

POLITECHNIKA ŚWIĘTOKRZYSKA
WYDZIAŁ ZARZĄDZANIA I MODELOWANIA KOMPUTEROWEGO

Kierunek – Inżynieria danych

Studia pierwszego stopnia

Sprawozdanie z zadań projektowych

Aleksandra Arczewska

Numer albumu: 96934

Karina Stokłosa

Numer albumu: 97834

Opiekun pracy:
dr Sławomir Koczubiej

Kielce 2025

3. Niesforne dane

WYKONANIE:

- `unzip dane.zip`
- `dos2unix dane.txt`
- `paste -d "\t" - - - <dane.txt`
- `paste -d "\t" - - - <dane.txt > nsfr_dane.txt`
- `sed -i '1i x\ty\tz' nsfr_dane.txt`

OBJAŚNIENIA:

- `unzip dane.txt` - rozpakowanie pliku
- `dos2unix dane.txt` - przekonwertowanie pliku na format Unix
- `paste -d "\t" - - - <dane.txt` - sortuje dane
- `paste -d "\t" - - - <dane.txt > nsfr_dane.txt` - do nsfr_dane.txt
- `sed -i` - edycja pliku bezpośrednio — bez tworzenia nowego pliku tymczasowego.
- `1i` – numer linii i insert (wstaw)
- `x\ty\tz` -to tekst, który ma zostać wstawiony
- `\t` – tabulator

REZULTAT:

W folderze dane.zip znajdowały się pliki w tym plik pliku dane.txt gdzie znajdowały się dane liczbowe w jednej kolumnie. Celem zadania było rozpakowanie pliku oraz przekształcenie tych danych tak, aby znajdowały się w trzech kolumnach. Dodatkowo, w pierwszym wierszu pliku należało umieścić nagłówki kolumn, np. x, y, i z. Pozwala to na lepszą czytelność i organizację danych.

```
dom@DESKTOP-60868JA MSYS ~
$ head nsfr_dane.txt
x      y      z
0      0      0
0,001  0,00099999833333342  0,000159235589171981
0,002  0,00199999866666693  0,000318470700637155
0,003  0,00299999550000203  0,00047770485668951
0,004  0,00399998933334187  0,000636937579624628
0,005  0,00499997916669271  0,000796168391740479
0,006  0,0059999640000648  0,000955396815338217
0,007  0,00699994283347339  0,00111462237272298
0,008  0,00799991466693973  0,00127384458620467
```

4. Dodawanie poprawek

WYKONANIE:

- `unzip lista.zip`
- `dos2unix lista.txt lista-pop.txt`
- `diff -u lista.txt lista-pop.txt >roznice.patch`
- `patch lista.txt <lista.patch`
- `md5sum lista.txt lista-pop.txt`

OBJAŚNIENIA:

- `unzip lista.zip` – rozpakowanie pliku
- `Dos2unix lista.txt lista-pop.txt` - konwersja pliku tekstowego z formatu DOS/Windows do formatu Unix/Linux
- `diff -u lista.txt lista-pop.txt > lista.patch` -tworzy łątkę
- `patch lista.txt < lista.patch` – aktualizuje plik
- `md5sum lista.txt lista-pop.txt` – sprawdza poprawność aktualizacji

REZULTAT:

W zadaniu porównaliśmy dwa pliki tekstowe znajdujące się w folderze lista.zip: lista.txt i lista-pop.txt. Stworzyliśmy na tej podstawie **łątkę roznice.patch**, która zawiera różnice między tymi plikami. Następnie zastosowaliśmy łątkę do pliku lista.txt, co pozwoliło go zaktualizować do wersji lista-pop.txt. Na końcu, dzięki komendzie md5sum dla obu plików, potwierdziliśmy, że aktualizacja przebiegła prawidłowo — oba pliki są teraz identyczne.

```
dom@DESKTOP-60868JA MSYS ~  
$ md5sum lista.txt lista-pop.txt  
683c1c85343c7337adfb13acb7598237 *lista.txt  
683c1c85343c7337adfb13acb7598237 *lista-pop.txt
```

