POLITECHNIKA ŚWIĘTOKRZYSKA

WYDZIAŁ ZARZĄDZANIA I MODELOWANIA KOMPUTEROWEGO

Kierunek – Inżynieria Danych

Studia pierwszego stopnia

Projekt z Podstaw Programowania

Łukasz Siudak (numer albumu: 96949) i Daniel Maj (numer albumu: 96943)

Opiekun pracy: Dr inż. Sławomir Piotr Koczubiej

Kielce 2025

Zadanie 3-Niesforne dane

Opis wykonania:

Wchodzimy do interesującego nas folderu(cd zadanie_3), a następnie wypakowujemy go za pomocą unzip dane.zip.

Następnie wykonujemy poniższe polecenia:

1**.** Rozpakowywujemy najpierw plik:**unzip dane.zip.**

2. Dzielimy plik na trzy kolumny: **paste - - - < dane.txt > dane2.txt.**

3. Dodajemy nagłówki kolumn do pliku wynikowego**:(echo -e "x y z" <dane2.txt>>dane3.txt) .**

4. Ostatnim krokiem jest podzielenie całości tak aby wszystko dobrze się prezentowało**:(echo -e "x\ty\tz"; paste - - - < dane.txt)>>dane4.txt.**

Wyjaśnienia funkcji:

A. **unzip dane.zip:** rozpkowywuje plik.
B. **paste - - - < dane.txt > dane2.txt**: dzieli plik na 3 kolumny
C. **(echo -e "x y z" <dane2.txt>>dane3.txt)** : dodaje nagłówki do kolumn
D. **echo -e "x\ty\tz"; paste - - - < dane.txt)>>dane4.txt:** odpowiednio dopasowywuje nagłówki do pozycji kolumn

Wynik:W pliku dane4.txt znajdują się dane z pliku dane.txt zorganizowane w trzech kolumnach oraz z dodanym nagłówkiem x y z.

Zadanie 4-Dodanie Poprawek

Opis wykonania:

Podobnie jak w poprzednim zadaniu wchodzimy do folderu który zawiera interesujące nas pliki(cd zadanie_4) po czym również wypakowujemy plik komendą unzip lista.zip.

Kolejno wykonujemy następujące polecenia:

1. Zmieniamy format obydów plików za pomocą komendy dos2unix (**dos2unix list.txt i dos2unix list-pop**).

2. Następnie używamy polecenia **diff -u lista.txt lista-pop.txt > poprawione.patch**

3. W dalszej kolejności korzystamy z polecenia **patch lista.txt poprawione.patch**

4. Odtąd lista.txt została uaktualniona o nowe dane z listy-pop.txt.

5. Za pomocą komendy md5sum**(md5sum lista.txt i md5sum lista-pop.txt)** sprawdzamy czy sumy kontrolne są takie same. Jeżeli tak to zadanie zostało wykonane prawidłowo.

Wyjaśnienia funkcji:

A. Obydwa pliki najpierw konwertujemy poleceniem dos2unix do wspólnego formatu.
B. Polecenie **diff -u lista.txt lista-pop.txt > poprawione.patch** porównuje zawartość obu plików i zapisuje wykryte różnice do nowego pliku **poprawione.patch** w formacie .patch.
C. Ten proces służy do zaktualizowania pliku lista.txt przy użyciu zmian zapisanych w pliku **poprawione.patch**, które pochodzą z pliku lista-pop.txt. Po zakończeniu operacji sprawdzane są sumy kontrolne MD5 dla obu plików (lista.txt i lista-pop.txt). Jeśli sumy kontrolne są takie same, oznacza to ,że całe zadanie zostało wykonane we właściwy sposób.

Wynik: Plik lista.txt został pomyślnie zaktualizowany danymi z lista-pop.txt, z dodaniem 7 nowych rekordów. Identyczne sumy kontrolne MD5 obu plików potwierdzają poprawność operacji.

