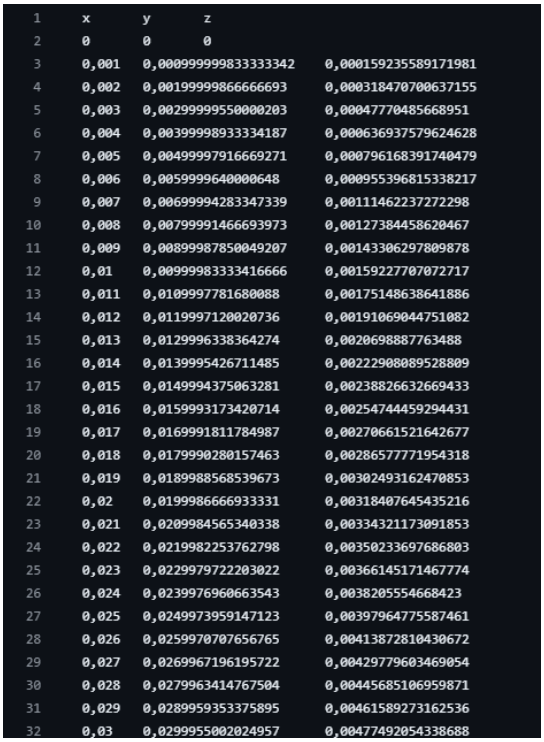
Maciej Krawczyk & **Jakub Piątek** – Sprawozdanie PROJEKT **programowanie** ID Stacjonarne I rok

(Zad. 1 i 2 uznaliśmy za wstęp do materiału, który był instrukcją zapoznawczą, stąd rozpoczęcie numeracji od 3)

Zad. 3. *Niesforne dane* (por. rys. 1 i 2)

Wykonanie:

1. Najpierw komendą unzip rozpakowujemy plik w zipie
2. Potem wklejamy trzy kolumny myślnikami do pliku tekstowego
3. Następnie dodajemy nagłówki x y z do pliku (echo)
4. Na koniec odpowiednio dzielimy całość, żeby ładnie wyglądało w kolumnach
5. Ten kod można byłoby napisać w jednej linii, ale tutaj tak rozpisaliśmy