5. Z CSV do SQL i z powrotem

WYKONANIE:

- `unzip csv.zip`
- `awk -F “;” ‘NR > 1 { printf “INSERT INTO stepsData (time, intensity, steps) VALUES (%s, %s, %s);\n”, $1, $2, $3 }’ steps-2sql.csv > steps-2sql.sql`
- `cat steps-2csv.sql | grep ‘VALUES’ | awk -F‘(,)’ ‘{if ($6 ~ /000$/) {sub (/000$/,””, $6); gsub (/^ +/, ””, $7); gsub (/^ +/, ””, $8); print $6”,”$7”,”$8}’ > steps-2csv.csv`
- `sed -i ‘1i dateTime;steps;synced’ steps-2csv.csv`

OBJAŚNIENIA:

- `unzip csv.zip` - rozpakowanie pliku
- `Awk` - narzędzie do przetwarzania pliku
- `-F “;”` - ustawia separator pól na średnik
- `NR > 1` – pomija pierwszą linię
- `cat steps-2csv.sql` - czyta plik SQL
- `grep 'VALUES'` - wybiera tylko linie zawierające dane INSERT INTO.
- `awk -F‘(,)’` - dzieli wiersz na pola na podstawie nawiasów i przecinków.
- `f ($6 ~ /000$/) {sub (/000$/,””, $6)}` - usuwa ostatnie 3 zera, czyli konwertuje milisekundy do sekund.
- `gsub (/^ +/, ””, $7) i $8` - usuwa ewentualne spacje z kolejnych pól.
- `sed -i` - modyfikuje plik w miejscu (bez tworzenia nowego).
- `dateTime;steps;synced` - nagłówek zgodny z formatem CSV.

REZULTAT:

Dane z plików CSV i SQL zostały przekształcone między sobą zgodnie z wymaganym formatem, z konwersją znaczników czasu i przygotowaniem do importu lub dalszej analizy. W rezultacie otrzymano dwa pliki: `steps-2sql.sql` z zapytaniami SQL typu INSERT oraz `steps-2csv.csv` w formacie CSV z nagłówkiem i timestampami w sekundach.

```
dom@DESKTOP-60868JA MSYS ~
$ head steps-2sql.sql
INSERT INTO stepData (time, intensity, steps) VALUES(1562001120, 19, 0);
INSERT INTO stepData (time, intensity, steps) VALUES(1562001180, 23, 0);
INSERT INTO stepData (time, intensity, steps) VALUES(1562001240, 13, 0);
INSERT INTO stepData (time, intensity, steps) VALUES(1562004900, 0, 0);
INSERT INTO stepData (time, intensity, steps) VALUES(1562004960, 53, 26);
INSERT INTO stepData (time, intensity, steps) VALUES(1562005020, 57, 15);
INSERT INTO stepData (time, intensity, steps) VALUES(1562005080, 22, 0);
INSERT INTO stepData (time, intensity, steps) VALUES(1562005140, 44, 0);
INSERT INTO stepData (time, intensity, steps) VALUES(1562005200, 30, 0);
INSERT INTO stepData (time, intensity, steps) VALUES(1562005260, 41, 0);
```

```
dom@DESKTOP-60868JA MSYS ~
$ head steps-2csv.csv
1562004600;41;0
1562005200;65;0
1562005800;86;0
1562006400;163;0
1562007000;234;0
1562007600;258;0
1562008800;385;0
1562009400;757;0
1562010000;829;0
1562041200;41;0
```

6. Marudny tłumacz

WYKONANIE:

- unzip tłumacz.zip
- awk '{ if (\$0 ~ /^ *//) { print "/" \$0}; print \$0} else {print \$0 }'
en-7.2.json5 > pl-7.2.txt
- sort en-7.2.json5 > sorted-7.2.txt
- sort en-7.4.json5 > sorted-7.4.txt
- comm -13 sorted-7.2.txt sorted-7.4.txt > new-entries.txt
- awk '{ if(\$0 ~ /^ *//) { print "/" \$0}; print \$0}}' new-entries.txt > pl-7.4.json5