Zadanie 5-Z CSV do SQL i z powrotem

Opis wykonania:

Podstawowym krokiem będzie wejście do folderu(cd zadanie_5) i rozpakowanie danych unzip csv.zip.

Dalej wykonujemy poniższe instrukcje:

Z CSV do SQL:

1.Najpierw używamy komendy:**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.sql.**

Z SQL do CSV:

1.Najpierw konwertujemy plik używając **echo "dateTime;steps;synced" > steps.csv.**

2.Następnym krokiem będzie polecenie **grep "INSERT INTO" steps-2csv.sql | sed -E 's/.*\(([^)]+)\);/\1/' | awk -F"," '{ts=substr($1,1,length($1)-3); printf "%s;%s;%s\n", ts, $2, $3}' >> two_steps.csv**

Wyjaśnienia funkcji:

Konwersja z CSV do SQL:

A. Polecenie **tail -n +2 steps-2sql.csv** pomija pierwszy wiersz pliku (czyli nagłówki).
B. Użycie **awk -F";"** ustawia średnik (;) jako separator pól.
C. Za pomocą printf **"INSERT INTO ..."** tworzona jest instrukcja SQL INSERT INTO na podstawie danych z każdej linii.
D. Wynik działania jest zapisywany do pliku **steps-2sql.sql** za pomocą operatora >.

Konwersja z SQL do CSV:

A. Polecenie **grep "INSERT INTO"** wybiera tylko te linie, które zawierają zapytania INSERT INTO.
B. Kolejne polecenie **sed -E 's/.*\(([^)]+)\);/\1/'** wyciąga dane znajdujące się w nawiasach.
C. Polecenie **awk -F","** rozdziela wartości oddzielone przecinkami.
D. Funkcja **substr($1,1,length($1)-3)** usuwa trzy końcowe znaki z pierwszego pola.
E. Instrukcja **printf "%s;%s;%s\n", ts, $2, $3** formatuje dane w postaci CSV jako: data;steps;synced.
F. Końcowo **>> two_steps.csv** dopisuje dane do pliku CSV.

Wynik:Plik steps-2sql.sql został utworzony na podstawie danych zawartych w pliku steps-2sql.csv. Pierwszy wiersz (z nagłówkami) został pominięty, a pozostałe dane zostały przekształcone w zapytania SQL typu INSERT INTO. Następnie ten plik SQL został odtworzony z powrotem do formatu CSV i zapisany jako steps.csv – zawierający prawidłowy nagłówek oraz przetworzone rekordy. Liczba przetworzonych rekordów to 25, a kontrola zgodności danych została potwierdzona poprzez dopasowanie sum kontrolnych MD5 plików wejściowych i wyjściowych.

Zadanie 6-Marudny tłumacz

Opis wykonanie:

1.Najpierw należy rozpakować oba pliki .json do jednego katalogu i przejść do tego folderu.
2.Następnie aby zdublowa linie i zakomentować pierwotną wersję, wykorzystujemy polecenie:
sed -E 's/^([[:space:]]*)"([^"]+)\.([^".]+)":[[:space:]]*"([^"]+)",*/\/\/ "\2.\3":
"\4",\n\1"\2.\u\3": "\4",/' en-7.2.json5 > pl-7.2.json5
3.W celu uzyskania jedynie nowych kluczy z pliku en-7.4.json5, trzeba najpierw wydobyć wszystkie klucze z obu plików .json5 i porównać je, aby zidentyfikować te, które występują tylko w nowszym pliku. Robimy to po kolei:
grep -o '"[^"]*":' en-7.2.json5 | tr -d '"' | tr -d ':' | sort > tekst-7.2.txt
oraz **grep -o '"[^"]*":' en-7.4.json5 | tr -d '"' | tr -d ':' | sort > tekst-7.4.txt**

