Sprawozdanie z projektu z podstaw programowania

Jakub Piskulak s096948, Szymon Ciborowski s096937

Wstęp

W repozytorium, w katalogach odpowiadających poszczególnym zadaniom, znajdują się zdjęcia przedstawiające rozwiązania.

Pracując w środowisku Windows, zadbaliśmy o czytelne uporządkowanie materiałów oraz poprawne formatowanie plików graficznych i tekstowych.

Rozwiązania zawarliśmy w pliku Projekt_z_Podstaw_Programowania_rozwiązania.zip

Poniżej prezentujemy rozwiązania zadań projektowych.

Wykorzystywane polecenia Unix/Linux:

1. cat – służy do wyświetlania zawartości plików tekstowych w terminalu.
2. head – pokazuje początkowe linie pliku (domyślnie pierwsze 10).
3. unzip – rozpakowuje archiwa w formacie ZIP.
4. zip – tworzy archiwa ZIP z podanych plików.
5. rm – usuwa pliki z systemu.
6. grep – wyszukuje linie zawierające określony wzorzec w plikach tekstowych.
7. sed – umożliwia przekształcanie tekstu, np. poprzez zamiany na podstawie wyrażeń regularnych.
8. awk – służy do analizy i przetwarzania danych tekstowych.
9. diff – porównuje zawartość dwóch plików lub katalogów, wskazując różnice.

Niesforne dane

Opis rozwiązania:

Na początku rozpakowywany jest plik dane.zip, po czym jego zawartość zostaje przekształcona. Za pomocą polecenia cat przekazywana jest zawartość pliku dane.txt do programu awk. W sekcji BEGIN skrypt wypisuje nagłówek "x y z", oddzielając wartości tabulatorami. Następnie przetwarzane są kolejne linie pliku — dane wypisywane są w trzech kolumnach. Separator wierszy ustawiany jest na tabulator, z wyjątkiem sytuacji, gdy numer przetwarzanego wiersza jest podzielny przez trzy — wtedy zamiast tego stosowany jest znak nowej linii.

Końcowy wynik zapisywany jest do pliku dane2.txt.

Dodawanie poprawek

Opis rozwiązania:

Na początku rozpakowujemy i konwertujemy dane potrzebne do wykonania zadania. Następnie za pomocą polecenia diff identyfikujemy różnice pomiędzy plikami lista.txt i lista-pop.txt, generując na tej podstawie plik łatki lista.patch. Plik ten zawiera instrukcje dotyczące zmian — czyli które linie należy dodać, usunąć lub zmodyfikować, aby przekształcić lista.txt w wersję zgodną z lista-pop.txt.
Następnie polecenie patch wykorzystuje wygenerowaną łatkę do wprowadzenia zmian w pliku lista.txt. Aby zweryfikować poprawność operacji, za pomocą md5sum porównywane są sumy kontrolne obu plików — ich zgodność potwierdza, że pliki są identyczne.

Z CSV do SQL i z powrotem

Opis rozwiązania:

Na początku rozpakowujemy i konwertujemy dane niezbędne do wykonania zadania. Następnie, za pomocą polecenia `cat`, zawartość pliku `steps-2sql.csv` przekazywana jest do programu awk. W sekcji `BEGIN` ustalany jest separator pól jako średnik (`;`), a separator wyjściowy ustawiany jest na pusty ciąg (czyli brak dodatkowych znaków między liniami). Dla każdego wiersza pliku awk generuje zapytanie SQL w postaci:
`INSERT INTO stepsData (time, intensity, steps) VALUES (<kolumna1>, <kolumna2>, <kolumna3>);`

W miejsce `<kolumna1>`, `<kolumna2>` i `<kolumna3>` podstawiane są wartości z odpowiednich kolumn pliku CSV. Wynik zapisywany jest do pliku `steps-2csv.sql`.

W kolejnym kroku, ponownie wykorzystując `cat`, zawartość pliku `steps-2csv.sql` przekazywana jest do narzędzia `sed`, które wykonuje szereg transformacji tekstu:

- `s/^.*VALUES //g` – usuwa wszystko od początku linii do frazy `VALUES` włącznie,
- `s/[();]//g` – usuwa wszystkie nawiasy okrągłe oraz średniki,
- `s/, /;/g` – zamienia przecinki i spacje na średniki,
- `s/000;/;/g` – zastępuje ciąg `000;` samym średnikiem.

Tak przetworzony wynik zapisywany jest do pliku `steps-2csv.csv`.

