Zadania - zestaw 2

Zapoznaj się ze składnią i operacjami wykonywanymi przez poniższe funkcje:

- funkcje operujące na plikach i katalogach: `open()`, `close()`, `read()`, `write()`, `fcntl()`, `stat()`, `fstat()`, `lstat()`, `mkdir()`, `rmdir()`, `opendir()`, `closedir()`, `readdir()`, `rewinddir()`, `nftw()`, `fopen()`, `fclose()`, `getc()`, `putc()`,
- funkcje i zmienne do obsługi błędów: *perror*, *errno*.

Zadanie 1. Porównanie wydajności systemowych i bibliotecznych funkcji We/Wy (55%)

- (30%) Celem zadania jest napisanie programu porównującego wydajność systemowych oraz bibliotecznych funkcji wejścia/wyjścia. Program operował będzie na przechowywanej w pliku tablicy napisów (rekordów). Dla uproszczenia pojedynczy napis będzie miał stałą wielkość. Nazwa pliku, wielkość oraz liczba i długość napisów stanowić będą argumenty wywołania programu.

  Program udostępniać powinien operacje:

  - `generate` - tworzenie pliku z rekordami wypełnionego wygenerowaną losową zawartością (można wykorzystać wirtualny generator `/dev/random`) lub w wersji uproszczonej funkcję `rand()`
  - `sort` - sortuje rekordy w pliku (w porządku leksykograficznym), używając sortowania szybkiego. Pivotem dla sortowania niech będzie wartość pierwszego napisu / rekordu. Podczas sortowania w pamięci powinny być przechowywane jednocześnie najwyżej dwa rekordy (porównywanie dwóch rekordów).
  - `copy` - kopiuje `plik1` do `pliku2`. Kopiowanie powinno odbywać się za pomocą bufora o zadanej wielkości rekordu.

  Sortowanie i kopiowanie powinno być zaimplementowane w dwóch wariantach:

  - `sys` - przy użyciu funkcji systemowych: `read` i `write`
  - `lib` - przy użyciu funkcji biblioteki C: `fread` i `fwrite`

Rodzaj operacji oraz sposób dostępu do plików ma być wybierany na podstawie argumentu wywołania, np.:
`./program generate dane 100 512` powinno losowo generować 100 rekordów o długości 512 bajtów (znaków)
do pliku `dane`,
`./program sort dane 100 512 sys` powinien sortować rekordy w pliku `dane` przy

użyciu funkcji systemowych,
zakładając że zawiera on 100 rekordów wielkości 512 bajtów
`./program copy plik1 plik2 100 512` lib powinno skopiować 100 rekordów pliku 1 do pliku 2 za pomocą funkcji
bibliotecznych z wykorzystaniem bufora 512 bajtów

- (25%) Dla obu wariantów implementacji przeprowadź pomiary czasu użytkownika i czasu systemowego operacji sortowania i kopiowania. Testy wykonaj dla następujących rozmiarów rekordu: 1, 4, 512, 1024, 4096 i 8192 bajty. Dla każdego rozmiaru rekordu wykonaj dwa testy różniące się liczbą rekordów w sortowanym pliku. Liczby rekordów dobierz tak, by czas sortowania mieścił się w przedziale od kilku do kilkudziesięciu sekund. Porównując dwa warianty implementacji, należy korzystać z identycznego pliku do sortowania (po wygenerowaniu, a przed sortowaniem, utwórz jego kopię). Zmierzone czasy zestaw w pliku `wyniki.txt`. Do pliku dodaj komentarz podsumowujący wnioski z testów.

Zadanie 2. Operacje na strukturze katalogów (45%)

Napisz prosty odpowiednik programu *find* — program powinien implementować następujące opcje: '-mtime', '-atime' oraz '-maxdepth'. W przypadku dwóch pierwszych, podobnie jak w przypadku *find*, argumentem może być: liczba (bezwzględna), liczba poprzedzonej znakiem '+' lub liczba poprzedzona znakiem '-'. Program ma wypisać na standardowe wyjście następujące informacje o znalezionych plikach:

- Ścieżka bezwzględna pliku,
- Liczbę dowiązań
- Rodzaj pliku (zwykły plik - file, katalog - dir, urządzenie znakowe - char dev, urządzenie blokowe - block dev, potok nazwany - fifo, link symboliczny - slink, soket - sock)
- Rozmiar w bajtach,
- Datę ostatniego dostępu,
- Datę ostatniej modyfikacji.

Ścieżka podana jako argument wywołania może być względna lub bezwzględna. Program nie powinien podążać za dowiązaniami symbolicznymi do katalogów.

Program należy zaimplementować w dwóch wariantach:

1. Korzystając z funkcji `opendir()`, `readdir()` oraz funkcji z rodziny `stat` (25%)
2. Korzystając z funkcji `nftw()` (20%)

W ramach testowania funkcji:

- Utwórz w badanej strukturze katalogów jakieś dowiązania symboliczne, zwykłe pliki i katalogi.

- Porównaj wynik szukania (własna implementacja) z wynikiem szukania za pomocą polecenia *find* — wywołaj polecenie *find* z opcjami '-mtime', '-atime' lub '-maxdepth' — przykłady:
- `find /etc -mtime 0  2> /dev/null | wc -l`
- `find /usr -atime -7 -maxdepth 2 2> /dev/null | wc -l`

  Następnie wywołaj swój program z takimi samymi opcjami i sprawdź, czy liczba znalezionych plików jest taka sama.