([^)]+)\)/, arr)
    split(arr[1], vals, ",")
    dateTime = int(vals[1]/1000)
    steps = vals[2]+0
    synced = vals[3]+0
    print dateTime ";" steps ";" synced
}
' "/c/Users/patry/OneDrive/Dokumenty/csv/steps-2csv.sql" > steps-2csv.csv

patry@Patrycja MSYS ~
$ cp steps-2csv.csv /c/Users/patry/OneDrive/Pulpit/
```

Zad 6

- Dodaje komentarze `//` przed liniami z kluczami (czyli `"klucz": ...`)
- Format: plik JSON staje się „półkomentowanym” plikiem `.json5`
- Ułatwia odróżnienie przetłumaczonych i nieprzetłumaczonych kluczy
- `grep -o '["]*"[^"]*"'` — wydobywa wszystkie klucze (ciągi w cudzysłowie)
- `sed 's/"//g'` — usuwa wszystkie cudzysłowy
- Efektem są listy samych **nazw kluczy** w plikach `.txt` do porównania
- `sort` – przygotowuje dane do porównania
- `comm -13` – pokazuje **linie obecne tylko w `keys_74`**, a nie w `keys_72`
- Ostatecznie `plik` zawiera **nowe klucze dodane w wersji 7.4**
- Szuka wszystkich linii w `en-7.4.json5`, które zawierają **nowe klucze**.
- Wynik to `p1-7.4.json5.txt` z nowymi wpisami.



Zad 7

Tworzy foldery:

- `kopie-1` i `kopie-2` – do rozpakowania ZIP-ów
- `przerobione` – na przetworzone pliki JPG
- `file -b --mime-type "$src"` – wykrywa typ MIME (np. `image/jpeg`).
- Jeśli to obraz (`image/*`), przetwarza go:
    - zmniejsza do wysokości 720 pikseli (`-resize x720`),

    - ustawia rozdzielczość 96 DPI,
    - zapisuje jako `.jpg` w folderze `przerobione`.

- Tworzy archiwum `przerobione_zdjecia.zip` z zawartością folderu `przerobione`.

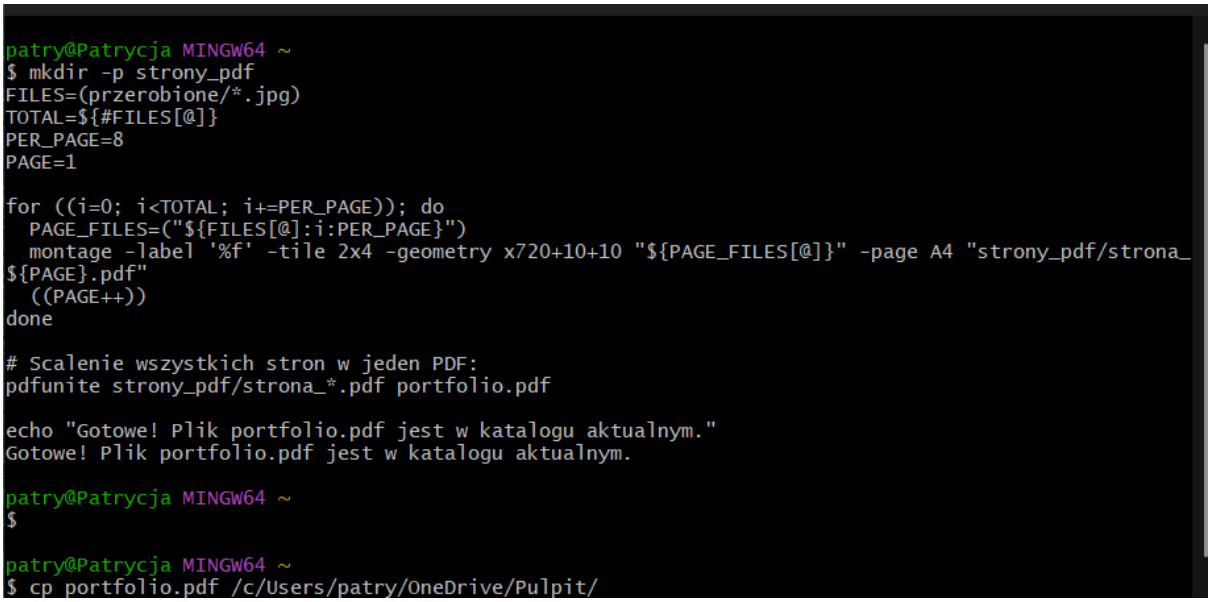
```
patry@Patrycja MSYS ~
$ #!/bin/bash

mkdir -p kopie-1 kopie-2 przerobione

unzip -o /c/Users/patry/Downloads/kopie-1.zip -d kopie-1
unzip -o /c/Users/patry/Downloads/kopie-2.zip -d kopie-2

