**Katarzyna Nowakowska** Inżynieria danych L02 II semestr

**Karolina Spurek**

**Sprawozdania do zadań projektowych**

Zadanie 1 - **Niesforne dane**

cat dane.txt (wyświetla plik).

dos2unix plik.txt (przekonwertowanie pliku na format Unix)

paste -d "\t" - - - <dane.txt (sortuje dane)

paste -d "\t" - - - <dane.txt > dane123123.txt (do pliku txt)

sed -i 'li x\ty\tz’ dane123123.txt

{dodaje nagłówki x y z; sed (stream editor – edycja pliku ) ; -i (zmiany dokonywane bezpośrednio bez zapisu do nowego pliku); li x\ty\tz (1 nr pierszej lini; i insert; x\ty\tz wstawiony tekst (\ tabulacja)}

Zadanie 2 - **Dodawanie poprawek**

diff -u lista.txt lista-pop.txt> lista.patch (porównuje oba pliki i zapisuje róznice w pliku patch)

dos2unix lista.txt (przekonwertowanie pliku na format Unix)

patch lista.txt < lista.patch (aplikowanie łatki)

md5sum lista-pop.txt lista.txt (md5sum sprawdza integralność miedzy nowo poprawionym plikiem lista.txt a lista-pop.txt. Ciągi liczb i cyfr zgadzają się w obu plikach co oznacza że zadanie zostało poprawnie wykonane).

Zadanie 3 - **Z CSV do SQL i z powrotem**

cat steps-2sql.csv (wklejanie pliku do msys)

tail -n +2 steps-2sql.csv | awk -F";" ‘{printf "INSERT INTO stepsData (time, intensity, steps) VALUES (%s, %s, %s);\n", $1, $2, $3}> steps-2sql-wynik.sql

{tail -n +2 - pomija nagtówek CSV.

awk -F";" - ustawia separator pól na ;.

printf - formatuje dane jako zapytania SQL. }

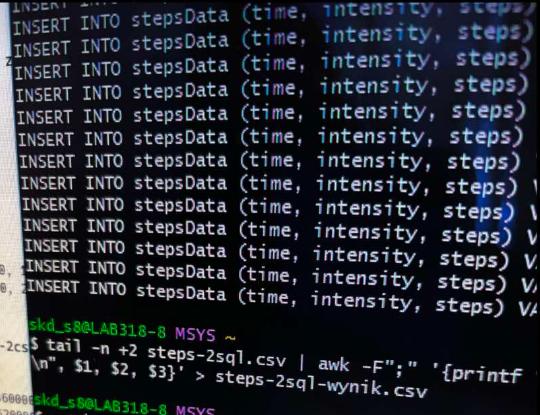
echo "dateTime; steps; synced"

grep "INSERT INTO" steps-2csv.sql | sed -E 's/.\*VALUES \(([0-9]+), ([0-9]+), ([0-9]+)\); /\1; \2; \3/' | awk -F";" '{printf "%d;%s; %s\n", $1/1000, $2, $3} > steps-2csv-wynik.csv

{grep - wybiera linie z INSERT INTO.

sed - wyciąga liczby z VALUES (...).

awk - dzieli przez 1000 (usuwa milisekundy) i zapisuje jako CSV. }



Zadanie 4 - **Marudny tłumacz**

echo “(“ >pl-7.2.4.json5 (tworzymy nowy plik)

grep –E ‘^\s\*”[^”]+:\s\*”[^”]+” en-7.2.json5 | sed –E ‘s/^(\s\*)”([^”]+)”:\s\*”([^”]+)”,?/\1W “\2”: “\3”, \n\1”2”: “\3”,/” >> pl-7.2.4.json5

{ grep wyszukuje tylko linie, które wyglądają jak wpisy w formacie JSON.

-E - używamy rozszerzonych wyrażeń regularnych.

^\s\*" - linia zaczyna się od dowolnej liczby spacji, potem cudzysłów.

[^”]+ - dowolny ciąg nie będący cudzysłowem.

:\s\*" - potem dwukropek i wartość w cudzysłowie.

sed - przekształca każdą linię JSON w komentarz + oryginalny wpis.

^(\s\*) - zapamiętuje w zmiennej \1 wcięcie.