Marudny tłumacz

Opis rozwiązania**:**

Na wstępie rozpakowujemy i konwertujemy dane potrzebne do wykonania zadania. Za pomocą polecenia `cat` zawartość pliku `en-7.2.json5` przekazywana jest do programu `awk`. W sekcji `BEGIN` drukowany jest znak otwierający nawias klamrowy. Następnie każda linia wejściowa jest wypisywana dwukrotnie — pierwszy raz jako komentarz, a drugi raz bez zmian. W sekcji `END` dodawany jest znak zamykający nawias klamrowy . Wynik zapisywany jest do pliku `pl-7.2.json5`.

W kolejnym etapie, przy użyciu polecenia `diff -u`, porównywane są pliki `en-7.4.json5` i `en-7.2.json5`, a wynik różnic w formacie ujednoliconym zapisywany jest do pliku `en.patch`.

Następnie plik `en.patch` jest przetwarzany przy użyciu `awk`. W sekcji `BEGIN` drukowany jest nawias klamrowy otwierający oraz znak nowej linii. Dla każdej linii rozpoczynającej się od znaku minus, czyli oznaczającej treść występującą w `en-7.4.json5`, a nieobecną w `en-7.2.json5`, linia ta zostaje wypisana dwukrotnie — raz jako komentarz, a drugi raz bez symbolu `-`. W sekcji `END` dodawany jest nawias klamrowy zamykający. Efekt końcowy zapisywany jest do pliku `pl-7.4.json5`.

Fotografik gamoń

Opis rozwiązania:

Na początku rozpakowywane są wszystkie niezbędne dane, a zbędne archiwa zostają usunięte. Następnie, przy użyciu polecenia `magick mogrify`, wszystkie pliki graficzne w formacie `.png` są konwertowane do formatu `.jpg`. Po zakończeniu konwersji, pliki `.png` zostają usunięte jako niepotrzebne.

Kolejnym krokiem jest ujednolicenie parametrów wszystkich obrazów `.jpg`. W tym celu przy pomocy `magick mogrify`:

- ustawiana jest wysokość każdego obrazu na 720 pikseli, przy zachowaniu oryginalnych proporcji,
- określana jest rozdzielczość na poziomie 96 DPI,
- jako jednostkę rozdzielczości ustawiane są piksele na cal,
- a każdy przetworzony plik nadpisywany jest z zachowaniem pierwotnej nazwy.

Wszędzie te PDF-y

Opis rozwiązania:

Za pomocą polecenia `magick montage` tworzony jest kolaż z obrazów `.jpg` przygotowanych w poprzednim zadaniu. Obrazy rozmieszczane są na stronach w formacie A4 w układzie 2 kolumn na 4 wiersze. Każdy obraz jest opatrzony podpisem z nazwą pliku, a czcionka podpisu ma rozmiar 72 punktów. Ustawiany jest również margines o wartości 200 pikseli zarówno w pionie, jak i poziomie. Gotowy kolaż zapisywany jest do pliku w formacie PDF jako `./montage.pdf`.

Porządki w kopiach zapasowych

Opis rozwiązania:

Na początku rozpakowywane są dane potrzebne do wykonania zadania. Następnie, w pętli `for`, przetwarzane są wszystkie pliki z rozszerzeniem `.zip`. Dla każdego z nich na podstawie nazwy pliku wyodrębniane są:

- rok: `rok=${plik:0:4}` – pierwsze 4 znaki z nazwy pliku,
- miesiąc: `miesiac=${plik:5:2}` – dwa znaki od 6. pozycji.

Na tej podstawie tworzona jest odpowiednia struktura katalogów przy użyciu polecenia `mkdir -p "$rok/$miesiac"`. Następnie dany plik `.zip` kopiowany jest do utworzonego folderu za pomocą `cp "$plik" "$rok/$miesiac/"`.

Po zakończeniu pętli, wszystkie pliki `.zip` znajdujące się w katalogu głównym zostają usunięte poleceniem `rm *.zip`.

Galeria dla grafika

Opis rozwiązania:

Na początku kopiujemy potrzebne pliki z zadania *fotografik gamoń* do katalogu bieżącego zadania. Następnie tworzymy zmienną `html_output`, w której umieszczany jest szkielet dokumentu HTML — zawierający podstawowe formatowanie i style CSS dla galerii.

W dalszej części, w pętli `for`, dla każdego pliku `.jpg` do zmiennej `html_output` dynamicznie dodawany jest blok `<div>`, zawierający obrazek oraz podpis z nazwą pliku, zgodnie ze specyfikacją zawartą w treści zadania.

Po zakończeniu pętli, do zmiennej `html_output` dopisywane jest zamknięcie dokumentu HTML. Całość zostaje zapisana do pliku `galeria.html`.