Systém souborů

Termín odevzdání:	**13.06.2021 23:59:59**
Hodnocení:	**0.0000**
Max. hodnocení:	**30.0000** (bez bonusů)
Odevzdaná řešení:	4 / 60
Nápovědy:	0 / 0

Cílem této práce je podrobněji se seznámit s tím, jak funguje systém souborů. Vaším úkolem je realizovat vrstvu, která bude implementovat jednoduchý systém souborů. Vaše implementace bude komunikovat s diskem (blokovým zařízením), které poskytuje funkce pro zápis a čtení sektorů. Vaše implementace naopak bude poskytovat rozhraní pro vytvoření/připojení/odpojení systému souborů a dále pak funkce pro otevření/čtení/zápis/zavření souborů.

Vlastní implementace nebude "opravdový" systém souborů běžící na úrovni kernelu. Pro zjednodušení celá implementace poběží v uživatelském prostoru, disk bude simulovaný voláním funkcí. Další zjednodušení spočívá v tom, že se nemusíte zabývat právy přístupu, vlastníky souborů, časovými značkami změn, adresáři, ... Nejdůležitějším úkolem je vymyslet, jak budou data v souborech ukládaná so sektorů a jak bude spravované volné místo.

Požadované rozhraní je realizováno třídou `CFileSystem` a podpůrnými strukturami `TBlkDev`, `TFile` a konstantami. Konstanty a struktury `TBlkDev` a `TFile` jsou deklarované v testovacím prostředí. Vaše implementace je bude pouze využívat, nesmí je měnit. Naopak, je potřeba implementovat třídu `CFileSystem` podle dodaného rozhraní.

Konstanty:

`FILENAME_LEN_MAX`
: udává maximální délku jména souboru. Pozor, pokud ukládáte jméno v ASCIIZ konvenci, může být potřeba až `FILENAME_LEN_MAX + 1` bajtů (+1 na ukončující nulu).

`DIR_ENTRIES_MAX`
: udává maximální počet souborů na disku.

`OPEN_FILES_MAX`
: určuje maximální počet najednou otevřených souborů.

`SECTOR_SIZE`
: udává velikost jednoho sektoru blokového zařízení. Blokové zařízení pracuje pouze s bloky této velikosti (a jejich násobky). Nelze pracovat s jednotlivými bajty.

`DEVICE_SIZE_MIN, DEVICE_SIZE_MAX`
: určuje přípustné rozmezí velikosti blokového zařízení, se kterým musí být Vaše implementace schopná pracovat.

`TFile`

Tato struktura slouží pro získávání informací o uložených souborech. Má složky `m_FileName` (jméno souboru) a `m_FileSize` (velikost souboru). Volání metod `CFileSystem::FindFirst` a `CFileSystem::FindNext` bude tyto struktury vyplňovat, jejich zpracováním pak volající získá informace o uložených souborech.

`TBlkDev`

Tato struktura slouží pro komunikaci s blokovým zařízením, kam filesystém bude ukládat data. Protože se jedná o blokové zařízení, umožňuje rozhraní práci pouze po celých sektorech. Rozhraní je:

`m_Sectors`
: udává počet sektorů, které jsou na blokovém zařízení k dispozici. Pro přístup k jednotlivým sektorům je potřeba zadat číslo sektoru v rozmezí `0` až `m_Sectors - 1`.

`m_Read (startSec, dst, cntSec)`
: tato složka odkazuje na funkci, která fyzicky provede čtení sektoru nebo více po sobě jdoucích sektorů. Parametrem je index prvního čteného sektoru (`startSec`), ukazatel na paměťový buffer pro zápis načtených dat (`dst`) a počet čtených sektorů (`cntSec`). Číst lze pouze sektory, které fyzicky existují (tedy `startSec + cntSec ≤ m_Sectors`). Volající je zodpovědný za připravení dostatečně dlouhého bufferu, do kterého se požadované sektory načtou. Návratovou hodnotou je počet skutečně přečtených sektorů, tedy `cntSec` pokud jsou správně načtené všechny požadované sektory.

`m_Write (startSec, src, cntSec)`
: tato složka odkazuje na funkci, která fyzicky provede zápis jednoho sektoru nebo více po sobě jdoucích sektorů. Rozhraní je stejné jako u volání `m_Read`.

použití

Vaše implementace dostane instanci `TBlkDev` při inicializaci. Je rozumné si předanou instanci zkopírovat do nějaké členské proměnné a následně jejím prostřednictvím provádět potřebné vstupně/výstupní operace.