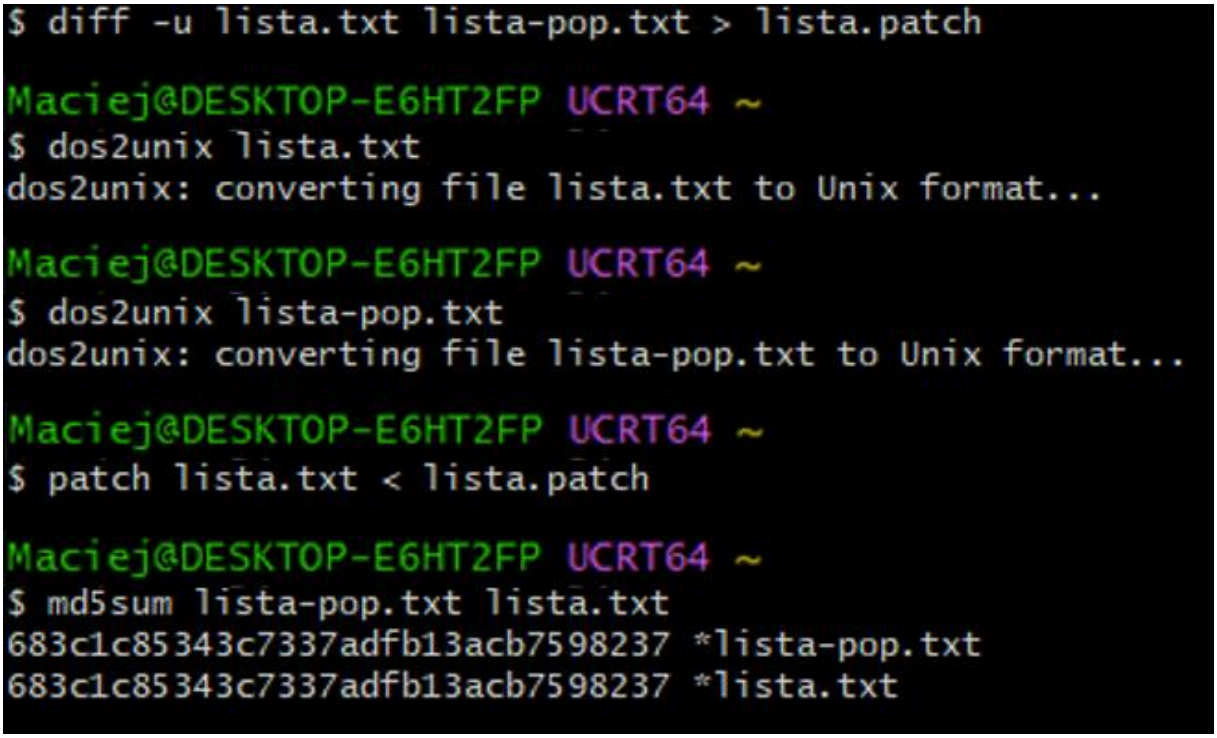
	x	y	z
1	0	0	0
2	0,001	0,000999999833333342	0,000159235589171981
3	0,002	0,00199999866666693	0,000318470700637155
4	0,003	0,00299999550000203	0,00047770485668951
5	0,004	0,00399998933334187	0,000636937579624628
6	0,005	0,00499997916669271	0,000796168391740479
7	0,006	0,0059999640000648	0,000955396815338217
8	0,007	0,00699994283347339	0,00111462237272298
9	0,008	0,00799991466693973	0,00127384458620467
10	0,009	0,00899987850049207	0,00143306297809878
11	0,01	0,00999983333416666	0,00159227707072717
12	0,011	0,0109997781680088	0,00175148638641886
13	0,012	0,0119997120020736	0,00191069044751082
14	0,013	0,0129996338364274	0,0020698887763488
15	0,014	0,0139995426711485	0,00222908089528809
16	0,015	0,0149994375063281	0,00238826632669433
17	0,016	0,0159993173420714	0,00254744459294431
18	0,017	0,0169991811784987	0,00270661521642677
19	0,018	0,0179990280157463	0,00286577771954318
20	0,019	0,0189988568539673	0,00302493162470853
21	0,02	0,0199986666933331	0,00318407645435216
22	0,021	0,0209984565340338	0,00334321173091853
23	0,022	0,0219982253762798	0,00350233697686803
24	0,023	0,0229979722203022	0,00366145171467774
25	0,024	0,023997696066543	0,0038205554668423
26	0,025	0,0249973959147123	0,00397964775587461
27	0,026	0,0259970707656765	0,00413872810430672
28	0,027	0,0269967196195722	0,00429779603469054
29	0,028	0,027996341476750 4	0,00445685106959871
30	0,029	0,0289959353375895	0,00461589273162536
31	0,03	0,0299955002024957	0,00477492054338688

Rys. 1 i 2. Kod w MSYS2 i jego wynik

Zad. 4. *Dodawanie poprawek* (por. rys. 3)

Wykonanie:

1. Na początku musielśmy zainstalować kilka pakietów dzięki komendzie pacman (szczegółowe krok po kroku wpisywanie linijek kodu wraz z instalacjami w github'ie)
2. Zastosowaliśmy polecenie diff –u, które porównuje dwa pliki tekstowe i wyświetla różnice w tzw. formacie unifikowanym (ang. unified diff)
3. Następnie konwertowaliśmy lista.txt z formatu DOS/Windows (CRLF) na format UNIX (LF) (dzięki poleceniu dos2unix)
4. Podobnie z lista-pop.txt
5. Potem za pomocą polecenia *patch* modyfikujemy plik lista.txt (polecenie *patch* automatycznie modyfikuje pliki, stosując zmiany opisane w pliku .patch lub .diff, który został wygenerowany)
6. Potem polecenie *md5sum* (polecenie md5sum służy do obliczania i sprawdzania sumy kontrolnej MD5 pliku lub danych wejściowych) – dzięki temu otrzymaliśmy wynik