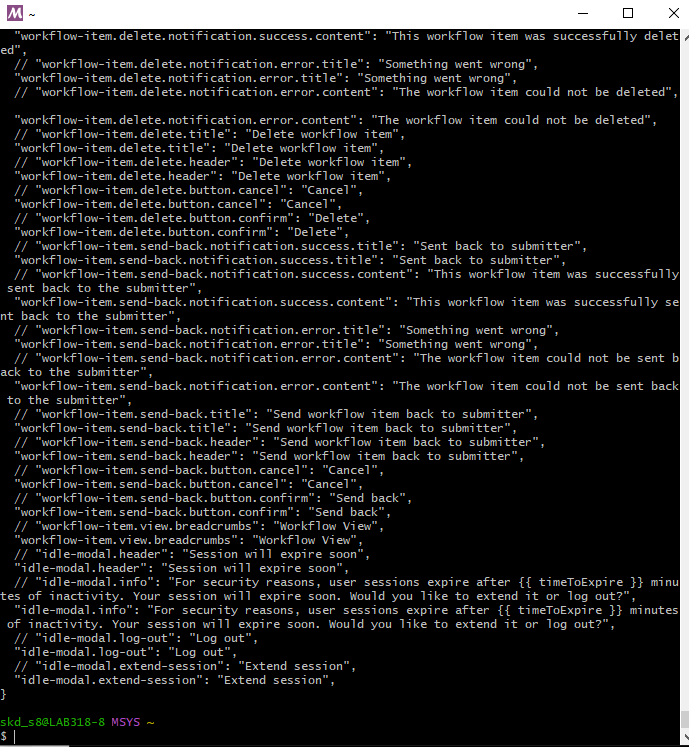
"([^"]+)" - zapamiętuje klucz, wartość.

,? - dopuszcza przecinek na końcu.

\1// "\2": "]3", - dodaje komentarz,

\n\1"\2": "\3", - oryginalna linia po komentarzu, z zachowaniem wcięcia. }

echo “}” >> pl-7.2.4.json5 (dopisuje przetworzone linie do wcześniej utworzonego pliku)



Zadanie 5 - **Fotografik gamoń**

unzip \\*.zip (rozpakowuje plik o zip)

rm kopie-2.zip kopie-1.zip (usuwa oba pliki o nazwach kopie-1 i kopie-2)

unzip \\*.zip (rozpakowuje plik zip)

rm \*.zip (usuwa wszystkie pliki zip)

for f in \*.png; do

magick "$f" "${f%.png}.jpg";

done (wykonuje tą operacje na każdym pliku .png i konwertuje go na plik jpg, ${f%.png}.jpg zamienia nazwę na taką z końcówką .jpg).

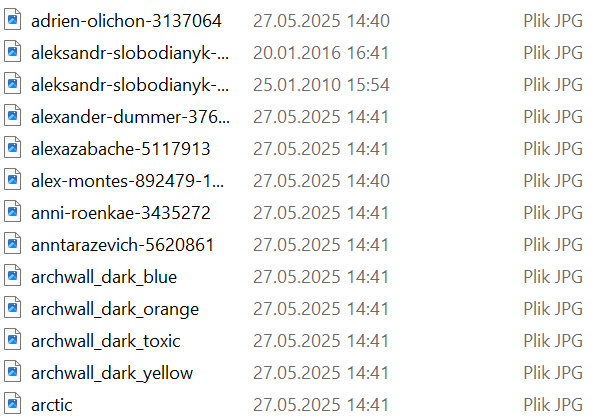
rm \*.png (usuwa wszystkie pliki png)

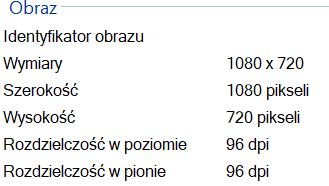
mkdir -p zdjecia (tworzy katalog )

for f in \*.jpg; do

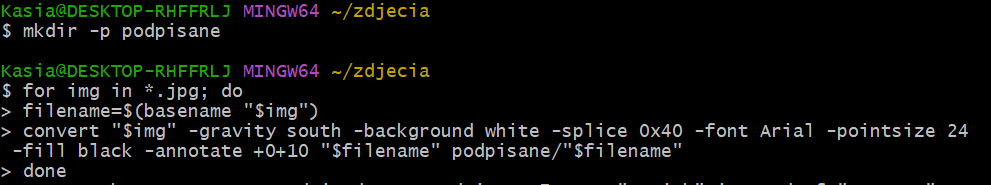
magick "$f" -resize x720 -units PixelsPerInch -density 96 "zdjecia/$f";

done (zaznacza każdy plik jpg, ustawia wysokość każdego zdjęcia do 720 pikseli, zamienia jednostkę na "PixelsPerInch" i rozdzielczość na 96"dpi")





Zadanie 6 - **Wszędzie te PDF-y**

{mkdir –tworzy nowy katalog;

for img in ~/photos/\*.jpg; do ... done - rozpoczyna pętlę w Bashu, która przejdzie po każdym pliku .jpg w katalogu zdjecia;

filename=$(basename "$img") - wyciąga samą nazwę pliku i zapisuje ją w zmiennej filename;

convert "$img" ... ~/photos/labeled/"$filename" - wywołuje narzędzie ImageMagick convert, które edytuje jeden obraz i zapisuje wynik pod nową ścieżką;}

