VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2017 Bc. Tomáš Konečný

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

OPRAVA METADAT SOUBOROVÉHO SYSTÉMU FAT32
REPAIRING FAT32 FILE SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Tomáš Konečný

VEDOUCÍ PRÁCE
SUPERVISOR
doc. Ing. Karel Burda, CSc.

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Tomáš Konečný ***ID:*** 155133
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Oprava metadat souborového systému FAT32

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte a popište souborový systém FAT32 a dostupné programy pro obnovu metadat uvedeného souborového systému (gpart, TestDisk apod.). Na tomto základě navrhněte a prakticky zrealizujte vlastní program pro obnovu poškozených metadat v souborovém systému FAT32 v USB flash disku. Vytvořený program prakticky otestujte a porovnejte jeho možnosti a výkonnost s obdobnými programy.

DOPORUČENÁ LITERATURA:

[1] Compact flash memory card driver technical manual. [online]. [cit. 7. 9. 2016]. Dostupné z URL: <goo.gl/n7qL4k>

[2] Microsoft Extensible Firmware Initiative FAT32 File System Specification. [online]. [cit. 16. 10. 2016]. Dostupné z URL: <http://goo.gl/jrJkow>.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 24.5.2017

Vedoucí práce: doc. Ing. Karel Burda, CSc.
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Tato práce se věnuje problematice obnovy metadat souborového systému FAT32 z poškozeného flash disku či paměťové karty a návrhu programu pro realizaci této obnovy s porovnáním s existujícím řešením pro tento úkon.

KLÍČOVÁ SLOVA

Souborový systém, obnova dat, FAT32, ddrescue, flash disk, paměť, sektor, cluster, TestDisk

ABSTRACT

This work relates to problematics of FAT32 file system metadata recovery from damaged flash drive or memory card and design of program capable of such recovery and comparision with existing solution for this task.

KEYWORDS

File system, data recovery, FAT32, ddrescue, flash drive, memory, sector, cluster, TestDisk

KONEČNÝ, Tomáš *Oprava dat souborového systému FAT32*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 58 s. Vedoucí práce byl doc. Ing. Karel Burda, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Oprava dat souborového systému FAT32“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Karlu Burdovi, CSc. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Pro splnění zadání této diplomové práce bylo zapotřebí pečlivě a dopodrobna nastudovat problematiku ohledně souborových systémů, zejména pak systému FAT32, který je v zadání přímo uveden. Pro splnění tohoto úkolu pomohla hlavně technická specifikace vydána přímo společností Microsoft, která je autorem FAT32. Na základě této nastudované teorie a vlastního testování jednotlivých sekcí FAT32 pak proběhl návrh samotného programu, který má dle zadání zvládat obnovovat metadata souborového systému, pokud by došlo k jejich poškození. Výsledný program by následně měl být otestován a funkčně porovnán s již existujícím řešením pro tyto účely (program TestDisk).

1 REPREZENTACE DAT NA MEDIU

Tato kapitola obsahuje obecné informace o souborových systémech, jejich účelu, typech a způsobech adresace dat na discích.

1.1 Souborový systém

Zjednodušeně řečeno, souborový systém (angl. file system) nám určuje, jak a kde jsou jednotlivá data uložena na zápisovém médiu. Tato paměťová media můžou být například plotnový disk, CD, DVD, magnetická páska, nebo paměť typu flash (mobilní telefon, mp3 přehrávač, USB disk, apod.). Souborový systém rozděluje uložená data, jak už název napovídá, do jednotlivých souborů, adresářů a podadresářů. Je pomocí něj určeno, na kterém místě v paměti jednotlivé soubory začínají, na kterém končí, kdy byly vytvořeny nebo kdo je jejich vlastníkem a jaké má práva. Informace uchovávané souborovým systémem tedy můžeme rozdělit na data a metadata. Data jsou vlastní informace souborů, které chceme na paměťovém mediu ukládat, metadata pak tvoří výše uvedené informace o umístění, časových značkách, o vlastníku a podobně [1]. Z předchozího popisu je tedy patrné, že bez použití nějakého souborového systému, který zajišťuje striktní reprezentaci dat na paměťovém mediu, by byla práce s uloženými daty prakticky nemožná, jelikož by se jednalo o prostou posloupnost dvojkových čísel bez jakéhokoliv popisu.

1.1.1 Běžně používané souborové systémy

S různými druhy souborových systémů se setkáváme zejména v prostředí osobních počítačů a přenosných zařízení. Takových souborových systému je nespočet, jmenovitě například FAT (FAT12, FAT16, FAT32), NTFS, ext3, ext4, HFS+. Použitelnost jednotlivých souborových systémů se odvíjí od použitého operačního systému (například HFS+ z OS X je pro Windows nečitelný) a dalších omezení, mezi která patří zejména [2]:

- Velikost paměťového média kterou je daný systém schopen pokrýt
- Délka souboru
- Délka jména souboru
- Počet zanořených podadresářů
- Podporovaná znaková sada

Žurnálové souborové systémy

Některé typy souborových systémů si uchovávají speciální záznamy o změnách provedených na uložených souborech. Tyto záznamy se nazývají žurnály. Pokud dojde

k neočekávané nehodě (pád operačního systému, přerušení napájení a podobně) ztratí po opětovném startu systému data a metadata vzájemnou integritu. Díky žurnálům je tedy možné dohledat prováděné změny, dokončit je, případně vrátit zpět a integritu tak obnovit [3].