Rys. 3. Kod w MSYS2 i jego wynik

Zad. 5. *Z CSV do SQL i z powrotem* (por. rys. 4)

Wykonanie:

1. Najpierw zastosowaliśmy *tail steps-2sql.csv*, który wyświetla ostatnie linie pliku steps-2sql.csv
2. *awk -F ";"* ustawia średnik jako separator pól w pliku CSV.
3. *{print "INSERT INTO stepsData(time,intensity,steps) Values (\"" $1 "\",\"" $2 "\",\"" $3 "\")"}* generuje polecenie SQL INSERT INTO dla każdej linii CSV, wstawiając wartości z trzech kolumn:
    a. $1 → (time),
    b. $2 → (intensity),
    c. $3 → (steps).
4. Polecenie *echo time; intensity; steps* wypisuje nagłówek kolumn.
5. cat steps-2csv.sql — wczytuje zawartość pliku SQL.
6. sed -e 's/INSERT INTO ... VALUES (//g' — usuwa fragment INSERT INTO stepsData (dateTime, steps, synced) VALUES (.
7. sed -e 's/)//g' — usuwa zamykające nawiasy).
8. sed -e "s/'//g" — usuwa apostrofy ', które otaczają wartości tekstowe w SQL.
9. sed -e 's/;//g' — usuwa średniki ; na końcu poleceń SQL.
10. sed -e 's/,/ /g' — zamienia przecinki na spacje, aby rozdzielić dane.
11. W ten sposób rozwiązaliśmy to zadanie

```
$ tail steps-2sql.csv |awk -F ";" '{print "INSERT INTO stepsData(time,intensity,steps) Values ("$1","$2","$3")"}' <steps-2sql.csv
INSERT INTO stepsData(time,intensity,steps) Values (time,intensity,steps)
INSERT INTO stepsData(time,intensity,steps) Values (1562001120,19,0)
INSERT INTO stepsData(time,intensity,steps) Values (1562001180,23,0)
INSERT INTO stepsData(time,intensity,steps) Values (1562001240,13,0)
INSERT INTO stepsData(time,intensity,steps) Values (1562004900,0,0)
INSERT INTO stepsData(time,intensity,steps) Values (1562004960,53,26)
INSERT INTO stepsData(time,intensity,steps) Values (1562005020,57,15)
INSERT INTO stepsData(time,intensity,steps) Values (1562005080,22,0)
INSERT INTO stepsData(time,intensity,steps) Values (1562005140,44,0)
INSERT INTO stepsData(time,intensity,steps) Values (1562005200,30,0)
INSERT INTO stepsData(time,intensity,steps) Values (1562005260,41,0)
INSERT INTO stepsData(time,intensity,steps) Values (1562005320,9,0)
INSERT INTO stepsData(time,intensity,steps) Values (1562005380,41,0)
INSERT INTO stepsData(time,intensity,steps) Values (1562005440,72,24)

Maciej@DESKTOP-E6HT2FP UCRT64 ~
$ echo time; intensity ; steps |cat steps-2csv.sql | sed -e 's/INSERT INTO stepsData (dateTime, steps, synced) VALUES (//g' | sed -e 's/)//g' | sed -e 's/;//g' | sed -e 's/,/;/g' | sed -e 's/ //g'
```

Rys. 4. Kod w MSYS2 i jego wynik

Zad. 6. *Marudny tłumacz* (por. rys. 5)

Wykonanie:

