

**Souborový systém FAT**

KIV/ZOS – Semestrální práce

student: *Jiří Velek*

osobní číslo: *A20B0269P*

email: [jvelek@students.zcu.cz](mailto:jvelek@students.zcu.cz)

Obsah

[1 Zadání 3](#_Toc121409118)

[2 FAT tabulka 3](#_Toc121409119)

[3 Adresář a jeho obsah 3](#_Toc121409120)

[4 Hlavička souborového systému 3](#_Toc121409121)

[5 Implementace 3](#_Toc121409122)

[1.1 Hlavička souborového systému 3](#_Toc121409123)

[1.2 FAT tabulka 4](#_Toc121409124)

[1.3 Adresářová struktura 4](#_Toc121409125)

[1.4 Příkazy 4](#_Toc121409126)

[6 Závěr 4](#_Toc121409127)

# Zadání

Tématem semestrální práce bude práce se zjednodušeným souborovým systémem založeným na pseudoFAT. Vaším cílem bude splnit několik vybraných úloh. Základní funkčnost, kterou musí program splňovat. Formát výpisů je závazný. Program bude mít jeden parametr a tím bude název Vašeho souborového systému. Po spuštění bude program čekat na zadání jednotlivých příkazů.

# FAT tabulka

Souborový systém FAT (File Allocation Table) je založen na principu indexované tabulky, která identifikuje „řetězy“ bloků asociované se souborem. Této tabulce se říká FAT a je staticky alokovaná při formátování oddílu. Tato tabulka je zřetězeným seznamem bloků. Každý tento zřetězený seznam je (většinou) seznam po sobě jdoucích bloků tak, aby operační systém mohl tento seznam přečíst, a poté přečíst obsah těchto jednotlivých bloků, dokud nenarazí na speciální značku indikující konec souboru.

# Adresář a jeho obsah

Souborový systém se skládá z adresářů, při formátování je vytvořen prázdný kořenový adresář. Každý adresář má přidělen počáteční blok, stejně jako každý soubor. Adresář obsahuje název (max. 11 znaků), číslo bloku, kde soubor(adresář) začíná, atributy, a velikost souboru. Každý adresář označený jako složka obsahuje minimálně odkaz na zpětný adresář (..) a odkaz na současný adresář (.).

# Hlavička souborového systému

Souborový systém začíná hlavičkou. Je uložena na prvním sektoru disku a obsahuje celkový počet bloků, velikost jednoho bloku, počátek oddílu, velikost a počet FAT tabulek. Všechny tyto hodnoty jsou inicializovány při formátu oddílu a měla by zůstat konstantní po celou dobu používání disku.

# Implementace

Souborový systém byl implementován v jazyce Rust.

## Hlavička souborového systému

Hlavička souborového systému je implementována jako struct, který obsahuje hodnoty, počet sektorů, počet FAT tabulek, hodnotu kontrolního součtu, kolik bytů obsahuje jeden sektor a kolik sektorů obsahuje jeden cluster.

Tato tabulka je vyplněna podle parametru zadaného příkazu format. Bytů na sektor je vždy 512, jedna FAT tabulka, sektorů na cluster 8. Kontrolní součet se využívá ke kontrole, zda je souborový systém správně naformátován.

## FAT tabulka

FAT tabulka je na disku zapsána hned za hlavičkou (na druhém sektoru). Její hodnoty jsou uloženy jako 4B celá čísla. Program s tabulkou pracuje tak, že si přečte pouze daný sektor, kde se nachází další číslo clusteru a přečte/zapíše další cluster. Při každé změně FAT tabulky překopírujeme FAT1 do FAT2. Při alokaci clusterů hledáme clustery označené číslem 0. Pokud jich není nalezeno dostatek, překopíruje se FAT2 do FAT1 (změny se vrátí).

## Adresářová struktura

Adresář je v programu implementován jako struct, který obsahuje název souboru, číslo clusteru a velikost souboru. Při čtení adresáře se přečte cluster a poté se ve FAT tabulce indexuje do dalšího, dokud není nalezena značka konce souboru. Složky mají „velikost souboru“ rovno nula.

## Serializace a deserializace

Serializace a deserializace jsou v programu řešeny manuálně, v Rustu není možné bezpečně (kvůli endianitě) vzít objekt (struct) a zapsat ho do souboru. Při serializaci jsou všechna pole, která se mají zapsat do souboru nejprve převedena na little endian byty a poté zapsána do souboru po jednom. Při deserializaci obdobně čte po „velikostích jednotlivých polí“ a pole jsou dekódována z little endian do nativní endianity tak, aby program fungoval i na různých platformách.

## Příkazy

Pro každý příkaz existuje jeden struct (třída), která implementuje Command trait. Tento trait obsahuje funkci handle, která daný příkaz obsluhuje.

# Závěr

Psaní této práce mi ukázalo, že bych neměl dělat více práce, než je požadováno, a že bych měl číst zadání dříve, než začnu pracovat. Můj chtíč po plně funkčním souborovém systému mě stál zbytečně moc času, a proto kód vypadá tak, jak vypadá. Zároveň je ale kód připraven na různá rozšíření, například změny souboru, počítání volných clusterů, nebo ukládání posledního volného clusteru. Jelikož jsem se souborovým systémem FAT pracoval už dříve, práce mě obohatila pouze o to, jak funguje rekurzivní procházení adresářů a alokace bloků. Práce byla původně napsána v jazyce C, ale na poslední chvíli jsem se rozhodl celou práci přepsat v jazyce Rust.