1.2 Organizace dat na disku

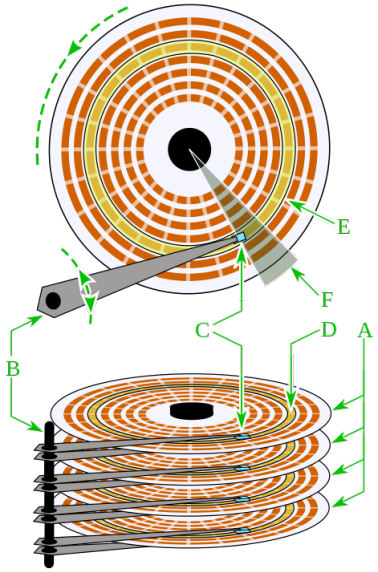
1.2.1 Základní pojmy

- Plotna - Kotouč s nanesenou magnetickou vrstvou
- Hlava - Zařízení pro čtení i zápis na plotnu
- Cylindr (Track) - Soustředná kružnice na plotně složená ze sektorů
- Sektor - Minimální fyzicky adresovatelná jednotka na disku, většinou o velikosti 512B [4]. U starších disků (adresace CHS) se jako sektor označovala celá kruhová výseč, protože se počet sektorů na vnitřní a vnější stopě neměnil a adresace výseče tak znamenala adresaci jednoho sektoru, jakožto elementární datové buňky disku. U novějších disků (adresace LBA) se pak jako sektor rozumí pouze ona elementární jednotka na disku.
- Cluster - Na rozdíl od sektoru je cluster nejmenší logicky adresovatelná jednotka na disku a skládá se z několika sektorů. Potřeba zavedení clusteru pramení ze zvětšování počtu sektorů a přímo úměrné zvyšování režie při zápisu souborů (adresace vysokého počtu sektorů). Spojením několika sektorů do clusterů se tak redukuje míra režie a práce s daty je rychlejší. Pokud by byl naopak cluster příliš velký, docházelo by k plýtvání kapacity media (do jednoho clusteru nelze zapsat více souborů, takže při zápisu malých souborů by se pro každý z nich musel adresovat nový cluster). Příklad tohoto chování je dobře vidět při zobrazení vlastností malých souborů v počítači, jako například libovolná ikona 1.1. Ačkoliv je skutečná velikost této ikony jen 1,8 kB, zabírá na disku 4 kB, což odpovídá velikosti clusteru v daném souborovém systému. Velikost clusteru se tedy liší pro různé souborové systémy a velikosti disků, či diskových oddílů.

Obr. 1.1: Velikost clusteru

1.2.2 CHS

Starší způsob adresace dat na pevném disku se nazývá Cylinder-Head-Sector (cylindr-hlava-sektor), zkráceně tedy CHS. Jak je zobrazeno na obrázku 1.2, adresace probíhá pomocí tří určujících souřadnic, kdy jednotlivé souřadnice vyjadřují konkretní cylindr (D, E), záznamovou a čtecí hlavu (C) na pohyblivém ramínku (B) a sektor (F). Zároveň je na tomto obrázku zobrazena konstrukce disku s vícero magnetickými plotnami (A) [5].



Obr. 1.2: Struktura disku [5]

1.2.3 LBA

Novější způsob adresace disku, nezkráceným názvem Logical Block Addressing. Od metody Cylinder-Head-Sector se liší zejména tím, že data na disku adresuje lineárně. Informace na disku jsou pak ve výsledku reprezentována vektorem po sobě jdoucích bajtů, uspořádaných do sektorů. Stejný druh adresace se pak používá i u medií s maticově uspořádanými paměťovými buňkami, jako například flash paměti, takže pochopení principu adresace u magnetických disků v podstatě představuje pochopení čtení a zápisu i u moderních paměťových zařízení [6] .

2 FAT32

V této kapitole je podrobně popsána struktura souborového systému FAT32, jeho funkční bloky, organizace na disku a technická omezení.
Souborový systém FAT32 je nástupcem souborového systému FAT16 z rodiny FAT, z něhož tento nový systém vychází. Potřeba pro vytvoření nového souborového systému spočívala zejména v možnosti aplikovat FAT souborové systémy na pevné disky větších kapacit. Systém FAT16 dokázal adresovat diskové oddíly o maximální velikosti 4 GB. Nový systém FAT32 pak dokázal adresovat oddíly až do velikosti 2 TB pro sektory o velikosti 512 bajtů a velikost jednotlivých souborů byla omezena na 4 GB (FAT32 se tedy nehodí pro ukládání velkých souborů, jako například dlouhá videa ve vysokém rozlišení a podobně). Podle společnosti Microsoft, což je autor souborových systémů FAT, je maximální diskový oddíl pro FAT32 limitován na 32 GB. Toto omezení je dáno historicky z dob používání MS DOS, který pro práci s diskem využívá BIOS a výsledná velikost diskového oddílu naráží na limity CHS adresace [7]. Minimální počet clusterů ve FAT32 je 65 525. Toto je jediný způsob, jak operační systém ověřuje, o jakou verzi FAT se jedná (pokud je clusterů méně než 4 085, jedná se o FAT12, pokud je clusterů méně než 65 525, jedná se o FAT16 a pokud je jich více, jedná se o FAT32) [8]. Výslednou velikost diskového oddílu neurčuje pouze počet clusterů, ale podílí se na ní logicky i velikost jednotlivých clusterů. Tato velikost se liší podle velikosti požadovaného oddílu a platí zde přímá úměra (čím větší oddíl, tím větší cluster), jak je ukázáno v tabulce 2.1.

Velikost svazku	**Windows NT 3.51**	**Windows 2000, XP, Vista, 7 Windows Server 2003, 2008**
7 MB–16 MB	Není podporováno	Není podporováno
16 MB–32 MB	512 bajtů	Není podporováno
32 MB–64 MB	512 bajtů	512 bajtů
64 MB–128 MB	1 KB	1 KB
128 MB–256 MB	2 KB	2 KB
256 MB–8 GB	4 KB	4 KB
8 GB–16 GB	8 KB	8 KB
16 GB-32 GB	16 KB	16 KB
32 GB–2 TB	32 KB	Není podporováno
> 2TB	Není podporováno	Není podporováno

Tab. 2.1: Výchozí velikosti clusteru pro FAT32 souborový systém [9]

2.1 Struktura FAT32

Souborové systémy FAT byly vyvíjeny pro operační systém DOS. DOS uvažuje disky jako lineární objekty, nebere tedy v potaz skutečnou geometrii disku. To znamená, že DOS pracuje s prostorem na disku jako s lineární sekvencí sektorů od prvního k poslednímu. Tahle skutečnost umožňuje jednodušší práci při manipulaci s moderními paměťovými médii typu flash, kde jsou jednotlivé paměťové buňky uspořádány lineárně. Je nutno podotknout, že FAT (File Allocation Table) je jednak označení celého souborového systému, ale zároveň se jedná i o jeho funkční součást, tedy samotnou tabulku, která určuje kde a jak jsou uložená data [4].