4.Następnie porównujemy obydwa pliki tekstowe zawierające klucze, używając comm -13, co ukrywa linie występujące tylko w pierwszym pliku oraz wspólne dla obu – zostają tylko nowe klucze: **comm -13 tekst-7.2.txt tekst-7.4.txt > nowy-tekst.txt**

5.W kolejnym kroku tworzymy nowy plik .json5, zawierający tylko nowe wpisy z pliku en-7.4.json5, pasujące do kluczy z nowy-tekst.txt:
grep -Ff nowy-tekst.txt en-7.4.json5 > pl-7.4-new.json5

6.Aby uzyskać pełnoprawny plik .json5, dopisujemy nawiasy klamrowe:

echo "{" > pl-7.4.json5

cat pl-7.4-new.json5 >> pl-7.4.json5

echo "}" >> pl-7.4.json5

7. Na końcu możemy usunąć zbędne pliki za pomocą komendy:**rm tekst-*.txt nowy-tekst.txt pl-7.4-new.json5**

Wyjaśnienia funkcji:

sed -E 's/^([[:space:]]*)"([^"]+)\.([^".]+)":[[:space:]]*"([^"]+)",*/.../'

A. **sed -E** – uruchamia sed z rozszerzonymi wyrażeniami regularnymi,

B. **^([[:space:]]*)** – przechwytuje wiodące spacje (czyli wcięcia),

C. **"([^"]+)\.([^".]+)"** – rozdziela klucz na dwie części na podstawie kropki,

D. **"([^"]+)"** – wartość tekstowa (ta sama powtarzana dla zdublowanego wpisu),

E. **:[[:space:]]*** – uwzględnia dwukropek i ewentualne spacje,

F. **,*** – pozwala przechwycić końcowy przecinek.

grep -o '"[^"]*":' en-7.2.json5 | tr -d '"' | tr -d ':' | sort > tekst-7.2.txt

A. **grep -o** – zwraca tylko dopasowane fragmenty ("klucz":),

B. **tr -d '"'** – usuwa cudzysłowy,

C. **tr -d ':'** – usuwa dwukropki,

D. **sort** – sortuje alfabetycznie klucze,

E. > – zapisuje wynik do tekst-7.2.txt.
Analogiczne polecenie stosujemy do drugiego pliku (en-7.4.json5).

comm -13 tekst-7.2.txt tekst-7.4.txt > nowy-tekst.txt:

A. comm porównuje zawartość dwóch plików,

B. -1 ukrywa linie występujące tylko w pierwszym pliku,

C. -3 ukrywa linie wspólne,

D. Zostają tylko te z drugiego pliku, które zapisujemy do nowy-tekst.txt.

grep -Ff new-keys.txt en-7.4.json5 > pl-7.4-new.json5

A. -Ff – szuka dosłownych wzorców z nowy-tekst.txt w en-7.4.json5,

B. wynik zapisuje do pl-7.4-new.json5.

Wynik:W efekcie otrzymujemy dwa przetworzone pliki: pl-7.2.json5 i pl-7.4.json5, które po sprawdzeniu np. poleceniami cat, head lub tail, mają prawidłową strukturę i zawartość

Zadanie 7-Fotografik gamoń

Opis wykonania:

1.Pierwszym krokiem będzie wejście do folderu (**cd zadanie_7**)

2.Utworzenie nowego folderu za pomocą komendy **mkdir (mkdir Wypakowane)**

3.Następnie wypakować **find . -name "*.zip" -exec sh -c 'unzip "$1" -d Wypakowane' _ {} \;**

4.Kolejnym krokiem jest znalezienie plików png - **ls *.png.**

5.Konwersja z png na jpg za pomocą komendy: **for f in *.png; do magick "$f" "${f%.png}.jpg"; done**

6.Usunięcie plików png poprzez polecenie **rm *.png**

7.Zmiana wymiarów zdjęć na wymagane w treści zadania: **for f in *.jpg; do magick "$f" -resize x720 -density 96 -units PixelsPerInch "$f"; done**