for src in kopie-1/* kopie-2/*; do
    filetype=$(file -b --mime-type "$src")
    case "$filetype" in
        image/*)
            base=$(basename "$src")
            name="${base%.*}"
            dst="przerobione/${name}.jpg"
            echo "Przetwarzam: $base → $dst"
            /mingw64/bin/magick "$src" -resize x720 -density 96 -units PixelsPerInch "$dst"
            ;;
        *)
            echo "Pomijam nieobrazowy plik: $src"
            ;;
    esac
done

zip -r przerobione_zdjecia.zip przerobione

echo "☑ Gotowe! Wyniki w przerobione_zdjecia.zip"
Archive:  /c/Users/patry/Downloads/kopie-1.zip
  inflating: kopie-1/2010-05-22.zip
  inflating: kopie-1/2010-05-24.zip
  inflating: kopie-1/2010-05-29.zip
  inflating: kopie-1/2010-06-03.zip
  inflating: kopie-1/2010-06-08.zip
  inflating: kopie-1/2010-08-12.zip
  inflating: kopie-1/2010-08-17.zip
  inflating: kopie-1/2010-08-20.zip
  inflating: kopie-1/2011-01-05.zip
  inflating: kopie-1/2011-05-17.zip
  inflating: kopie-1/2011-05-22.zip
  inflating: kopie-1/2011-06-26.zip
  inflating: kopie-1/2011-07-24.zip
  inflating: kopie-1/2011-07-29.zip
  inflating: kopie-1/2011-08-05.zip
  inflating: kopie-1/2011-08-06.zip
  inflating: kopie-1/2011-08-11.zip
```

Zad 8

mkdir -p strony_pdf
Tworzy katalog strony_pdf, gdzie będą zapisane poszczególne strony PDF.
Zapisuje wszystkie obrazy JPG z katalogu `przerobione/` do tablicy `FILES`.

- `TOTAL`: ile jest plików JPG.
- `PER_PAGE`: ile obrazów ma znaleźć się na jednej stronie (8 obrazów = 2x4 kafelki).
- `PAGE`: licznik strony (numerowanie od 1).
- Dzieli obrazy na grupy po 8.

- `montage` (z ImageMagick) układa obrazy w siatkę 2x4 z opisami (nazwa pliku).
- Tworzy pliki PDF: `strona_1.pdf`, `strona_2.pdf`, itd.
- Łączy wszystkie `strona_*.pdf` w jeden plik: `portfolio.pdf`.



```
patry@Patrycja MINGW64 ~
$ mkdir -p strony_pdf
FILES=(przerobione/*.jpg)
TOTAL=${#FILES[@]}
PER_PAGE=8
PAGE=1

for ((i=0; i<TOTAL; i+=PER_PAGE)); do
  PAGE_FILES=("${FILES[@]:i:PER_PAGE}")
  montage -label '%f' -tile 2x4 -geometry x720+10+10 "${PAGE_FILES[@]}" -page A4 "strony_pdf/strona_
${PAGE}.pdf"
  ((PAGE++))
done

# Scalenie wszystkich stron w jeden PDF:
pdfunite strony_pdf/strona_*.pdf portfolio.pdf

echo "Gotowe! Plik portfolio.pdf jest w katalogu aktualnym."
Gotowe! Plik portfolio.pdf jest w katalogu aktualnym.

patry@Patrycja MINGW64 ~
$

patry@Patrycja MINGW64 ~
$ cp portfolio.pdf /c/Users/patry/OneDrive/Pulpit/
```

Zad 9

Użyte zostało polecenie:
for file in 20*.zip; do
Pętla **`for`** przechodzi po wszystkich plikach **`.zip`**, których nazwa zaczyna się od **`20`**.
rok=$(echo "$file" | cut -d'-' -f1)
Wyciąga **pierwszą część nazwy pliku**, oddzieloną myślnikiem (`-`), czyli **rok**.
miesiac=$(echo "$file" | cut -d'-' -f2)
Wyciąga **drugą część** (czyli **miesiąc**).
mkdir -p "$rok/$miesiac"
Tworzy foldery **`rok/miesiąc`**, np. **`2023/11`**
mv "$file" "$rok/$miesiac/"
Przenosi dany plik ZIP do folderu.
done

Zad.10

Kod dla wszystkich zdjęć jest zrobiony na podstawie tego
<
`responsive` sprawia, że zdjęcia układają się obok siebie w wierszu, a na małym ekranie automatycznie schodzą w dół

To **wewnętrzny kontener** dla obrazka i jego opisu.

To **link do pełnego zdjęcia**.
<img src="przerobione/adrien-olichon-3137064.jpg">
To **obrazek widoczny na stronie**.

adrien-olichon-3137064.jpg
To **opis pod obrazkiem**.

(cały kod jest na dysku w pliku **index.html**)