



Zadanie 2.4

2. System plików FAT12/16

4. Projekt: ścieżki bezwzględne, wielkości liter (4,0)

Treść Pliki Historia ? Pomoc

Termin

2020-12-20

zostało 10 dni

Trudność



Punkty: 5

Próba

30

Ocena maszyny



Czas testu: 20sek

Inspekcja kodu



Plagiat



Raporty

Raport główny

Czytnik systemów plików, wymagania na ocenę 4,0

Rozbuduj swój kod czytnika woluminów, sformatowanych w systemie FAT12/16, o następujące funkcjonalności:

1. Poprawną obsługę różnych wielkości liter podczas wyszukiwania wpisów w katalogu.
2. Obsługę ścieżek bezwzględnych do plików w funkcji `file_open()`.
3. Obsługę ścieżek bezwzględnych do katalogów w funkcji `dir_open()`.

Kod przygotowany w ramach zadania musi spełniać również wymogi na ocenę **3,0** (2.3 Projekt: czytnik dysku, woluminu, katalogu głównego i plików (3,0)).

Poprawna obsługa różnych wielkości liter podczas wyszukiwania wpisów w katalogu

Specyfikacja FAT nie uwzględnia wielkości liter. Jest to podejście odmienne to tego, stosowanego w systemach plików z rodziny Unix. Zatem nazwy `ALAİKOT.TXT`, `alaikot.txt` oraz `AlaIKOt.TxT` opisują dokładnie ten sam plik fizyczny. Podobnie sprawa ma się ze ścieżkami. Poniższe wskazują na ten sam plik:

```
/DOK/PRACA/REPOS/TEST.TXT
/dok/Praca/repOS/test.txt
```

Należy zmodyfikować funkcje `file_open()` oraz `dir_open()` tak, aby wyszukując pliki oraz katalogi po podanych (w parametrach) nazwach, ignorowały wielkość liter.

Pamiętaj, że specyfikacja FAT wymaga, aby nazwy plików w strukturach dyskowych FAT były **zawsze** zapisane wielkimi literami, w alfabecie łacińskim (A-Z). Oznacza to, że jeżeli użytkownik utworzy plik `test.txt`, to na fizycznym nośniku zapisana jest nazwa `TEST.TXT`. A właściwie to `TEST TXT` 🤔

Obsługa ścieżek bezwzględnych do plików w funkcji `file_open()` oraz katalogów w funkcji `dir_open()`

W rozszerzeniu na ocenę 4,0 należy uwzględnić hierarchiczną budowę struktury katalogów w woluminie FAT. Oznacza to konieczność modyfikacji funkcji `file_open()` oraz `dir_open()` tak, aby nazwy plików oraz katalogów, do nich przekazywanych, można było podawać pełnymi ścieżkami.

Ścieżka to ciąg nazw katalogów, rozdzielonych symbolem `\`, np. `aaa\bbb\ccc`. Wszystkie ścieżki są bezwzględne, zatem oba poniższe zapisy wskazują na ten sam obiekt:

```
\muzyka\IcedEart.h\dante.mp3
muzyka\IcedEart.h\dante.mp3
```

Zwróć uwagę, że w systemach POSIX-owych katalogi rozdzielone są znakiem `/` podczas gdy w systemie plików FAT, jako wywodzącym się z linii systemów operacyjnych DOS (MS-DOS, PC-DOS), katalogi rozdzielane są symbolem `"\"`.

Wprowadzana modyfikacja musi uwzględnić również standardowe symbole `..` oraz `.` oznaczające odpowiednio katalog nadrzędny oraz katalog bieżący. Z ich wykorzystaniem poniższe zestawy ścieżek wskazują na te same obiekty:

```
\archiwum\praca\moje\muzyka\...\..\dane.txt
\archiwum\dane.txt
```

```
zakupy.txt
\archiwum\..\zakupy.txt
\archiwum\..\archiwum\..\zakupy.txt
\zakupy.txt
```

```
\katalog\.\..\..\..\plik.txt
\katalog\plik.txt
```

```
utils\ndd\ndd.exe
\utils\.\ndd\.\ndd.exe
\.\utils\..\utils\.\ndd\ndd.exe
```

Wszystkie podrzędne katalogi zawierają wpisy `.` oraz `..`. Wyjątkiem jest katalog główny - on nie ma katalogów nadrzędnych. Nie są to specjalne operatory/symbole, a nazwy fizycznych wpisów w podkatalogach. Dlatego te ścieżki są niepoprawne -- występuje w nich próba wyjścia "ponad" katalog główny:

```
\archiwum\...\dane.txt
\archiwum\...\archiwum\dane.txt
```

Ponadto, jeżeli któryś z katalogów w ścieżce nie istnieje, to cała ścieżka też nie istnieje -- wskazuje na nieistniejący obiekt. Stwierdzenie to jest prawdziwe nawet w przypadku użycia nazwy nieistniejącego katalogu z następującym, zaraz po nim, "wyjściem" `...`. Zakładając przyrostek `!xxx` jako nieistniejący katalog `xxx`, poniższe ścieżki są niepoprawne:

```
\archiwum\!dane\test.txt
\archiwum\!dane\..\test.txt
\archiwum\!dane\...\test.txt
```

(Wykrzyknik jest tylko przykładem w tym opisie; FAT może posiadać wpisy zaczynające się tym znakiem.)

Co więcej, nawet jeżeli plik `\archiwum\test.txt` istnieje, to ścieżka `\archiwum\!dane\..\test.txt` nadal wskazuje na nieistniejący plik.

Przykładowe obrazy:

- FAT12: 1 2 3 4 5

- FAT16: [1](#) [2](#) [3](#) [4](#) [5](#)

Uwaga

- W zadaniu nie jest testowana funkcja `main()`. Można ją wykorzystać do swoich testów.
 - Wszystkie struktury oraz prototypy, wymagane specyfikacją zadania, należy umieścić w pliku nagłówkowym `file_reader.h`.
-