8. Utworzenie katalogu gotowe(**mkdir Gotowe**) i przeniesienie tam plików typu JPG **(mv *.jpg Gotowe/**)

9.Ostatnim krokiem jest utworzenie katalogu zip: **zip -r ../Gotowe.zip Gotowe/**

Wyjaśnienia funkcji:

A. Na początku trzeba wejść do folderu z zapakowanymi zdjęciami za pomocą komendy cd. Następnie sprawdzić czy są tam intersujące nas pliki za pomocą **ls*.zip**
B. Kolejno utworzyć folder: **mkdir Wypakowane**
C. Kolejny krok to wypakowanie wszystkich plików za pomocą komendy:**find . -name "*.zip" -exec sh -c 'unzip "$1" -d Wypakowane' _ {} \;**
D. Potem sprawdzamy czy w folderze są pliki typu jpg za pomocą **ls *.png**
E. Dla znalezionych plików używamy komendy która przekonwertuje je na właściwy format:**for f in *.png; do magick "$f" "${f%.png}.jpg"; done**
F. Poprzednia funkcja utworzyła obrazy , które były w formacie png na obrazy w formacie jpg. Skoro już ich nie potrzebujemy możemy użyć funkcji rm*.png która usunie zbędne pliki.
G. Kiedy mamy już obrazy tylko i wyłącznie w formacie który nas interesuje to wtedy możemy zmienić ich wymiary na wymagane**:for f in *.jpg; do magick "$f" -resize x720 -density 96 -units PixelsPerInch "$f"; done**
H. Tworzymy nowy katalog(**mkdir Gotowe**) przenosimy tam pliki jpg(mv*.jpg Gotowe/), a na końcu tworzymy katalog zip do którego wszystko “pakujemy” komendą: **zip -r ../Gotowe.zip Gotowe/**

Wynik: Po wykonaniu wszystkich powyższych kroków powstanie nowy katalog zip zawierający pliki tylko w formacie jpg o interesujących nas wymiarach.

Zadanie 8-Wszędzie te PDF-y

Opis wykonania:

1. Najpierw wejść do folderu w którym mamy przekonwertowane zdjęcia z png na jpg

2. W tym zadaniu lepiej znaleźć się w wersji MSYS'a MINGW.

3. Następnie dodajemy do zdjęć wymagane przypisy za pomocą komendy:**for f in *.jpg; do magick "$f" -gravity south -background white -splice 0x40 -fill black -pointsize 20 -annotate +0+5 "$f" "$f">done**

4. Używając komendy montage możemy stworzyć pożądany plik PDF:**montage *.jpg -tile 2x4 -geometry +10+10 portfolio.pdf**

5. W folderze ze zdjęciami bo wpisaniu wszystkich komend będzie się znajdował plik o nazwie portfolio, ale zalecane jest przeniesienie tego pliku do innego folderu(o tej samej nazwie mkdir portfolio) tak aby nie trzeba było wpisywać jego nazwy w folderze ze zdjęciami.

Wyjaśnienia funkcji:

Polecenie **for f in *.jpg; do magick "$f" -gravity south -background white -splice 0x40 -fill black -pointsize 20 -annotate +0+5 "$f" "$f"** to pętla, która wykonuje się dla każdego pliku z rozszerzeniem .jpg:

A. **-gravity south** – ustawia punkt odniesienia na dolną część obrazu

B. **-background white** – ustala białe tło dla nowej przestrzeni dodawanej do obrazu

C. **-splice 0x40** – dodaje pasek o wysokości 40 pikseli u dołu obrazu – miejsce na podpis,

D. **-fill black** – kolor tekstu zostaje ustawiony na czarny,

E. **-pointsize 20** – rozmiar użytej czcionki wynosi 20 punktów,

F. **-annotate +0+5 "$f"** – dodaje podpis (nazwę pliku) przesunięty o 5 pikseli w górę od dolnej krawędzi

G. **"$f"** – zapisuje zmodyfikowany obraz, nadpisując plik źródłowy.

Wynik:W efekcie otrzymujemy plik portfolio.pdf, w którym na każdej stronie znajduje się 8 obrazów. Każdy z nich zawiera podpis będący nazwą odpowiedniego pliku graficznego. Tekst został umieszczony w dolnej części każdego obrazu, dzięki czemu końcowy efekt jest zgodny z założeniem

Zadanie 9-Porządki w kopiach zapasowych

Opis wykonania:

1.Najpierw wchodzimy do folderu w którym znajdują się interesujące nas pliki (kopie-1 i kopie-2) komendą (**cd Zadanie_9)**

2.Następnie rozpakowywujemy obydwa pliki komendą unzip (**unzip kopie-1** i **unzip kopie-2**)

3.Kolejnym krokiem jest utworzenie pliku kodu komendą touch: touch dane_sort.sh

4.Korzystamy z edytora nano piszemy treść kodu:

```
#!/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
```

5.Zatwierdzamy wszystkie zmiany CRTL+O, następnie klikamy ENTER i wychodzimy z edytora nano skrótem CRTL+X.
6.Nadajemy uprawnienia skryptowi korzystając z komendy:**chmod u+x dane_sort.sh**, a następnie uruchamiamy kod **./dane_sort.sh.**

Wyjaśnienia funkcji:

A. **mkdir -p kopie** – tworzy katalog o nazwie *Posortowane*, nawet jeśli nie istnieje,
B. **for file in *.zip** – pętla przetwarzająca każdy plik .zip w bieżącym katalogu,
C. **rok=${file:0:4}** – pobiera z nazwy pliku pierwsze cztery znaki, interpretując je jako rok
D. **mies=${file:5:2}** – wycina kolejne dwa znaki od piątej pozycji, czyli miesiąc
E. **mkdir -p Posortowane/$rok/$mies** – tworzy strukturę katalogów wg wzoru *rok/miesiąc* wewnątrz folderu *Posortowane*,
F. **mv "$file" Posortowane/$rok/$mies/** – przenosi plik do właściwego katalogu zgodnego z datą odczytaną z jego nazwy

Wynik:Celem zadania było zorganizowanie plików kopii zapasowych w czytelną strukturę katalogów, w której każdy rok posiada własny folder, a w jego wnętrzu znajdują się podkatalogi przypisane do konkretnych miesięcy.Rezultatem jest logicznie uporządkowany układ folderów, który znacznie ułatwia późniejsze przeszukiwanie i archiwizowanie plików.

Zadanie 10-Galeria dla grafika

Opis wykonania:

1**.**Przechodzimy do katalogu zawierającego obrazy, które wcześniej przekonwertowaliśmy na jpg
2.Tworzymy nowy plik skryptu o nazwie **zdjecia_html.sh** za pomocą polecenia **touch zdjecia_html.sh.**
3.Następnie otwieramy ten plik w edytorze nano, wpisując: **nano zdjecia_html.sh.**
4.W edytorze wprowadzamy następujący kod:

```
#!/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
```

5.Po zakończeniu edycji zapisujemy zmiany klawiszami CTRL + O, zatwierdzamy ENTER, a następnie wychodzimy z edytora, naciskając CTRL + X.
6.Aby skrypt mógł zostać uruchomiony, należy nadać mu prawa wykonania komendą **chmod u+x zdjecia_html.sh**.
7.Po nadaniu uprawnień uruchamiamy skrypt za pomocą **./zdjecia_html.sh**. Po jego wykonaniu, w bieżącym folderze pojawi się plik galeria.html zawierająca wszystkie obrazy

Wynik: Końcowym efektem będzie plik galeria,który będzie zawierał wszystkie obrazy.