CFileSystem

Třída implementuje vlastní systém souborů. Poskytuje rozhraní obvyklé pro práci se soubory (otevření, čtení, zápis), data vhodným způsobem serializuje do sektorů a ukládá na poskytnuté blokové zařízení. Implementace metod této třídy je Váš úkol.

`CreateFs(dev)`
: metoda slouží k vytvoření servisních záznamů na blokovém zařízení. Jedná se o obdobu příkazu `mkfs`, kterým se zařízení formátuje. Vaše implementace si na blokovém zařízení může poznamenávat nějaké servisní informace, např. seznam volných/alokovaných bloků. Toto volání by mělo servisní informace nastavit do výchozího stavu. Volání samozřejmě zničí původní obsah (pokud na disku nějaký byl). Není ale chytré celý disk vyplnit např. nulami - toto by zbytečně zdržovalo. Parametrem volání je struktura, kterou použijete pro komunikaci s blokovým zařízením, návratovou hodnotou je signalizace úspěchu (`true`)/neúspěchu (`false`). Samotná metoda `CreateFs` disk ještě nepřipojuje (`Mount` bude voláno později, explicitně). Předpokládá se, že režijní záznamy Vaší implementace spotřebují nějakou část kapacity blokového zařízení. Limit je nastaven na 10% (tj. 90% kapacity musí zůstat pro data souborů).

`Mount (dev)`
: volání připojí filesystém. Odpovídá UNIXovému příkazu `mount`. Po připojení filesystému mohou být volané metody pro práci se soubory (`OpenFile`, ...). Parametrem volání je struktura pro komunikaci s blokovým zařízením. Metoda vrací instanci `CFileSystem`, kterou následně půjde manipulovat s jednotlivými soubory (jde o factory).

`Umount`
: metoda odpojí dříve připojený filesystém. Odpovídá volání `umount` v UNIXu. Metoda uzavře všechny dosud otevřené soubory, uloží na blokové zařízení všechna dosud neuložená data a uvolní všechny Vámi alokované prostředky (paměť). Vrátí hodnotu `false` pro neúspěch, `true` pro úspěch.

`OpenFile`
: metoda otevře soubor zadaného jména. Soubor je otevřen buď pro zápis (`writeMode` je `true`) nebo pro čtení (`writeMode` je `false`). Pokud je soubor otevíraný pro zápis a neexistuje, je vytvořený nový (s nulovou délkou). Pokud je soubor otevíraný pro zápis a již existuje, je zkrácen na délku 0 bajtů (truncate). Konečně, pokud je soubor otevíraný pro čtení a neexistuje, ve vrácena chyba. Návratovou hodnotou je identifikátor otevřeného souboru (file deskriptor). Platné identifikátory otevřených souborů jsou kladná čísla a nula. Hodnota -1 znamená neúspěch (neexistující soubor otevíraný pro čtení, nově vytvářený soubor při zaplněném systému souborů).

`ReadFile, WriteFile`
: metody slouží pro čtení ze souboru/zápis do souboru. Parametry jsou podobné: `fd` udává identifikátor dříve otevřeného souboru (návratová hodnota z `OpenFile`), `buffer` je paměťový prostor, kam budou data načtená / odkud budou zapisovaná a `len` udává počet čtených/zapisovaných bajtů. Návratovou hodnotou metod je počet skutečně přečtených/zapsaných bajtů (např. pokud čteme u konce souboru, nemusí být splněn celý požadavek na čtení, načte se pouze zbývající část). Počet čtených/zapisovaných bajtů nemusí být násobkem velikosti sektoru, je úkolem Vaší implementace pomocí vyrovnávacích pamětí data ze sektorů/do sektorů poskládat.

`CloseFile`
: metoda uzavře dříve otevřený soubor (uloží případně nezapsaná data). Zadaný identifikátor je od tohoto okamžiku neplatný, může být případně použit pro práci s jiným souborem (ale až po volání `OpenFile`). Metoda `CloseFile` vrací `true` pro úspěch, `false` pro chybu.