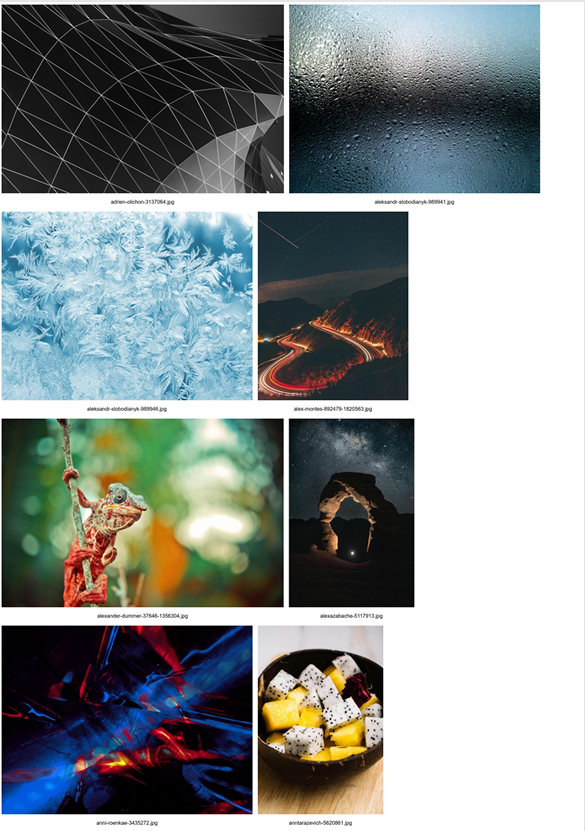
montage podpisane/\*.jpg -tile 2x4 -geometry 360x220+10+10 -background white -gravity center portfoli\_zdjec.pdf

{montage podpisane/\*.jpg - uruchamia program montage z ImageMagick i przekazuje mu wszystkie obrazy JPG z katalogu podpisane;

-tile 2x4 - ustala, że na jednej stronie powstanie siatka z 2 kolumn i 4 wierszy, czyli razem maks. 8 zdjęć na stronie;

-geometry 360x220+10+10 - mówi, jak duże ma być „miejsce” dla każdego obrazka: 360 pikseli szerokości i 220 pikseli wysokości, +10+10 to odstęp między obrazami: 10 pikseli poziomo i 10 pikseli pionowo;

portfolio\_zdjec.pdf - nazwa pliku wynikowego, do którego montage zapisze całość.}



Zadanie 7 - **Porządki w kopiach zapasowych**

cd kopie | (wchodzi w katalog kopie)

cd | (wraca do poprzedniego katalogu)

bash sortowanie.bat | (Otwiera i uruchamia skrypt batchowy zawarty w pliku .bat)

{Zawartość pliku sortowanie.bat:

cd kopie

for file in \*.zip; do

y=$(echo "$file" | cut -d'-' -f1)

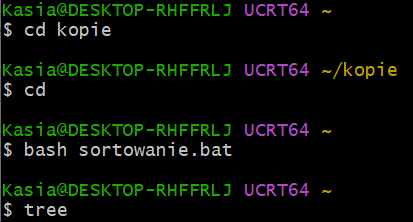
m=$(echo "$file" | cut -d'-' -f2)

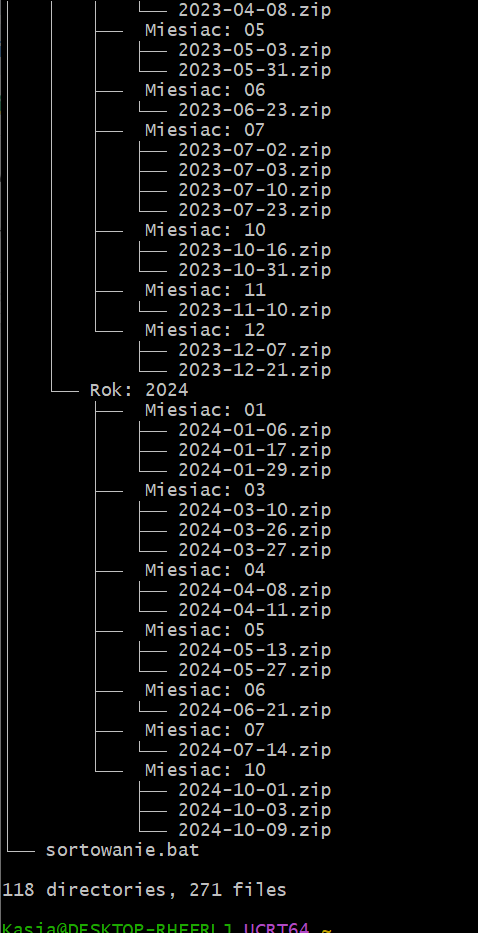
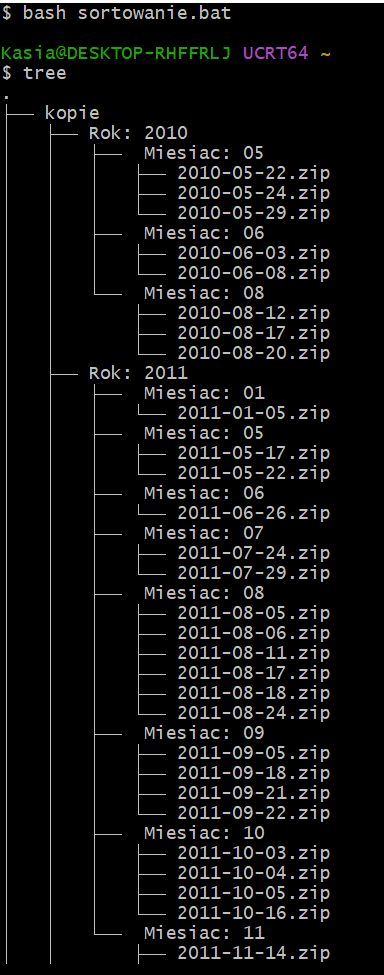
mkdir -p "Rok: $y/ Miesiac: $m"