2.1.1 Pořadí bitů

Systémy FAT využívají tzv. little endian. To znamená, že první bajt, který se při čtení nějakého záznamu načítá, je zároveň nejméně významný bajt, tzv. Least Significant Byte (LSB). Logicky pak poslední čtený bajt je nejvíce významný, tedy Most Significant Byte (MSB). Tato logika platí i pro bity, ze kterých se skládají bajty, tedy že nejdřív se načítají ty nejméně významné [4] [8]. Příklad 32bitového záznamu FAT32 je uveden na obrázku 2.1.

```
byte[3]         3 3 2 2 2 2 2 2
                1 0 9 8 7 6 5 4

byte[2]         2 2 2 2 1 1 1 1
                3 2 1 0 9 8 7 6

byte[1]         1 1 1 1 1 1 0 0
                5 4 3 2 1 0 9 8

byte[0]         0 0 0 0 0 0 0 0
                7 6 5 4 3 2 1 0
```

Obr. 2.1: 32bitový záznam FAT32 [8]

Tento záznam je rozdělen do čtyř bajtů, každý obsahující 8 bitů indexovaných 00-31 (00 je least significant bit). Pro čtení a zápis dat je potřeba znát, jestli daný systém využívá little endian, nebo big endian z důvodu kompatibility uložených, nebo čtených dat. V případě rozdílnosti je potřeba zajistit překlad adresování [8].

2.1.2 Rozvržení FAT32 oddílu

V následujících tabulkách je zobrazeno, jak jsou jednotlivé sekce FAT32 disku organizovány. Šedivě zabarvené buňky představují pole, která se můžou objevit vícekrát, nebo také vůbec [4].

Počáteční adresa	**Velikost**	**Obsah**	
0x00000000	512 bajtů	Master Boot Record (MBR)	
Adresa začátku oddílu + 0	512 bajtů	1. Oddíl	Boot Record (umístěn v prvním sektoru oddílu)
Adresa začátku oddílu + 512	Specifikováno v Boot Record		FAT tabulka 1
Adresa začátku oddílu + 512 + (velikost FAT tabulky x (FAT tabulka # - 1))	Specifikováno v Boot Record		FAT tabulka # (specifikováno v Počet kopií FAT v MBR, běžně 2)
Adresa začátku oddílu + 512 + (velikost FAT tabulky x počet FAT kopií)	Vypočítáno z MBR (Celková velikost oddílu)		Datová oblast pro soubory a adresáře

Tab. 2.2: Základní struktura FAT32 (1. oddíl) [4] [8]

Pokud je vytvořen více než jeden oddíl, pak následují další oddíly:

Počáteční adresa	**Velikost**	**Obsah**	
Adresa začátku oddílu + 0	512 bajtů	2. Oddíl	Boot Record (umístěn v prvním sektoru oddílu)
Adresa začátku oddílu + 512	Specifikováno v Boot Record		FAT tabulka 1
Adresa začátku oddílu + 512 + (velikost FAT tabulky x (FAT tabulka # - 1))	Specifikováno v Boot Record		FAT tabulka # (specifikováno v Počet kopií FAT v MBR, běžně 2)
Adresa začátku oddílu + 512 + (velikost FAT tabulky x počet FAT kopií)	Vypočítáno z MBR (Celková velikost oddílu)		Datová oblast pro soubory a adresáře

Tab. 2.3: Základní struktura FAT32 (2. oddíl) [4] [8]

2.2 Master Boot Record

První sektor disku je vyhrazen pro Master Boot Record (MBR). Toto je nezávislé na operačním systému. Master Boot Record obsahuje tabulku oddílů, zvanou též Partition Table, která definuje různé sekce pevného disku, nebo jiného záznamového media [4].

<table>
<tr><th colspan="2">Bajt
0x00000000+#</th><th colspan="3">Hodnota</th></tr>
<tr><td>0</td><td>0x0000</td><td colspan="3" rowspan="3">446 bajtů bootovacího spustitelného kódu a dat.</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>445</td><td>0x01BD</td></tr>
<tr><td>446</td><td>0x01BE</td><td rowspan="16">1. Oddíl</td><td>Offset 0x00</td><td>Stav oddílu (00h=Neaktivní, 80h=Aktivní)</td></tr>
<tr><td>447</td><td>0x01BF</td><td>Offset 0x01</td><td>Začátek oddílu (Hlava)</td></tr>
<tr><td>448</td><td>0x01C0</td><td>Offset 0x02</td><td>Začátek oddílu (Cylindr/Sektor)</td></tr>
<tr><td>449</td><td>0x01C1</td><td>Offset 0x03</td><td>
<table>
<tr><th>15 14 13 12 11 10 9 8</th><th>7 6</th><th>5 4 3 2 1 0</th></tr>
<tr><td>Bity cilindru 7 - 0</td><td>Bity cilindru 9+8</td><td>Bity sektoru 5 - 0</td></tr>
</table>
</td></tr>
<tr><td>450</td><td>0x01C2</td><td>Offset 0x04</td><td>Typ oddílu
0x00 Neznámý nebo žádný
0x01 12-bit FAT
0x04 16-bit FAT (<32MB)
0x05 Rozšířený MS-DOS oddíl
0x06 16-bit FAT (>32MB)
0x0B 32-bit FAT (až 2048GB)
0x0C Jako 0x0B, používá LBA 0x13 přípony
0x0E Jako 0x06, používá LBA 0x13 přípony
0x0F Jako 0x05, používá LBA 0x13 přípony
Hodnoty výše se vztahují k operačním systémům Microsoft</td></tr>
<tr><td>451</td><td>0x01C3</td><td>Offset 0x05</td><td>Konec oddílu (Hlava)</td></tr>
<tr><td>452</td><td>0x01C4</td><td>Offset 0x06</td><td>Konec oddílu (Cylindr/Sektor)</td></tr>
<tr><td>453</td><td>0x01C5</td><td>Offset 0x07</td><td>
<table>
<tr><th>15 14 13 12 11 10 9 8</th><th>7 6</th><th>5 4 3 2 1 0</th></tr>
<tr><td>Bity cilindru 7 - 0</td><td>Bity cilindru 9+8</td><td>Bity sektoru 5 - 0</td></tr>
</table>
</td></tr>
<tr><td>454</td><td>0x01C6</td><td>Offset 0x08</td><td rowspan="4">Počet sektorů mezi MBR a prvním sektorem oddílu</td></tr>
<tr><td>455</td><td>0x01C7</td><td>Offset 0x09</td></tr>
<tr><td>456</td><td>0x01C8</td><td>Offset 0x0A</td></tr>
<tr><td>457</td><td>0x01C9</td><td>Offset 0x0B</td></tr>
<tr><td>458</td><td>0x01CA</td><td>Offset 0x0C</td><td rowspan="4">Počet sektorů oddílu</td></tr>
<tr><td>459</td><td>0x01CB</td><td>Offset 0x0D</td></tr>
<tr><td>460</td><td>0x01CC</td><td>Offset 0x0E</td></tr>
<tr><td>461</td><td>0x01CD</td><td>Offset 0x0F</td></tr>
</table>