OBJAŚNIENIA:

- unzip tłumacz.zip - rozpakowywuje plik
- awk - czyta plik linia po linii

- `if($0 ~ /^ *"/)` - jeśli linia zaczyna się od cudzysłowu `"`, czyli jest wpisem JSON (klucz-wartość)
- `print "/" $0` - wypisuje tę samą linię, ale z przedrostkiem `//` — czyli jako komentarz
- `print $0` - następnie wypisuje oryginalną linię
- `else { print $0 }` linie, które nie zaczynają się od `"`, np. nawiasy `{}`, pozostają bez zmian
- `Sort` - sortuje linie plików
- `comm` - porównuje dwa posortowane pliki linia po linii

REZULTAT:

Zadanie polegało na zdublowaniu i skomentowaniu wpisów z pliku `en-7.2.json5` oraz wyodrębnieniu nowych fraz z `en-7.4.json5` do tłumaczenia. W efekcie powstały dwa pliki `pl-7.2.json5` i `pl-7.4.json5`, które ułatwiają tłumaczowi pracę, zawierając odpowiednio pełny zestaw i tylko nowe teksty do przetłumaczenia.

```
dom@DESKTOP-60868JA MSYS ~
$ head pl-7.2.json5
{
// "401.help": "You're not authorized to access this page. You can use the button below to get back to the home page.",
// "401.link.home-page": "Take me to the home page",
// "401.unauthorized": "unauthorized",
// "403.help": "You don't have permission to access this page. You can use the button below to get back to the home page.",
```

```
dom@DESKTOP-60868JA MSYS ~
$ head pl-7.4.json5
// "access-status.embargo.lisitelement.badge": "Embargo",
"access-status.embargo.lisitelement.badge": "Embargo",
// "access-status.metadata.only.lisitelement.badge": "Metadata only",
"access-status.metadata.only.lisitelement.badge": "Metadata only",
// "access-status.open.access.lisitelement.badge": "Open Access",
"access-status.open.access.lisitelement.badge": "Open Access",
// "access-status.restricted.lisitelement.badge": "Restricted",
"access-status.restricted.lisitelement.badge": "Restricted",
// "access-status.unknown.lisitelement.badge": "Unknown",
"access-status.unknown.lisitelement.badge": "Unknown",
```

7. Fotografik gamoń

WYKONANIE:

- cd zadanie7
- mkdir obrazy
- cd obrazy
- mkdir wypakowane
- cd wypakowane
- find . -name "*.zip" -exec unzip -d /obrazy/wypakowane { } \;
- rm *.zip
- for f in *.png; do convert "\$f" "\${f%}.jpg"; done
- rm *.png
- for f in *.jpg; do magick "\$f" -resize x720 -density 96 -units PixelsPerInch "\$f"; done

OBJAŚNIENIA:

- mkdir – tworzy nowy katalog
- find . – przeszukuje bieżący katalog i jego podkatalogi
- -name "*.zip" – znajduje tylko pliki z rozszerzeniem .zip
- { } - zostanie zastąpione nazwą znalezionego pliku .zip
- rm – usuwanie plików
- for f in *.png; do ... done – pętla przechodząca przez wszystkie pliki PNG
- convert "\$f" "\${f%}.jpg" – użycie narzędzia ImageMagick do konwersji pliku z formatu PNG do JPG.
- \${f%}.jpg – zmienia rozszerzenie pliku z .png na .jpg.
- -resize x720 – zmienia rozmiar obrazu tak, aby wysokość wynosiła 720 pikseli
- -units PixelsPerInch – ustawia jednostkę gęstości na DPI
- -density 96 – ustawia rozdzielczość na 96 DPI w obu kierunkach

REZULTAT:

Zdjęcia z archiwów ZIP zostały rozpakowane, pliki PNG przekonwertowano na JPG, a wszystkie obrazy JPG przeskalowano do wysokości 720 pikseli przy rozdzielczości 96 DPI, gotowe do łatwego przesłania lub przechowywania. Usunięto niepotrzebne pliki

i przygotowano gotowy zestaw zdjęć w zoptymalizowanym formacie i rozdzielczości do dalszej wysyłki lub archiwizacji.