`DeleteFile`
: metoda vymaže soubor zadaného jména. Pokud soubor existoval a byl úspěšně smazán, metoda vrací `true`. Pokud výmaz selže (neexistující soubor), bude vrácena hodnota `false`.

`FileSize`
: metoda zjistí velikost zadaného souboru. Pokud soubor existuje, metoda vrátí zjištěnou velikost jako návratovou hodnotu. Pokud soubor zadaného jména neexistuje, metoda vrátí `SIZE_MAX`.

`FindFirst`
: metoda slouží k nalezení souborů, které jsou na disku uložené. Pokud je disk prázdný (neobsahuje žádné soubory), metoda nechá výstupní parametr beze změny a vrátí hodnotu `false`. V opačném případě metoda ve výstupním parametru vyplní jméno a velikost prvního nalezeného souboru a vrátí hodnotu `true`.

`FindNext`
: metoda najde další soubor a podobně jako `FindFirst` vyplní jeho jméno a velikost a vrátí `true`. Pokud již žádný další soubor na disku není, metoda vrací hodnotu `false`. Pořadí souborů při procházení není důležité, Vaše implementace ale musí garantovat, že budou postupně nalezeny všechny existující soubory a že žádný soubor nebude vyjmenován vícekrát. Přiložený zdrojový kód obsahuje ukázku použití těchto metod pro výpis všech souborů.

Odevzdávejte zdrojový kód s implementací požadované třídy a s případnými dalšími podpůrnými funkcemi a třídami, které Vaše implementace potřebuje. Deklaraci struktury `TBlkDev`, definice konstant a vkládání hlavičkových souborů ponechte v bloku podmíněného překladu. Do Vaší implementace nevkládejte funkci `main` ani direktivy pro vkládání hlavičkových souborů. Funkci `main` a hlavičkové soubory lze ponechat pouze v případě, že jsou zabalené v bloku podmíněného překladu. Pro implementaci není k dispozici STL.

Nápověda:

- Rozhraní je navrženo tak, že lze vytvořit více najednou aktivních systémů souborů. Všechna data týkající se konkrétní instance systému souborů byste měli ukládat do instance `CFileSystem` vytvořené při volání metody `Mount`. Testovací prostředí nevytváří více instancí najednou, ale pracuje s více různými filesystémy, které průběžně přepíná (jeden odpojí, jiný připojí). Nezapomeňte proto na uvolnění nepotřebných prostředků v destruktoru `CFileSystem`.
- Ve Vaší implementaci si můžete alokovat paměť (celkem až řádově stovky KiB), kapacita disku ale bude několikrát větší než dostupná velikost paměti.
- Nezapomeňte, že veškeré I/O operace na disku probíhají s celými sektory (tedy s 512 B). Tedy např. pokud budete načítat 2 sektory najednou, musíte připravit paměťový blok velikosti 1024 B.
- Zápis několika sektorů najednou může I/O operace zrychlit. Počet najednou zapisovaných sektorů ale není rozumné přehánět (i kvůli omezení paměťových alokací).
- Využijte přiložený ukázkový soubor. V souboru je implementován jednoduchý diskový subsystém, který můžete použít jako základ pro Vaše testování. Pro důkladné otestování bude potřeba dodaný soubor rozšířit, zejména je potřeba implementovat mnohem důkladnější testování obsahu, zápisu a přepisu souborů.
- Vaše implementace je testována pouze jedním vláknem, sekvenčně. Nezabývejte se zamykáním.
- Požadované rozhraní neobsahuje adresáře. Tedy není důvod vyčleňovat zvláštní znaky pro oddělení cesty. Znaky '/' (dopředné lomítko) a '\\' (zpětné lomítko) jsou chápané jako "obyčejné" znaky, které se mohou vyskytovat kdekoliv v platném jménu souborů. Na znaky ve jménu souborů není kladeno žádné omezení, pouze délka jména souborů je shora omezena (`FILENAME_LEN_MAX`).
- Funkce `ReadFile` funguje tak, jak je obvyklé. Soubor je čten sekvenčně. Například předpokládáme soubor o délce 1234 bajtů a čteme jej třemi po sobě jdoucími voláními funkce `ReadFile`:
  1. `ReadFile ( fd, buffer, 654 )` načte prvních 654 bajtů do zadaného bufferu (bajty souboru s offsetem 0 až 653), návratová hodnota bude 654,
  2. `ReadFile ( fd, buffer, 400 )` načte dalších 400 bajtů do zadaného bufferu (bajty souboru s offsetem 654 až 1053 umístí do bufferu na pozice 0 až 399), návratová hodnota bude 400,
  3. `ReadFile ( fd, buffer, 285 )` načte zbývajících 180 bajtů (bajty souboru s offsetem 1054 až 1233 umístí do bufferu na pozice 0 až 179), návratová hodnota bude 180.
- Funkce `WriteFile` funguje tak, jak je obvyklé. Soubor je zapisován sekvenčně (append). Přepsání souboru je provedenou pouze na začátku při volání `OpenFile ( ..., true)`. Pokud postupně provedeme tři zápisy o délce 1234, 536 a 652 bajtů, bude mít výsledný soubor délku 1234 + 536 + 652 = 2422 B
- Testovací prostředí netestuje soubory v režimu sdílení. Tedy soubor je pro zápis otevřen max. jedním file descriptorem, zapisovaný soubor není zároveň čten a zapisovaný/čtený soubor není v průběhu operací vymazán (přesněji, testovací prostředí se nepokouší to dělat).