Tab. 2.4: Struktura MBR (1. oddíl) [4] [8]

462	0x01CE	2. Oddíl	Offset 0x00	Stav oddílu (00h=Neaktivní, 80h=Aktivní)
463	0x01CF		Offset 0x01	Začátek oddílu (Hlava)
464	0x01D0		Offset 0x02	Začátek oddílu (Cylindr/Sektor)
465	0x01D1		Offset 0x03	(Formát jako v 1. oddílu)
466	0x01D2		Offset 0x04	Typ oddílu (Formát jako v 1. oddílu)
467	0x01D3		Offset 0x05	Konec oddílu (Hlava)
468	0x01D4		Offset 0x06	Konec oddílu (Cylindr/Sektor)
469	0x01D5		Offset 0x07	(Formát jako v 1. oddílu)
470	0x01D6		Offset 0x08	Počet sektorů mezi MBR a prvním sektorem
471	0x01D7		Offset 0x09	oddílu
472	0x01D8		Offset 0x0A	
473	0x01D9		Offset 0x0B	
474	0x01DA		Offset 0x0C	Počet sektorů oddílu
475	0x01DB		Offset 0x0D	
476	0x01DC		Offset 0x0E	
477	0x01DD		Offset 0x0F	
478	0x01DE	3. Oddíl	Offset 0x00	Stav oddílu (00h=Neaktivní, 80h=Aktivní)
479	0x01DF		Offset 0x01	Začátek oddílu (Hlava)
480	0x01E0		Offset 0x02	Začátek oddílu (Cylindr/Sektor)
481	0x01E1		Offset 0x03	(Formát jako v 1. oddílu)
482	0x01E2		Offset 0x04	Typ oddílu (Formát jako v 1. oddílu)
483	0x01E3		Offset 0x05	Konec oddílu (Hlava)
484	0x01E4		Offset 0x06	Konec oddílu (Cylindr/Sektor)
485	0x01E5		Offset 0x07	(Formát jako v 1. oddílu)
486	0x01E6		Offset 0x08	Počet sektorů mezi MBR a prvním sektorem
487	0x01E7		Offset 0x09	oddílu
488	0x01E8		Offset 0x0A	
489	0x01E9		Offset 0x0B	
490	0x01EA		Offset 0x0C	Počet sektorů oddílu
491	0x01EB		Offset 0x0D	
492	0x01EC		Offset 0x0E	
493	0x01ED		Offset 0x0F	
494	0x01EE	4. Oddíl	Offset 0x00	Stav oddílu (00h=Neaktivní, 80h=Aktivní)
495	0x01EF		Offset 0x01	Začátek oddílu (Hlava)
496	0x01F0		Offset 0x02	Začátek oddílu (Cylindr/Sektor)
497	0x01F1		Offset 0x03	(Formát jako v 1. oddílu)
498	0x01F2		Offset 0x04	Typ oddílu (Formát jako v 1. oddílu)
499	0x01F3		Offset 0x05	Konec oddílu (Hlava)
500	0x01F4		Offset 0x06	Konec oddílu (Cylindr/Sektor)
501	0x01F5		Offset 0x07	(Formát jako v 1. oddílu)
502	0x01F6		Offset 0x08	Počet sektorů mezi MBR a prvním sektorem
503	0x01F7		Offset 0x09	oddílu
504	0x01F8		Offset 0x0A	
505	0x01F9		Offset 0x0B	

506	0x01FA	4. Oddíl	Offset 0x0C	Počet sektorů oddílu
507	0x01FB		Offset 0x0D	
508	0x01FC		Offset 0x0E	
509	0x01FD		Offset 0x0F	
510	0x01FE	Boot signature (= 0xAA55)		
511	0x01FF			

Tab. 2.5: Struktura MBR (další oddíly) [4] [8]

2.3 Boot Record

První sektor každého oddílu obsahuje tzv. Boot Record, nazývaný též Boot Sector, Nultý sektor, BPB (BIOS Parameter Block), nebo dalšími podobnými názvy. Jedná se o zavaděč operačního systému [8].

Offset	**Popis**
0x0000	Skoková instrukce k boot kódu + číslo oddílu
0x0001	
0x0002	
0x0003	8bajtový OEM Název
...	
0x000A	
0x000B	Počet bajtů na sektor
0x000C	
0x000D	Počet Sektorů na Cluster (mocniny 2)
0x000E	Rezervované sektory
0x000F	
0x0010	Počet kopií FAT (doporučené 2)
0x0011	Maximum zápisů kořenového adresáře
0x0012	(FAT32 nepoužívá, vždy 0)
0x0013	Počet sektorů pro oddíly menší než 32MB
0x0014	(FAT32 nepoužívá, vždy 0)
0x0015	Popis Media (0xF0 pro vyměnitelná media)
0x0016	Počet sektorů na FAT
0x0017	(FAT32 nepoužívá, vždy 0)
0x0018	Počet sektorů na stopu
0x0019	
0x001A	Počet hlav
0x001B	
0x001C	Počet skrytých sektorů v oddílu
0x001D	
0x001E	
0x001F	

0x0020	Počet sektorů v oddílu
0x0021	
0x0022	
0x0023	
0x0024	Počet sektorů ve FAT
0x0025	
0x0026	
0x0027	
0x0028	Příznaky:
0x0029	Bity 15-8 a 6-4: Rezervovány Bit 7: 1=FAT zrcadlení zakázáno, 1 FAT aktivní, 0=FAT zrcadlení do všech FAT Bity 3-0: Počet aktivních FAT (0-#)Jen při zakázaném zrcadlení
0x002A	Verze FAT32 jednotky (vyšší bajt = major verze, nižší bajt = minor verze)
0x002B	Určuje možnost rozšíření oddílu bez ohledu na stáří ovladačů
0x002C	Číslo clusteru začátku kořenového adresáře (většinou 2)
0x002D	
0x002E	
0x002F	
0x0030	Číslo sektoru, poskytující informace o souborovém systému
0x0031	
0x0032	Číslo sektoru pro záložní boot (běžně 6)
0x0033	
0x0034	Rezervováno (12 bajtů)
...	
0x003F	
0x0040	Číslo logické jednotky oddílu
0x0041	Nepoužito
0x0042	Rozšířená signatura (0x29) Označuje přítomnost dalších tří polí v boot sektoru
0x0043	Sériové číslo oddílu
0x0044	
0x0045	
0x0046	
0x0047	11bajtový název oddílu
...	
0x0051	
0x0052	8bajtový název FAT (FAT32)
...	
0x0059	
0x005A	
...	420 bajtů spustitelného kódu a dat
0x00FD	
0x00FE	Boot Signatura = 0xAA55 (pokud má sektor velikost 512 Bajtů)
0x00FF	

