✔ Wykonano

Otwarto: poniedziałek, 6 marca 2023, 00:00
Wymagane do: środa, 22 marca 2023, 23:59

W zadaniu 1 proszę skorzystać z obu wariantów implementacji:

- lib - przy użyciu funkcji biblioteki C: **fread()** i **fwrite()**
- sys - przy użyciu funkcji systemowych: **read()** i **write()**

Dla obu wariantów implementacji należy przeprowadzić pomiar czasu wykonywania. Wyniki należy przedstawić w formie pliku pomiar_zad_1.txt

Zadanie 1 (25%) Napisz program, który przyjmuje 4 argumenty wiersza poleceń:

1. znak ASCII, który należy znaleźć w pliku wejściowym
2. znak ASCII, na który należy zamienić wszystkie wystąpienia pierwszego argumentu
3. nazwa pliku wejściowego, w którym należy znaleźć pierwszy argument
4. nazwa pliku wyjściowego, do którego należy zapisać zawartość pliku wejściowego z zamienionym znakami argv[1] na argv[2].

Porównaj wyjście programu z wynikiem polecenia **tr:**

./zamien [znak1] [znak2] plik_wejściowy plik_wyjściowy ; tr [znak1] [znak2] < plik_wejściowy > tmp ; diff -s tmp plik_wyjściowy

W zadaniu 2 można wybrać do zaimplementowania tylko jeden wariant:

- albo **fopen(), fseek(), fread(), fwrite(), fclose()**
- albo **open(), lseek(), read(), write(), close()**

Wybrany wariant należy opracować na dwa sposoby:

1. Czytanie po 1 znaku.
2. Czytanie bloków po 1024 znaki (plik wynikowy powinien być identyczny jak w wariancie 1.)

Dla obu sposobów implementacji należy przeprowadzić pomiar czasu wykonywania. Wyniki należy przedstawić w formie pliku pomiar_zad_2.txt

Zadanie 2 (25%) Napisz program, który kopiuje zawartość jednego pliku do drugiego, odwróconą bajt po bajcie.

Wskazówki: Wywołania w rodzaju **fseek(infile, +1024, SEEK_END)** lub **lseek(in, +1024, SEEK_END)** są zupełnie legalne i nie powodują żadnych skutków ubocznych. Aby po przeczytaniu bloku znaków cofnąć się na początek poprzedniego bloku, należy jako drugi argument funkcji **fseek(..., ..., SEEK_CUR)** lub **lseek(..., ..., SEEK_CUR)** podać *podwojoną* długość bloku ze znakiem minus. Działanie programu należy zweryfikować następująco: 1) odwrócić krótki plik tekstowy, podejrzeć wynik, sprawdzić szczególnie początkowe i końcowe znaki. 2) **./reverse plik_binarny tmp1 ; ./reverse tmp1 tmp2 ; diff -s tmp2 plik_binarny** 3) można też porównać (**diff -s**) wynik działania programu i wynik polecenia **tac < plik_wejściowy | rev > plik_wyjściowy**

Zadanie 3 (25%) Napisz program, który będzie przeglądał bieżący katalog, korzystając z funkcji **opendir()**, **readdir()** i **stat()**. Dla każdego znalezionego pliku, który nie jest katalogiem, czyli **!S_ISDIR(bufor_stat.st_mode**), należy wypisać rozmiar i nazwę pliku. Ponadto na koniec należy wypisać sumaryczny rozmiar wszystkich plików. Nie należy przeglądać podkatalogów! Sumaryczny rozmiar plików należy przechowywać w zmiennej typu **long long** i wypisywać ją przez format **%lld**.

Działanie programu porównaj z działaniem polecenia **wc --bytes ***

Zadanie 4 (25%) Napisz program, który będzie przeglądał katalog podany jako argument wywołania i jego podkatalogi, korzystając z funkcji **ftw()** (uproszczonej wersji funkcji **nftw()**). Dla każdego znalezionego pliku, który nie jest katalogiem, czyli **!S_ISDIR(bufor_stat.st_mode**), należy wypisać rozmiar i nazwę pliku. Ponadto na koniec należy wypisać sumaryczny rozmiar wszystkich plików. Dobra wiadomość: funkcja **ftw()** przyjmuje ścieżki i bezwzględne, i względne.

Działanie programu porównaj z działaniem polecenia **find nazwa_katalogu | xargs wc --bytes**