ZADÁNÍ SEMESTRÁLNÍ PRÁCE
SIMULÁTOR SOUBOROVÉHO SYSTÉMU

Zadání

Naprogramujte v ANSI C přenositelnou[1] **konzolovou aplikaci**, která načte ze souboru obraz souborového systému a seznam příkazů, které se mají nad souborovým systémem vykonat.

Program se bude spouštět příkazem: `fssim.exe` ⟨`files`⟩ ⟨`commands`⟩. Symbol ⟨`files`⟩ zastupuje jméno souboru s obrazem souborového systému a symbol ⟨`commands`⟩ zastupuje jméno souboru s příkazy (popis vstupních souborů bude uveden dále). Váš program tedy může být během testování spuštěn například takto:

```
...\>fssim.exe files.txt commands.txt
```

Výstupem programu bude výstup jednotlivých příkazů vykonaných nad simulovaným souborovým systémem vypsaný do příkazové řádky. Pokud nebudou uvedeny právě dva argumenty, vypište chybové hlášení a stručný návod k použití programu v angličtině podle běžných zvyklostí (viz např. ukázková semestrální práce na webu předmětu Programování v jazyce C). **Vstupem programu jsou pouze argumenty na příkazové řádce – interakce s uživatelem pomocí klávesnice či myši v průběhu práce programu se neočekává.**

Hotovou práci odevzdejte v jediném archivu typu ZIP prostřednictvím automatického odevzdávacího a validačního systému. Archiv nechť obsahuje všechny zdrojové soubory potřebné k přeložení programu, **makefile** pro Windows i Linux (pro překlad v Linuxu připravte soubor pojmenovaný `makefile` a pro Windows `makefile.win`) a dokumentaci ve formátu PDF vytvořenou v typografickém systému TEX, resp. LATEX. Bude-li některá z částí chybět, kontrolní skript Vaši práci odmítne.

Specifikace výstupu programu

Veškerý výstup programu bude směřován na standardní výstup procesu (`stdout`). Na výstupu bude vždy uveden vykonávaný příkaz uvozený znakem `$`. Na následujícím řádku pak bude uveden výstup provedeného příkazu. Výstup může vypadat například takto:

```
$pwd
/home/user/directory
$cd ..
/home/user
$ls
/home/user/dir1/
/home/user/dir2/
/home/user/file1
/home/user/file2
```

[1] Je třeba, aby bylo možné Váš program přeložit a spustit na PC s operačním prostředím Win32/64 (tj. operační systémy Microsoft Windows NT/2000/XP/Vista/7/8/10) a s běžnými distribucemi Linuxu (např. Ubuntu, Mint, OpenSUSE, Debian, atp.). Server, na který budete Vaši práci odevzdávat a který ji otestuje, má nainstalovaný operační systém Debian GNU/Linux 8.2 Jessie s jádrem verze 3.2.51-1 x86_64 a s překladačem gcc 4.7.2.

Specifikace vstupu programu

Vstupem programu jsou dva textové soubory. Jeden soubor obsahuje obraz souborového systému a druhý soubor obsahuje seznam příkazů, které se budou nad daným souborovým systémem vykonávat. Po spuštění je *kořenový adresář aktuálním adresářem.*

Pravidla pro pojmenování

Jméno souboru se skládá z číslic, malých a velkých písmen anglické abecedy. Jméno souboru může obsahovat nepovinnou koncovku souboru. Tato koncovka je oddělená tečkou (hodnota 0x2E v ASCII tabulce). Jméno souboru má délku v rozsahu 1-255 znaků (koncovka i s tečkou se započítavají do tohoto limitu).

Jméno adresáře se skládá z číslic, malých a velkých písmen anglické abecedy. Jméno adresáře může délku v rozsahu 1-255 znaků. U všech jmen **záleží** na velikosti písmen. Jméno adresáře *vždy* končí znakem `/` (hodnota 0x2F v ASCII tabulce). Na každém řádku textového souboru bude uvedena absolutní cesta k jednomu souboru nebo adresáři. Pojmenování souborů a adresářů je inspirováno konvencemi operačního systému Linux.

Všechny absolutní cesty začínají ve speciálním kořenovém adresáři. Tento adresář nemá jméno a je proto v textu reprezentován pouze jako `/`. Každý adresář může obsahovat další adresáře a soubory. Soubor nemůže obsahovat žádné prvky souborového systému.
Vstupní soubor může vypadat například takto:

```
/home/user/dir1/ /*prázdný adresář*/
/home/user/dir2/file /*soubor bez přípony*/
/home/user/file1.xzy /*soubor s příponou*/
```

Absolutní a relativní cesta

Všechny příkazy, pro které to má smysl musí podporovat práci s absolutní a relativní cestou. Absolutní cesta vždy začíná kořenovým adresářem (např. `/home/file`).