1. Polecenie *cat en-7.2.json5* wczytuje zawartość pliku JSON5.
2. awk -F ':' '{print $1}' – tutaj dzieli linie na części wg dwukropka (:) i wypisuje pierwszą część (czyli klucz w strukturze JSON).
3. Następnie użyliśmy sed 's/"//g' — usuwa wszystkie cudzysłowy ", które zwykle otaczają klucze w JSON.
4. Potem sed 's/[{}]//g' , który usuwa nawiasy klamrowe {}.
5. Następnie awk NF — usuwa puste linie (NF = number of fields; jeśli 0, to linia jest pusta).
6. sed 's/ //g' — usuwa wszystkie spacje.
7. > var.txt — zapisuje wynik do pliku var.txt
8. Punkty 1-7 to była pierwsza linia kodu, którą skonstruowaliśmy
9. Teraz, trzeba wyszukać, czyli grep.
10. *grep -f var.txt -v en-7.4.json5* — szuka w pliku en-7.4.json5 linii, które nie zawierają żadnego z wzorców (kluczy) znajdujących się w var.txt.
    (-v oznacza negację — pokazuje linie, które nie pasują).
11. awk NF — usuwa puste linie (NF – Number of Fields; jeśli linia pusta, to NF=0).
12. sed 's/"//g' — usuwa wszystkie cudzysłowy (").
13. sed 's/[{}]//g' — usuwa nawiasy klamrowe {}
14. To drugie polecenie ma na celu wyciągnięcie z pliku en-7.4.json5 tych elementów, które nie są na liście kluczy w var.txt.

```
Maciej@DESKTOP-E6HT2FP UCRT64 ~
$ cat en-7.2.json5 | awk -F ':' '{print $1}' | sed 's/"//g' | sed 's/[{}]//g' | awk NF | sed 's/ //g' > var.txt

Maciej@DESKTOP-E6HT2FP UCRT64 ~
$ grep -f var.txt -v en-7.4.json5 | awk NF | sed 's/"//g' | sed 's/[{}]//g'
researcher.profile.public.visibility : PUBLIC,
researcher.profile.status: Status:,
researcherprofile.claim.not-authorized: You are not authorized to claim this item. For more details contact the
researcherprofile.error.claim.body : An error occurred while claiming the profile, please try again later,
researcherprofile.success.claim.body : Profile claimed with success,
person.orcid.sync.setting: ORCID Synchronization settings,
person.orcid.registry.queue: ORCID Registry Queue,
person.orcid.registry.auth: ORCID Authorizations,
home.recent-submissions.head: Recent Submissions,
```

Rys. 5. Kod w MSYS2 i jego wynik

Zad. 7. *Fotografik gamoń* (por. rys. 6)

Wykonanie:

1. Najpierw rozpakowujemy plik zip
2. Następnie usuwamy zip (komenda rm …)
3. Ponownie rozpakowujemy
4. Komendą rm .zip usuwamy wszystkie pozostałe pliki zip
5. Teraz ważne: konwertujemy z PNG do JPG (konstrukcja 5. linijki *for f in...*)
6. Usuwamy pliki PNG *rm .png*
7. Tworzymy katalog dzięki *mkdir –p* na przekonwertowane już obrazy
8. Na koniec zmieniamy rozmiar i przenosimy JPG (to jest ostatnia linijka kodu)
9. **zip -r Przekonwertowane.zip Przekonwertowane** → ta komenda pozwala jeszcze na stworzenie archiwum ZIP

```
unzip *.zip
rm kopie-2.zip kopie-1.zip
unzip *.zip
rm .zip
for f in.png; do magick "$f" "${f%.png}.jpg"; done
rm .png
mkdir -p Przekonwertowane
for f in.jpg; do magick "$f" -resize x720 -units PixelsPerInch -density 96 "Przekonwertowane/$f"; done
```

Rys. 6. Kod w MSYS2 i jego wynik

Zad. 8. *Wszędzie te PDF-y* (por. rys. 7)

Wykonanie:

1. Zastosowaliśmy polecenie *montage *.jpg* , które używa narzędzia ImageMagick do stworzenia kolażu z wszystkich plików JPG w bieżącym katalogu (*.jpg).
2. Potem trzeba wpisać -tile 2x4 - układa obrazy w siatkę o 2 kolumnach i 4 wierszach (łącznie do 8 obrazów na jednej stronie PDF).
3. Następnie Jakub dodał -geometry 400x200+30+40 , co ustanawia rozmiar każdego obrazu na 400x200 pikseli.
4. +30+40 — odstępy między obrazami:
    a. 30 pikseli w poziomie (między kolumnami),
    b. 40 pikseli w pionie (między wierszami). **Żeby ładnie wyglądało**
5. Potem zastosowaliśmy -font C:/Windows/fonts/arial.ttf, czyli użycie czcionki Arial z systemu Windows do etykiet.