Tab. 2.6: Boot Record [4] [8]

2.4 FAT Tabulky

Tabulky FAT obsahují záznam pro každý cluster na oddílu. Každý z těchto záznamů má 32 bitů. Při zápisu souboru je vyhledán první volný cluster z FAT tabulky a uložen do záznamu souborových umístění. Soubor je pak zapsán do clusteru. Pokud se soubor do jednoho clusteru nevejde, je vyhledán další volný cluster v pořadí a jeho číslo je opět zapsáno do FAT záznamu. Tento proces pokračuje až k poslednímu clusteru, který je pro zápis daného souboru potřeba. Značka EOC (End Of Cluster-chain) je zapsána do FAT tabulky k poslednímu clusteru, aby indikovala, že žádné další clustery nejsou použity. Tudíž, při čtení souboru je počáteční číslo clusteru určeno ze souborového záznamu v adresáři, kde je soubor umístěn. Následně je použita FAT tabulka k nalezení dalšího clusteru, který uchovává další část dat daného souboru, pak dalšího clusteru a podobně. Zatímco EOC značka indikuje, že cluster je tím posledním, který uchovává soubor, tak přesná velikost souboru je uložena v záznamu souborových umístění, takže může být určeno číslo posledního použitého bajtu daného souboru. Systém FAT32 používá až 4 FAT tabulky. To umožňuje zálohu v případě poškození jedné z těchto tabulek. Každá další FAT tabulka přímo následuje tu předchozí. Doporučený počet tabulek je 2, kvůli starším systémům, které tuto hodnotu předpokládají. Nicméně počet FAT tabulek nemusí být 2 a pro flash disky, kde je záloha FAT redundantní, může být použita pouze jedna.

Bajt (Adresa začátku oddílu + 512 + #)		**FAT záznam**	**Hodnota**
0	0x0000	1	Rezervováno. Obsahuje typ media ve spodních 8 bitech. Ostatní bity jsou nastaveny na hodnotu 1.
1	0x0001		
2	0x0002		
3	0x0003		
4	0x0004	2	Rezervováno. Horní 2 bity slouží jako příznaky stavu ‘dirty volume’
5	0x0005		
6	0x0006		
7	0x0007		
8	0x0008	3	FAT záznam prvního clusteru datové oblasti oddílu
9	0x0009		
10	0x000A		
11	0x000B		
...	...		
#	0x####	#	FAT záznam posledního clusteru datové oblasti oddílu
#	0x####		
#	0x####		
#	0x####		

Tab. 2.7: FAT32 tabulka [4]

2.5 Kořenový adresář a jiné adresáře

Adresář FAT je prostý soubor, obsahující lineární seznam 32bajtových záznamů. Jediný speciální adresář, který musí být vždy přítomen, je kořenový adresář (Root Directory). Pro FAT32 může mít kořenový adresář proměnnou velikost a je tvořen řetězcem clusterů, stejně jako každý jiný soubor, či adresář. První cluster kořenového adresáře je specifikován v sekci Boot Record 2.3. Zašedlé řádky byly v původní DOS specifikaci nepoužity a nevyužité mohou být stále, pokud je požadováno [4].

<table>
<tr><th colspan="2">Bajt</th><th>Hodnota</th></tr>
<tr><td>0</td><td>0x00</td><td rowspan="3">Název
(Název souborů o 8 znacích)</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>7</td><td>0x07</td></tr>
<tr><td>8</td><td>0x08</td><td rowspan="3">Přípona
(Přípona souboru o 3 znacích)</td></tr>
<tr><td>9</td><td>0x09</td></tr>
<tr><td>10</td><td>0x0A</td></tr>
<tr><td>11</td><td>0x0B</td><td>Atributy
<table>
<tr><td>Bit:</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Hodnota:</td><td>0</td><td>0</td><td>Archiv</td><td>Adresář</td><td>Štítek oddílu</td><td>Systém</td><td>Skrytý</td><td>Pro čtení</td></tr>
</table></td></tr>
<tr><td>12</td><td>0x0C</td><td>NT (Rezervováno pro Windows NT, vždy 0)</td></tr>
<tr><td>13</td><td>0x0D</td><td>Čas vytvoření v ms (0 při nepoužití)</td></tr>
<tr><td>14</td><td>0x0E</td><td rowspan="2">Čas vytvoření - hodina a minuta (0 při nepoužití)</td></tr>
<tr><td>15</td><td>0x0F</td></tr>
<tr><td>16</td><td>0x10</td><td rowspan="2">Datum vytvoření (0 při nepoužití)</td></tr>
<tr><td>17</td><td>0x11</td></tr>
<tr><td>18</td><td>0x12</td><td rowspan="2">Datum posledního přístupu (0 při nepoužití)</td></tr>
<tr><td>19</td><td>0x13</td></tr>
<tr><td>20</td><td>0x14</td><td rowspan="2">2 horní bajty čísla prvního clusteru tohoto záznamu</td></tr>
<tr><td>21</td><td>0x15</td></tr>
<tr><td>22</td><td>0x16</td><td rowspan="2">Čas posledního zápisu do souboru</td></tr>
<tr><td>23</td><td>0x17</td></tr>
<tr><td>24</td><td>0x18</td><td rowspan="2">Datum posledního zápisu do souboru</td></tr>
<tr><td>25</td><td>0x19</td></tr>
<tr><td>26</td><td>0x1A</td><td rowspan="2">První cluster
(Odkázáno ze začátku datové oblasti svazku)</td></tr>
<tr><td>27</td><td>0x1B</td></tr>
<tr><td>28</td><td>0x1C</td><td rowspan="4">Velikost souboru</td></tr>
<tr><td>29</td><td>0x1D</td></tr>
<tr><td>30</td><td>0x1E</td></tr>
<tr><td>31</td><td>0x1F</td></tr>
</table>

Tab. 2.8: Struktura adresářového záznamu [4]

Pokud je první bajt adresáře roven 0xE5, pak to znamená, že byl záznam vymazán. Pokud je tento první bajt roven 0x00, pak nebyl záznam nikdy použit (toho může být

využito při hledání konce tabulky, jelikož i následující záznamy budou rovny 0x00). Datová oblast se nachází hned za kořenovým adresářem. Jediným rozdílem mezi kořenovou a každou jinou složkou je to, že kořenová složka má specifické umístění a pevně daný počet záznamů. Kořenový adresář se od běžných adresářů liší tím, že sám o sobě nemá žádné značky data, času ani vlastní název, kromě znaku „\“. Dalším jediným zvláštním aspektem kořenového adresáře je to, že se jedná o jediný adresář na FAT svazku, pro který je v pořádku, že má nastavený pouze bit atributu ID Svazku.