Relativní cesta se vždy vztahuje k aktuálnímu adresáři (získaný příkazem `pwd`). Relativní cesta tak nikdy nebude začínat lomítkem, protože by se jednalo o cestu absolutní. Relativní cesta může obsahovat namísto jména adresáře sekvenci `../` (dvě tečky a lomítko). Tato sekvence označuje nadřazený adresář pro aktuální adresář. Tyto sekvence mohou být dále zřetězeny, pokud je požadovaný adresář nebo soubor o několik úrovní výše. Relativní cesta pak může vypadat například takto: `../../file` – jedná se o soubor `file` o dvě úrovně výše, než je aktuální adresář.

Posun o adresář výše se může vyskytovat i uprostřed cesty. Cesta `/home/user/../..` tak označuje kořenový adresář. Při zpracování cesty je nutno ověřit, že každý uvedený adresář existuje! A to i v případě, že uvedený adresář nebude použitý. Cesta `dir1/missing/..` označuje adresář `dir1`, pokud však adresář `missing/` neexistuje, cesta není validní.

Seznam příkazů

Druhý soubor obsahuje seznam příkazů, které se budou vykonávat nad simulovaným souborovým systémem. Příkazy mohou mít argumenty, argumenty jsou dvou typů – povinné (označené `<>`) a nepovinné (označené `[]`). Podporovány musí být následující příkazy:
`pwd`

- vypíše absolutní cestu k aktuálnímu adresáři

`cd <path>`

- změní aktuální adresář na `<path>`. Parametr je absolutní nebo relativní cesta, která určuje jak se změní aktuální adresář.
- pokud parametr `<path>` chybí, vypíše `ERROR not enough arguments`

- pokud cesta neexistuje, vypíše `ERROR path does not exist`
- pokud je cílem soubor, vypíše `ERROR target is not a directory`

`ls [path]`

- vypíše obsah adresáře, parametr `[path]` je nepovinný
- pokud `[path]` není zadaný, tak vypíše aktuální adresář (určený pomocí `pwd`)
- pokud `[path]` je zadaný, tak vypíše určený adresář
- pokud `[path]` ukazuje na soubor, `ls` vypíše pouze jméno souboru
- vypíše každý soubor/adresář na jeden řádek, seznam je seřazený podle abecedy. Pořadí pro jednotlivé skupiny povolených znaků je následující: malá písmena mají přednost před velkými, písmena mají přednost před číslicemi.
- pokud `[path]` neexistuje, vypíše `ERROR path does not exist`

`mv <source> [target]`

- přesune soubor `<source>` do adresáře `[target]`
- pokud `[target]` není zadaný, použije se aktuální adresář (`pwd`)
- pokud parametr `<source>` chybí, vypíše `ERROR not enough arguments`
- pokud `<source>` neexistuje, vypíše `ERROR source file not found`
- pokud `<source>` je adresář, vypíše `ERROR source is directory`
- pokud adresář `[target]` obsahuje soubor stejného jména jako `<source>`, bude tento soubor přepsán bez varování

`cp <source> [target]`

- zkopíruje soubor `<source>` do adresáře `[target]`
- pokud `[target]` není zadaný, použije se aktuální adresář (`pwd`)
- pokud parametr `<source>` chybí, vypíše `ERROR not enough arguments`
- pokud `<source>` neexistuje, vypíše `ERROR source file not found`
- pokud `<source>` je adresář, vypíše `ERROR source is directory`
- pokud adresář `[target]` obsahuje soubor stejného jména jako `<source>`, bude tento soubor přepsán bez varování

`find <what> [where]`

- najde všechny soubory se jménem `<what>` v adresáři `[where]` a jeho podadresářích
- pokud `[where]` není zadaný, použije se aktuální adresář (`pwd`)
- pokud `[where]` neexistuje, vypíše `ERROR path does not exist`
- pokud parametr `<what>` chybí, vypíše `ERROR not enough arguments`
- parametr `<what>` je textový řetězec představující jméno hledaného souboru. Parametr sestává ze znaků platných pro jméno souboru. Při vyhledávání záleží na velikosti písmen. Parametr dále může obsahovat symbol `*`, který zastupuje libovolný počet znaků (0-255) platných pro jméno souboru.
- pokud nebyl nalezen žádný soubor, výstupem je prázdný řádek

Soubor s příkazy může vypadat například takto:

```
pwd
cd /home/user/
cd ../sytem/
ls
ls /home/user/Music/
mv /home/user/Music/file.mp3 /Trash/
find *.mp3 /home/user/Music/
```

Užitečné techniky a odkazy

Uvedené techniky je možné (ale nikoliv nezbytně nutné) využít při řešení úlohy. Protože se jedná o postupy standardní, lze k nim nalézt velké množství dokumentace:

1. n-ární strom,
2. rekurze,
3. systémy souborů,
4. porovnávání textových řetězců.

Řešení úlohy je zcela ve vaší kompetenci – zvolte takové algoritmy a techniky, které podle vás nejlépe povedou k cíli.