Co znamenají jednotlivé testy:

- `Kratke soubory, pouze zapis a vypis` disk je zformátován (`CreateFs`), připojen (`Mount`), na disk jsou zapisované soubory o velikosti do 4 KiB. Data nejsou čtena, jen se kontroluje seznam souborů na disku uložených a jejich velikosti. V jeden okamžik je otevřený nejvýše jeden soubor. Disk je po testu odpojen (`Umount`).
- `Kratke soubory, zapis, vypis, obsah` testování je obdobné, ale je porovnáván obsah souborů na disku uložených (data již jsou čtena).
- `Kratke soubory, zapis, vypis, obsah, vymaz` stejné jako předchozí, ale soubory jsou i mazány.
- `Libovolne soubory, vse (i vymaz)` stejné jako předchozí, ale soubory jsou i delší než 4KiB.
- `Kapacita FS` je testováno, zda se na disk skutečně vejde požadovaných 90 % kapacity souborových dat.
- `Mount (po rebootu)` disk není zformátován (tedy musí obsahovat data z minulého testu). Disk je pouze připojen (`Mount`) a je zkontrolován obsah. Navíc, mezi tímto a minulým testem je simulován restart počítače, tedy proměnné mezi těmito testy ztratí svůj obsah. Veškerá data, které při minulém `Umount` nebyla uložena na disk, jsou ztracena.
- `Vice otevrenych souboru` najednou je otevřeno více souborů. Zápisy a čtení souborů probíhají z jednoho vlákna (sekvenčně), ale file deskriptor otevřeného souboru je volen náhodně.

Vzorová data:	**Download**

☐ **Referenční řešení**

4	**13.06.2021 21:42:32**	**Download**
Stav odevzdání:	Ohodnoceno	
Hodnocení:	0.0000	

- **Hodnotitel: automat**
  - Chyba při kompilaci v režimu 'pedantic' - 10% penalizace
  - Test 'Kratke soubory, pouze zapis, vypis': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.004 s (limit: 10.000 s)
    - Využití paměti: 85528 KiB (limit: 113899 KiB)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Kratke soubory, zapis, vypis, obsah': Úspěch

        - Dosaženo: 100.00 %, požadováno: 100.00 %
        - Celková doba běhu: 0.005 s (limit: 9.996 s)
        - Využití paměti: 85528 KiB (limit: 113899 KiB)
        - Úspěch v závazném testu, hodnocení: 100.00 %
    - Test 'Kratke soubory, zapis, vypis, obsah, vymaz': Úspěch
        - Dosaženo: 100.00 %, požadováno: 100.00 %
        - Celková doba běhu: 0.005 s (limit: 9.991 s)
        - Využití paměti: 85528 KiB (limit: 113899 KiB)
        - Úspěch v závazném testu, hodnocení: 100.00 %
    - Test 'Libovolne soubory, vse (i vymaz)': Program provedl neplatnou operaci a byl ukončen (Segmentation fault/Bus error/Memory limit exceeded/Stack limit exceeded)
        - Celková doba běhu: 0.141 s (limit: 9.986 s)
        - Neúspěch v závazném testu, hodnocení: 0.00 %
    - Celkové hodnocení: 0.00 % (= (1.00 * 1.00 * 1.00 * 0.00) * 0.9)
- Celkové procentní hodnocení: 0.00 %
- Celkem bodů: 0.00 * 30.00 = 0.00