Formát data a času

Pokud není datum a čas podporován, tak je zapsán jako 0. Bajty 22–25 (čas posledního zápisu a datum posledního zápisu) musí být podporovány podle FAT specifikace, ale pokud zařízení nedisponuje vnitřními hodinami, pak není určení času možné.

Datum:

- Bity 15–9: Počet roků od 1980, platný rozsah je 0–127 (1980–2107)
- Bity 8–5: Měsíc v roce, platný rozsah je 1–12 (1 = Leden)
- Bity 4–0: Den v měsíci, platný rozsah je 1–31

Čas:

- Bity 15–11: Hodiny, platný rozsah je 0–23
- Bity 10–5: Minuty, platný rozsah je 0–59
- Bity 4–0: Sekundy, 2sekundový krok, platný rozsah je 0–29 (0–58 sekund)

2.6 Oblast dat

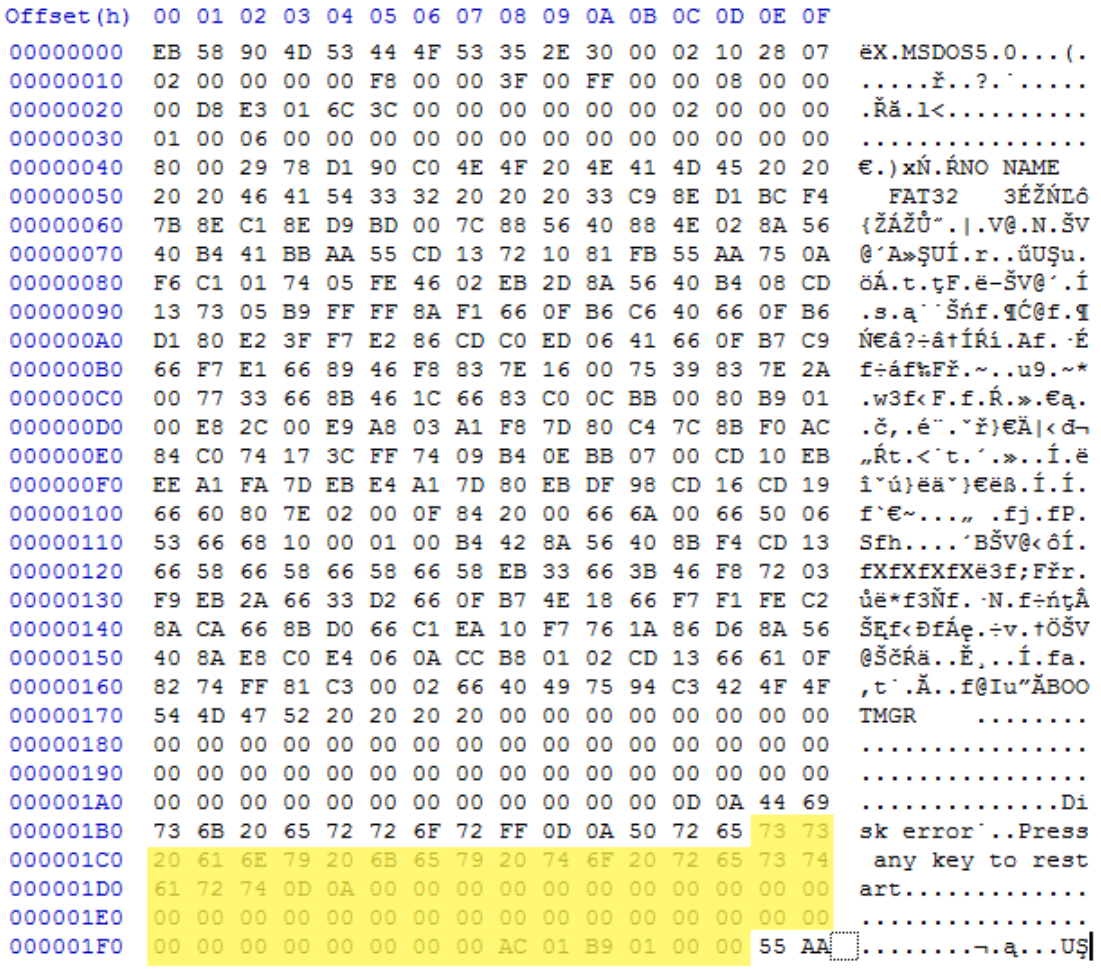
Zbytek svazku tvoří oblast dat, která obsahuje soubory a adresáře. Jedná se o oblast, na kterou je odkazováno ve FAT tabulkách. Pro FAT32 je počáteční adresa datové oblasti následující: Počáteční adresa oddílu + Počet rezervovaných sektorů + (Počet FAT tabulek + velikost FAT tabulky)