mv "$file" "Rok: $y/ Miesiac: $m/"

done}

tree ( wyświetla liste katalogów, listę plików z wcięciami w głąb)

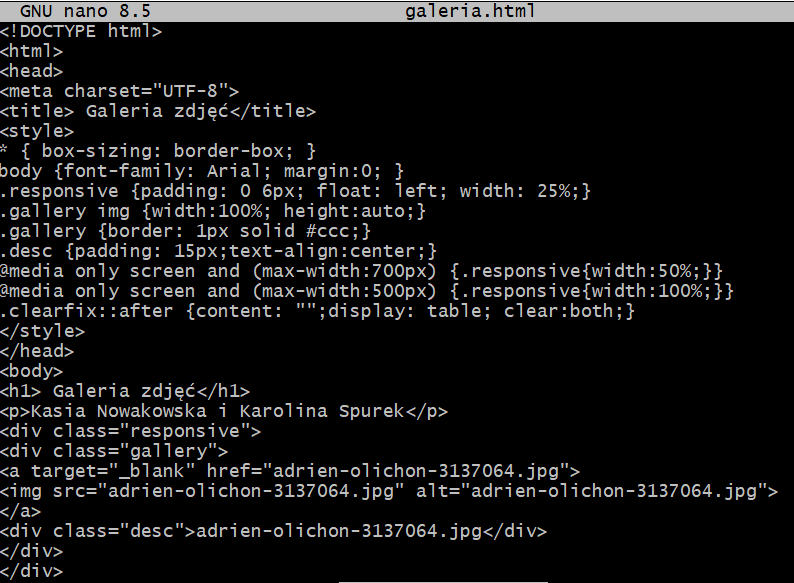




Zadanie 8 - **Galeria dla grafika**

nano galeria.html (tworzymy plik galeria i otwieramy edytor nano)

Wpisujemy w nim poniższy kod:



Jest to tworzenie szkieletu strony html i dodanie css. Zapisujemy i wychodzimy (Ctrl+O, Enter, Ctrl+X).

Następnie w terminalu wpisujemy pętle for.



Przeszukuje katalog w poszukiwaniu wszystkich plików pasujących do wzorca \*.jpg (czyli wszystkich obrazków JPG). Za każdym razem zmienna img przyjmuje nazwę jednego z tych plików. Wszystko, co jest między do a done, będzie wykonane dla każdego pliku osobno.

cat <<EOF ... EOF – pozwala podać wieloliniowy tekst bezpośrednio do programu cat

>> index.html oznacza, że tekst, który wytworzy cat, zostanie dopisany na końcu pliku galeria.html. Dzięki temu do pliku HTML dopisywany jest za każdym razem cały blok kodu HTML dla pojedynczego obrazka. W tym bloku HTML są użyte zmienne bashowe (np. $img), które są rozwijane – czyli wstawiana jest faktyczna nazwa pliku obrazu:

* w href="$img" → link do pełnego rozmiaru zdjęcia,
* w src="$img" → źródło obrazka w galerii,
* w alt="$img" i <div class="desc">$img</div> → nazwa pliku jako podpis pod obrazkiem.

echo '<div class="clearfix"></div></body></html>' >>galeria.html

{echo '...' >> galeria.html – po prostu dodaje ostatni fragment HTML do pliku;

<div class="clearfix"></div> – element CSS do „czyszczenia” floatów, żeby kontener objął wszystkie elementy .responsive.

</body></html> – zamyka dokument HTML.

>> oznacza dopisanie na końcu pliku, a nie nadpisanie.}