6. Ważna etykieta: -label '%f' dodaje pod każdym obrazem etykietę z nazwą pliku (%f = filename).
7. Aby czcionka była 14 -pointsize 14 - dla etykiet to 14 punktów. (Może być inna)
8. -background white - tło całego kolażu będzie białe.
9. Skoro miała być kartka A4, to wpisaliśmy następnie -page A4 - rozmiar strony PDF to standardowe A4.
10. -density 150 - ustawiliśmy rozdzielczość dokumentu PDF na 150 DPI, co jest średnią jakością — dobre do podglądu i druku tekstowego.
11. output2.pdf - nazwa pliku wyjściowego to output2.pdf, czyli gotowy PDF z kolażem.
12. FINITO Mamy nadzieję, że jest dobrze ☺

```
Maciej@DESKTOP-E6HT2FP UCRT64 ~/Zadanie 7/Przekonwertowane
$ montage *.jpg -tile 2x4 -geometry 400x200+30+40 -font C:/Windows/fonts/arial.ttf -label '%f' -pointsize 14 -background white -p
age A4 -density 150 output2.pdf
```

Rys. 7. Kod w MSYS2 i jego wynik

Zad. 9. *Porządki w kopiach zapasowych* (por. rys. 8 i 9)

Wykonanie:

1. Najpierw tworzymy katalog roboczy *kopie*
2. Potem przechodzimy do niego *cd*
3. Następnie rozpakowujemy zip
4. Wracamy z powrotem
5. Włączamy skrypt sortujący dzięki *bash*
6. Wracamy do folderu z plikami *cd kopie*
7. Potem tworzymy pętlę for, w której sortujemy pliki ZIP do folderów wg roku i miesiąca
8. Na koniec chcemy pokazać strukturę folderów za pomocą *tree* – wyświetla posortowane wg roku i miesiąca

```
mkdir kopie
cd kopie
unzip .zip
cd
bash SortowanieDrzewkowe.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/"
```

Rys. 8. Kod w MSYS2

Rys. 9. Efekt działania

Zad. 10. *Galeria dla grafika* (por. rys. 10)

Wykonanie:

1. Wpierw trzeba wejść do katalogu, gdzie są obrazy (przekonwertowane na JPG)
2. Potem stworzyliśmy dzięki poleceniu touch obrazy_html.sh
3. Weszliśmy w tryb edycji skryptu dzięki naszemu ulubionemu *nano* (bardzo lubimy tę komendę ☺) i użyliśmy polecenia nano obrazy_html.sh
   Wpisaliśmy:

```
#!/bin/bash
echo ’<div class=”responsive”>’>galeria.html
echo „<header>”>> galeria.html
echo „<h1>Te galerie zrobilismy na podstawie obrazkow.</h1>”>> galeria.html
echo „</header>”>> galeria.html
for file in *.jpg; do
    echo ‘<div class = „gallery”>’>>galeria.html
    echo „ <a target=\”_blank\”href=\”$file\”>”>>galeria.html
    echo „ <img src=\”$file\”>”>> galeria.html
    echo „ </a>”>> galeria.html
    echo „<div class=\”desc\”>$file</div>”>> galeria.html
    echo „</div>”>> galeria.html
done
echo ‘</div>’>> galeria.html
```

Rys. 10. Wpisany kod

5. Zapisaliśmy zmiany, a potem wychodzimy z nano
6. Daliśmy także uprawnienia do skryptu: *chmod u+x galeria_html.sh*
7. Po ich nadaniu, uruchamiamy polecenie ./obrazy_html.sh
8. Powinno się pojawić plik galeria.html, gdzie będą nagłówki wg wzoru i przede wszsystkim obrazki
9. *Tutto è fatto*

Pozdrawiamy

Maciej Krawczyk

Jakub Piątek

ID, I ROK Stacjonarne