SW metriky:

	Celkem	Průměr	Maximum	Jméno funkce
Funkce:	**44**	--	--	--
Řádek kódu:	**439**	**9.98 ± 6.83**	**30**	`OpenedFile::ReadData`
Cyklomatická složitost:	**88**	**2.00 ± 1.26**	**6**	`OpenedFile::ReadData`

3 **13.06.2021 18:37:17** **Download**

Stav odevzdání: Ohodnoceno
Hodnocení: 0.0000

- **Hodnotitel: automat**
    - Chyba při kompilaci v režimu 'pedantic' - 10% penalizace
    - Test 'Kratke soubory, pouze zapis, vypis': Úspěch
        - Dosaženo: 100.00 %, požadováno: 100.00 %
        - Celková doba běhu: 0.004 s (limit: 10.000 s)
        - Využití paměti: 85156 KiB (limit: 113899 KiB)
        - Úspěch v závazném testu, hodnocení: 100.00 %
    - Test 'Kratke soubory, zapis, vypis, obsah': Úspěch
        - Dosaženo: 100.00 %, požadováno: 100.00 %
        - Celková doba běhu: 0.005 s (limit: 9.996 s)
        - Využití paměti: 85156 KiB (limit: 113899 KiB)
        - Úspěch v závazném testu, hodnocení: 100.00 %
    - Test 'Kratke soubory, zapis, vypis, obsah, vymaz': Úspěch
        - Dosaženo: 100.00 %, požadováno: 100.00 %
        - Celková doba běhu: 0.004 s (limit: 9.991 s)
        - Využití paměti: 85156 KiB (limit: 113899 KiB)
        - Úspěch v závazném testu, hodnocení: 100.00 %
    - Test 'Libovolne soubory, vse (i vymaz)': Program provedl neplatnou operaci a byl ukončen (Segmentation fault/Bus error/Memory limit exceeded/Stack limit exceeded)
        - Celková doba běhu: 0.116 s (limit: 9.987 s)
        - Neúspěch v závazném testu, hodnocení: 0.00 %
    - Celkové hodnocení: 0.00 % (= (1.00 * 1.00 * 1.00 * 0.00) * 0.9)
- Celkové procentní hodnocení: 0.00 %
- Celkem bodů: 0.00 * 30.00 = 0.00

SW metriky:

	Celkem	Průměr	Maximum	Jméno funkce
Funkce:	**43**	--	--	--
Řádek kódu:	**378**	**8.79 ± 5.97**	**30**	`OpenedFile::ReadData`
Cyklomatická složitost:	**77**	**1.79 ± 1.17**	**6**	`OpenedFile::ReadData`

2 **13.06.2021 01:32:31** **Download**

Stav odevzdání: Ohodnoceno
Hodnocení: 0.0000

- **Hodnotitel: automat**
    - Chyba při kompilaci v režimu 'pedantic' - 10% penalizace

- Test 'Kratke soubory, pouze zapis, vypis': Program provedl neplatnou operaci a byl ukončen (Segmentation fault/Bus error/Memory limit exceeded/Stack limit exceeded)
  - Celková doba běhu: 0.119 s (limit: 10.000 s)
  - Neúspěch v závazném testu, hodnocení: 0.00 %
- Celkové hodnocení: 0.00 %

- Celkové procentní hodnocení: 0.00 %
- Celkem bodů: 0.00 * 30.00 = 0.00

		Celkem	Průměr	Maximum	Jméno funkce
SW metriky:	Funkce:	**41**	--	--	--
	Řádek kódu:	**339**	**8.27 ± 5.34**	**26**	`OpenedFile::ReadData`
	Cyklomatická složitost:	**72**	**1.76 ± 1.08**	**5**	`OpenedFile::ReadData`

1	**13.06.2021 01:22:32**	**Download**
Stav odevzdání:	Ohodnoceno	
Hodnocení:	0.0000	

- **Hodnotitel: automat**
  - Chyba při základní kompilaci
- Celkové procentní hodnocení: 0.00 %
- Celkem bodů: 0.00 * 30.00 = 0.00