Ten komputer > Dokumenty > GitHub > programowanie > ZADANIE 7 > ZADANIE_7							
	Nazwa	Typ	Rozmiar po skomp...	Chronione...	Rozmiar	Stopień	Data modyfikacji
ep	adrien-olichon-3137064	Plik JPG	111 KB	Nie	113 KB	2%	15.05.2025 20:27
	aleksandr-slobodianyik-989941	Plik JPG	1 015 KB	Nie	1 026 KB	2%	15.05.2025 20:27
	aleksandr-slobodianyik-989946	Plik JPG	1 222 KB	Nie	1 232 KB	1%	15.05.2025 20:27
ty	alexander-dummer-37646-1356304	Plik JPG	70 KB	Nie	71 KB	1%	15.05.2025 20:27
	alexazabache-5117913	Plik JPG	39 KB	Nie	40 KB	2%	15.05.2025 20:27
	alex-montes-892479-1820563	Plik JPG	42 KB	Nie	43 KB	2%	15.05.2025 20:27

8. Wszędzie te PDF-y

WYKONANIE:

- `cd /c/zdjecia`
- `for f in *.jpg; do`
`magick "$f" -gravity south -background white -splice 0x40 -fill black -pointsize 20 -`
`annotate +0+5 "$f" "$f"`
`done`
- `montage *.jpg -tile 2x4 -geometry +10+10 portfolio.pdf`
- `mkdir Portfolio`
- `mv portfolio.pdf Portfolio/`

OBJAŚNIENIA:

- `cd /c/zdjecia` - przejście do katalogu ze zdjęciami
- `for f in *.jpg; do ... done` — pętla wykonująca polecenia dla każdego pliku .jpg w folderze
- `magick "$f"` — wywołanie programu ImageMagick na pliku \$f
- `-gravity south` — ustawia punkt odniesienia (do tekstu) na dół obrazu (pod zdjęciem)
- `-background white` — tło pod tekstem będzie białe
- `-splice 0x40` — dokładnie 40 pikseli wysokości z tłem na dole obrazu, żeby zmieścić podpis
- `-fill black` — kolor tekstu czarny
- `-pointsize 20` — rozmiar czcionki 20 punktów

- -annotate +0+5 "\$f" — dodaje napis (nazwę pliku) lekko przesunięty w pionie (+5 pikseli)

REZULTAT:

Zdjęcia zostały podpisane nazwami plików i ułożone w czytelny, estetyczny układ 2 kolumn na 4 wiersze na stronę formatu A4. Wynikiem jest gotowy do wydruku plik PDF („portfolio.pdf”) zawierający osiem zdjęć na stronie wraz z podpisami, co ułatwia prezentację i organizację prac fotograficznych. Plik PDF został uporządkowany w osobnym folderze „Portfolio” dla wygodnego dostępu.

9. Porządki w kopiach zapasowych

WYKONANIE:

- Cd/d/projekt
- Ls *.zip
- Unzip kopie-1.zip
- Unzip kopie-2.zip
- Mkdir -p
- For file *.zip; do rok=\${file:0:4}; mies=\${file:5:2}; mkdir -p kopie/\$rok/\$mies; mv “\$file” kopie/\$rok/ \$mies/; done (otrzymujemy gotowy wynik po wpisaniu tej komendy)

OBJAŚNIENIA:

- Cd/d/projekt - przechodzi z katalogu w folderze D w folder projektu
- Ls *.zip - sprawdza czy pliki znajdują się w folderze
- Unzip kopie-1.zip - rozpakowuje plik z kopie-1.zip
- Unzip kopie-2.zip - rozpakowuje plik z kopie-2.zip
- Mkdir -p kopie – tworzy nowy folder o nazwie kopie
- For file in *.zip - bierze każdy plik, który jest zipem
- Do rok=\${file:0:4} - wycina rok z nazwy np. 2010