3 FUNKCE PROGRAMU

Následující kapitola se zabývá funkcí výsledného programu, které bloky Boot sectoru budou obnoveny, které nikoliv a proč.
Součástí zadání je návrh a praktická realizace programu schopného opravy poškozených flash disků, nebo paměťových karet. Původní koncept byl takový, že výsledný program bude přepisovat bajty na poškozeném zařízení na místech, kde mají být uloženy hodnoty přímo dané specifikací souborového systému FAT32. Zasahovat se mělo do sektorů, kde je uloženo MBR a Boot sector. Po prozkoumání těchto sektorů pomocí hexeditoru HxD [10] bylo nicméně zjištěno, že nultý sektor zařízení neodpovídá tomu, jak by měl dle FAT32 specifikace vypadat. Toto chování bylo zcela nezávislé na kapacitě, nebo výrobci flashdisku, stejně jako na použitém operačním systému pro formátování (vyzkoušeny byly systémy Windows 7, Windows 10 a Ubuntu). Příklad korektního nultého sektoru s MRB je na obrázku 3.1, zatímco mnou zobrazované nulté sektory vypadaly tak, jak je ukázáno na obrázku 3.2.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	33	C0	8E	D0	BC	00	7C	FB	50	07	50	1F	FC	BE	1B	7C	3ÀŽÐ¼.|ûP.P.ü¾.|
00000010	BF	1B	06	50	57	B9	E5	01	F3	A4	CB	BE	BE	07	B1	04	¿..PW¹å.ó¤Ë¾¾.±.
00000020	38	2C	7C	09	75	15	83	C6	10	E2	F5	CD	18	8B	14	8B	8,|.u.ƒÆ.âõÍ.‹.‹
00000030	EE	83	C6	10	49	74	16	38	2C	74	F6	BE	10	07	4E	AC	îƒÆ.It.8,tö¾..N¬
00000040	3C	00	74	FA	BB	07	00	B4	0E	CD	10	EB	F2	89	46	25	<.tú»..´.Í.ëò‰F%
00000050	96	8A	46	04	B4	06	3C	0E	74	11	B4	0B	3C	0C	74	05	–ŠF.´.<.t.´.<.t.
00000060	3A	C4	75	2B	40	C6	46	25	06	75	24	BB	AA	55	50	B4	:Äu+@ÆF%.u$»ªUP´
00000070	41	CD	13	58	72	16	81	FB	55	AA	75	10	F6	C1	01	74	AÍ.Xr..ûUªu.öÁ.t
00000080	0B	8A	E0	88	56	24	C7	06	A1	06	EB	1E	88	66	04	BF	.Šà^V$Ç.¡.ë.^f.¿
00000090	0A	00	B8	01	02	8B	DC	33	C9	83	FF	05	7F	03	8B	4E	..¸...‹Ü3Éƒÿ...‹N
000000A0	25	03	4E	02	CD	13	72	29	BE	46	07	81	3E	FE	7D	55	%.N.Í.r)¾F..>þ}U
000000B0	AA	74	5A	83	EF	05	7F	DA	85	F6	75	83	BE	27	07	EB	ªtZƒï..Ú…öuƒ¾'.ë
000000C0	8A	98	91	52	99	03	46	08	13	56	0A	E8	12	00	5A	EB	Š˜'R™.F..V.è..Zë
000000D0	D5	4F	74	E4	33	C0	CD	13	EB	B8	00	00	00	00	00	00	ÕOtä3ÀÍ.ë¸.....
000000E0	56	33	F6	56	56	52	50	06	53	51	BE	10	00	56	8B	F4	V3öVVRP.SQ¾..V‹ô
000000F0	50	52	B8	00	42	8A	56	24	CD	13	5A	58	8D	64	10	72	PR¸.BŠV$Í.ZX.d.r
00000100	0A	40	75	01	42	80	C7	02	E2	F7	F8	5E	C3	EB	74	49	.@u.B€Ç.â÷ø^ÃëtI
00000110	6E	76	61	6C	69	64	20	70	61	72	74	69	74	69	6F	6E	nvalid partition
00000120	20	74	61	62	6C	65	00	45	72	72	6F	72	20	6C	6F	61	table.Error loa
00000130	64	69	6E	67	20	6F	70	65	72	61	74	69	6E	67	20	73	ding operating s
00000140	79	73	74	65	6D	00	4D	69	73	73	69	6E	67	20	6F	70	ystem.Missing op
00000150	65	72	61	74	69	6E	67	20	73	79	73	74	65	6D	00	00	erating system..
00000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000180	00	00	00	8B	FC	1E	57	8B	F5	CB	00	00	00	00	00	00	...‹ü.W‹õË.....
00000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001B0	00	00	00	00	00	00	00	00	51	83	59	E3	00	00	80	01	........QƒYã..€.
000001C0	01	00	07	FE	E0	FF	20	00	00	00	A0	AC	8A	08	00	00	...þàÿ ... ¬Š...
000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA	.....Uª

Obr. 3.1: Korektní nultý sektor s MBR [11]



