Sprawozdanie – Zadania 3–10 (MSYS2)

Wprowadzenie

Celem zestawu zadań było zastosowanie narzędzi systemu MSYS2 (środowisko bash i zestaw narzędzi Unixowych) do przetwarzania danych tekstowych i graficznych.
Wykorzystano m.in. `cat`, `grep`, `awk`, `sed`, `diff`, `patch`, `md5sum`, `magick`, `tr`, `dos2unix`, `unix2dos`.

Zadanie 3 - Niesforne dane (cat + paste + echo)

Cel

- Podzielić plik na trzy kolumny, tak aby był gotowy do importu do pliku programu Excel
- Dodać nagłowki ("X","Y","Z")

Wykonanie

```
cat dane.txt
paste - - - < dane.txt
echo "x y z" >> dane.txt
```

Uwagi

- `cat` wyświetla podany plik dane.txt
- `paste` dzieli tekst na trzy kolumny
- `echo 'x y z"` dodaje nagłowki do kolumn

Zadanie 4 – Dodawanie poprawek (diff + patch)

Cel

- Porównać dwa pliki tekstowe `lista.txt` i `lista-pop.txt`.
- Wygenerować różnice jako plik `.patch`.
- Zaaplikować zmiany na pliku `lista.txt`, aby uzyskać `lista-pop.txt`.

Wykonanie

```
diff -u lista.txt lista-pop.txt > lista.patch
patch lista.txt lista.patch
```

Uwagi

- `diff -u` tworzy różnice w formacie unifikowanym, wymaganym przez `patch`.
- Jeśli pliki są puste, może to oznaczać błędne kodowanie – można użyć `dos2unix`:

```
dos2unix lista.txt lista-pop.txt
```

Zadanie 5 – Z CSV do SQL i z powrotem

Cel

- Wyświetlić unikalne nazwiska z pliku `dane.csv`.
- Posortować je alfabetycznie.
- Przygotować dane do importu do bazy danych SQL w poniższej postaci. INSERT INTO stepsData (time, intensity, steps) VALUES (1562001120, 19, 0); INSERT INTO stepsData (time, intensity, steps) VALUES (1562001180, 23, 0);
- Opracować w jaki sposób zamienić plik SQL na plik typu CSV

Rozwiązanie z użyciem `awk`, `sort`, `uniq`, `tail`

```
awk -F ';' '{ print $2 }' dane.csv | sort | uniq
dos2unix steps-2sql.csv
tail -n +2 steps-2sql.csv | awk -F';' '{print "INSERT INTo stepsData (time, intensity, steps) VALUES (" $1 ", " $2 ", " $3 ");"}' > s
cat steps-2sql
cat steps-2sql.csv
```

Objaśnienie

- `-F ';'` ustawia separator pól na średnik.
- `$2` to drugie pole – nazwisko.
- `sort | uniq` usuwa duplikaty i porządkuje wyniki alfabetycznie.

Zadanie 6 – Marudny tłumacz (md5sum)

Cel

- Sprawdzić, czy pliki `plik1.txt`, `plik2.txt`, itd. nie zostały zmodyfikowane.
- Dodać komentarz
- Porównać ich sumy MD5 z oryginalnymi.

Wykonanie

Całość kodu wykonanego zadania:

```
  md5sum lista.txt
  md5sum lista-pop.txt
  diff lista.txt lista-pop.txt
  cat lista.patch
  unzip tlumacz.zip
  cat en-7.2.json5
  cat en-7.4.json5
  cat en-7.2.json5 | awk '{print "// " $0; print $0}' > pl-7.2.json5
  cat en-7.2.json5
  cat pl-7.2.json5
  grep ':'en-7.2.json5 > old.txt
  grep ':'en-7.4.json5 > new.txt
  dos2unix en-7.2.json5
  dos2unix en-7.4.json5
  grep ':'en-7.2.json5 > old.txt
  grep ':'en-7.2.json5 > old.txt
  grep -E '^\s*".*":' en-7.2.json5 > old.txt
  grep -E '^\s*".*":' en-7.4.json5 > new.txt
  grep -Fvx -f old.txt new.txt > only_new.txt
  awk '{print "// " $0; print $0}' only_new.txt > pl-7.4.json5
  cat pl-7.4.json5
```

1. Obliczenie sum kontrolnych:

```
md5sum plik1.txt plik2.txt plik3.txt > suma.md5
```

2. Sprawdzenie później:

```
md5sum -c suma.md5
```

Wynik

- `OK` – plik niezmieniony,
- `FAILED` – zawartość pliku się zmieniła.