- `Mies=${file:5:2}` - wycina miesiąc mp. 05
- `Mkdir -p kopie/$rok/$mies` – tworzy odpowiednie foldery, które znajdują się w folderze kopie
- `Mv "$file" kopie/$rok/$mies` – przenosi plik do odpowiedniego folderu

REZULTAT:

Zadanie polegało na uporządkowaniu plików kopii zapasowych w strukturę katalogów, w której każdy rok znajduje się w osobnym folderze, a w jego obrębie umieszczone są podkatalogi odpowiadające poszczególnym miesiącom. Do realizacji wykorzystano jednolinijkowe polecenie bash, co pozwoliło na uproszczenie całego procesu i wyeliminowanie potrzeby tworzenia oddzielnego skryptu. Efektem jest przejrzysta, logiczna struktura katalogów, ułatwiająca dalsze zarządzanie i archiwizację danych.

```
dom@DESKTOP-60868JA MSYS ~/zadanie9
$ cat data_sort.sh
#!/bin/bash

for file in *.zip; do
    rok=${file:0:4};
    mies=${file:5:2};
    mkdir -p Posortowane/$rok/$mies;
    mv "$file" Posortowane/$rok/$mies/;
done
```

10. Galeria dla grafika

WYKONANIE:

- `unzip galeria.zip -d galeria`
- `cd /c/zdjecia`
- `touch obrazy_html.sh`
- `nano obrazy_html.sh`
- (w edytorze)

```
#!/bin/bash

echo '<div class="responsive">' >> 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

```

- Zapisać i wyjść z nano
Ctrl + O
Ctrl + X
- Chmod +X obrazy_html.sh
- ./obrazy_html.sh

OBJAŚNIENIA:

- unzip galeria.zip -d galeria – rozpakowywuje plik do katalogu
- cd /c/zdjecia - zmienia bieżący katalog na folder
- touch obrazy_html.sh - tworzy pusty plik
- Nano obrazy_html.sh - uruchamia edytor tekstowy nano i otwiera plik obrazy_html.sh do edycji
- #!/bin/bash - skrypt uruchamiany przez bash
- echo '<div class="responsive">' >> galeria.html - zapis do pliku pierwszej linii, otwierającej div o klasie “responsive”
- for file in *.jpg; do - pętla przechodząca po wszystkich plikach
- echo " " >> galeria.html -dodaje link otwierający obrazek w nowej karcie
- echo " " >> galeria.html - umieszcza obraz w elemencie , który ładuje dany plik .jpg
- echo " " >> galeria.html - zamyka znacznik <a>
- echo " <div class=\"desc\">\$file</div>" >> galeria.html - dodaje podpis pod zdjęciem z nazwą pliku
- echo "</div>" >> galeria.html - zamyka sekcję <div class="gallery"> dla jednego zdjęcia

- Ctrl + 0 – zapisuje zmiany w pliku
- Ctrl + X – wychodzi z edytora
- chmod +X obrazy_html.sh - nadaje uprawnienia do wykonywania (uruchamia jako program)
- ./obrazy_html.sh - uruchamia skrypt

REZULTAT:

W ramach zadania został utworzony skrypt Bash (`obrazy_html.sh`), który automatycznie generuje prostą galerię zdjęć w formacie HTML na podstawie wszystkich plików `.jpg` znajdujących się w wybranym folderze. Powstały plik zawiera każdą fotografię w osobnym bloku, wyświetla miniatury obrazów `.jpg`, umożliwia otwarcie zdjęcia w nowej karcie (`target="_blank"`) i dodaje **nazwę pliku** jako podpis pod każdym zdjęciem.

```
dom@DESKTOP-60868JA MSYS ~/zadanie7/obrazy/wypakowane
$ cat obrazy_html.sh
#!/bin/bash

echo '<div class="responsive">' > 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
```