Zadanie 7 – Fotografik gamoń (konwersja obrazów)

Cel

- Rozpakować dwa archiwa ze zdjęciami (`kopie-1.zip`, `kopie-2.zip`).
- Skonwertować obrazy do `.jpg`.
- Zmniejszyć ich wysokość do 720 px.
- Ustawić rozdzielczość 96x96 DPI.
- Spakować je do jednego archiwum ZIP.

Wykorzystane narzędzia

- `unzip`, `cp`, `magick`, `zip`, `bash`.

Skrypt:

```
unzip kopie-1.zip -d kopie1
unzip kopie-2.zip -d kopie2

mkdir all
cp kopie1/* all/
cp kopie2/* all/

mkdir -p kopie

for file in all/*; do
  if [ -f "$file" ]; then
    name=$(basename "$file")
    name_noext="${name%.*}"
    magick "$file" -resize x720 -density 96x96 "kopie/${name_noext}.jpg"
  fi
done
```

Napotkane problemy

- **"no decode delegate for this image"** – brak obsługi formatu (np. TIFF, HEIC):
  - Rozwiązanie: zainstalowanie pełnej wersji ImageMagick w `msys64`.
- **"[: missing ']'"** – błąd składni `if`:
  - Rozwiązanie: poprawne użycie `if [ -f "$file" ]; then`.

Zadanie 8 – Wszędzie te PDF-y

Cel

- Pliki które w zadaniu 7 modyfikowaliśmy, trzeba z nich stworzyć PDF
- PDF ma mieć 8 zdjęć na stronie (po 2 w 4 kolumnach)
- Dodatkowo dodać nazwy plików do dokumentu PDF

Wykorzystane narzędzia

-

Skrypt:

```

```

Napotkane problemy

-

Zadanie 9 - Porządki w kopiach zapasowych

Cel

- uporządkowanie plików kopii w taki sposób, żeby każdy rok miał swój katalog i w każdym roku kopie z jednego miesiąca znajdowały się w osobnych podkatalogach.

Wykorzystane narzędzia

- `unzip`, `bash`, `mkdir`, `mv`, `rm`.

Skrypt:

```
mkdir -p temp

unzip -q kopie-1.zip -d temp
unzip -q kopie-2.zip -d temp

cd temp

for f in *.zip; do
  rok=${f:0:4}; miesiac=${f:5:2};
  mkdir -p "../kopie2/$rok/$miesiac";
  mv "$f" "../kopie2/$rok/$miesiac/";
done

cd .. && rm -r temp
```

Uwagi

- Kod działa poprawnie tylko jeśli pliki ZIP mają nazwę w formacie YYYY-MM-DD.zip. Jeśli trafi się inny format (np. kopia_maj2020.zip), zostanie pominięty lub źle posortowany.
    - możliwe rozwiązanie: dodać warunek `[[ "$f" =~ ^[0-9]{4}-[0-9]{2}-[0-9]{2}\.zip$ ]]` w pętli for, aby sprawdzać format.

Zadanie 10 - Galeria dla grafika

Cel

- poprawić plik index.html tak, aby zdjecia w galerii byly wyświetlane prawidłowo

Wykorzystane narzędzia

- `echo`, `for`.

Skrypt:

```
for img in *.png; do

  echo '<div class="responsive">'
  echo '  <div class="gallery">'
  echo "    <a target=\"_blank\" href=\"$img\">"
  echo "      <img src=\"$img\">"
  echo '    </a>'
  echo "    <div class=\"desc\">$img</div>"
  echo '  </div>'
  echo '</div>'

done >> index.html
```