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   EB 58 90 4D 53 44 4F 53 35 2E 30 00 02 10 28 07   ëX.MSDOS5.0...(.
00000010   02 00 00 00 00 F8 00 00 3F 00 FF 00 00 08 00 00   .....ř..?.˙.....
00000020   00 D8 E3 01 6C 3C 00 00 00 00 00 00 02 00 00 00   .Řă.l<..........
00000030   01 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000040   80 00 29 78 D1 90 C0 4E 4F 20 4E 41 4D 45 20 20   €.)xŃ.ŔNO NAME
00000050   20 20 46 41 54 33 32 20 20 20 33 C9 8E D1 BC F4     FAT32   3ÉŽŃLô
00000060   7B 8E C1 8E D9 BD 00 7C 88 56 40 88 4E 02 8A 56   {ŽÁŽŮ˝.|.V@.N.ŠV
00000070   40 B4 41 BB AA 55 CD 13 72 10 81 FB 55 AA 75 0A   @´A»ŞUÍ.r..űUŞu.
00000080   F6 C1 01 74 05 FE 46 02 EB 2D 8A 56 40 B4 08 CD   öÁ.t.ţF.ë-ŠV@´.Í
00000090   13 73 05 B9 FF FF 8A F1 66 0F B6 C6 40 66 0F B6   .s.ą˙˙Šńf.¶Ć@f.¶
000000A0   D1 80 E2 3F F7 E2 86 CD C0 ED 06 41 66 0F B7 C9   Ń€â?÷â†ÍŔí.Af.·É
000000B0   66 F7 E1 66 89 46 F8 83 7E 16 00 75 39 83 7E 2A   f÷áf‰Fř.~..u9.~*
000000C0   00 77 33 66 8B 46 1C 66 83 C0 0C BB 00 80 B9 01   .w3f‹F.f.Ŕ.».€ą.
000000D0   00 E8 2C 00 E9 A8 03 A1 F8 7D 80 C4 7C 8B F0 AC   .č,.é¨.˘ř}€Ä|‹đ¬
000000E0   84 C0 74 17 3C FF 74 09 B4 0E BB 07 00 CD 10 EB   „Ŕt.<˙t.´.»..Í.ë
000000F0   EE A1 FA 7D EB E4 A1 7D 80 EB DF 98 CD 16 CD 19   î˘ú}ëä˘}€ëß.Í.Í.
00000100   66 60 80 7E 02 00 0F 84 20 00 66 6A 00 66 50 06   f`€~...„ .fj.fP.
00000110   53 66 68 10 00 01 00 B4 42 8A 56 40 8B F4 CD 13   Sfh....´BŠV@‹ôÍ.
00000120   66 58 66 58 66 58 66 58 EB 33 66 3B 46 F8 72 03   fXfXfXfXë3f;Fřr.
00000130   F9 EB 2A 66 33 D2 66 0F B7 4E 18 66 F7 F1 FE C2   ůë*f3Ňf.·N.f÷ńţÂ
00000140   8A CA 66 8B D0 66 C1 EA 10 F7 76 1A 86 D6 8A 56   ŠĘf‹ĐfÁę.÷v.†ÖŠV
00000150   40 8A E8 C0 E4 06 0A CC B8 01 02 CD 13 66 61 0F   @ŠčŔä..Ě¸..Í.fa.
00000160   82 74 FF 81 C3 00 02 66 40 49 75 94 C3 42 4F 4F   ‚t˙.Ă..f@Iu”ĂBOO
00000170   54 4D 47 52 20 20 20 20 00 00 00 00 00 00 00 00   TMGR    ........
00000180   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000190   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001A0   00 00 00 00 00 00 00 00 00 00 00 00 0D 0A 44 69   ..............Di
000001B0   73 6B 20 65 72 72 6F 72 FF 0D 0A 50 72 65 73 73   sk error˙..Press
000001C0   20 61 6E 79 20 6B 65 79 20 74 6F 20 72 65 73 74    any key to rest
000001D0   61 72 74 0D 0A 00 00 00 00 00 00 00 00 00 00 00   art.............
000001E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001F0   00 00 00 00 00 00 00 00 AC 01 B9 01 00 00 55 AA   ........¬.ą...UŞ
```

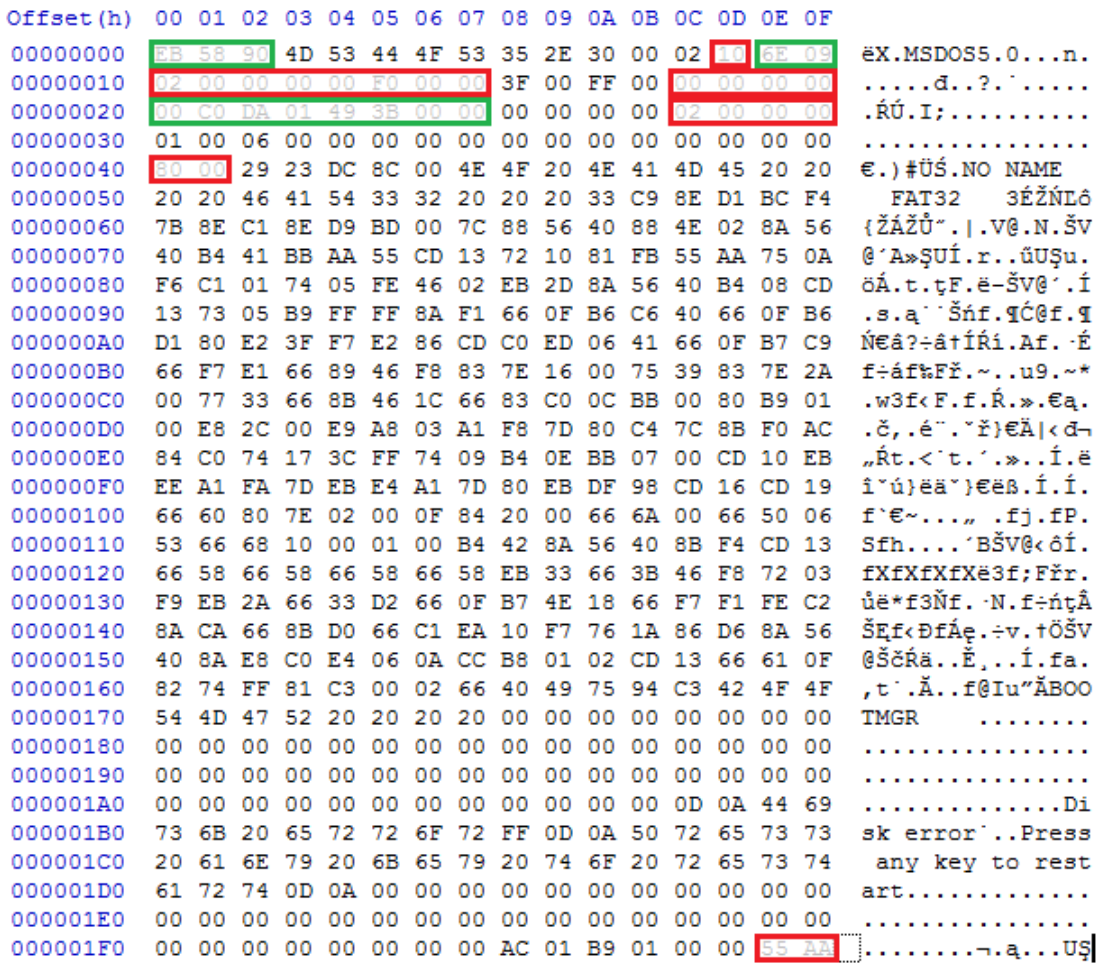
Obr. 3.2: Nultý sektor bez MBR

Vzhledem k tomu, že prvních 446 bajtů MBR tvoří spustitelný kód, který nemá jednoduše určenou strukturu podle specifikace (modře zvýrazněná oblast na obrázku 3.1), nešlo by s jistotou určit, že některé bajty nabývají nesprávných hodnot. Porovnání je tedy potřeba provádět u bajtů, které mají mít dle specifikace jasně danou hodnotu, konkrétně se pak jedná o tabulku oddílů (na obrázcích žlutě zvýrazněná oblast). Tato oblast obsahuje čtyři šestnáctibajtové záznamy oddílů, které musí začínat hodnotami buďto 80_H, nebo 00_H, což nebylo dodrženo. Po důkladnějším prozkoumání tohoto nultého sektoru se ukázalo, že se ve skutečnosti jedná o Boot sector, který se běžně nachází až v prvním sektoru disku. Tento fakt potvrzuje i informace Microsoftu, která uvádí, že floppy disky (diskety) MBR vůbec nemají a nultý sektor takovýchto disků je obsazen Boot sectorem [12]. Flash disky a paměťové karty jsou tedy operačními systémy vnímány jako diskety, MBR se na nich nenachází a případné pokusy o jeho obnovu jsou zcela bezpředmětné. Obnova případně poškozených bajtů se tedy týká pouze Boot sectoru. Zejména za pomoci dat z tabulky 2.3 a informací přímo ze stránek Microsoftu [12] bylo určeno, které bajty

musí nabývat přesně specifikovaných hodnot. Mimo výše zmíněné zdroje informací bylo využito metody pokus-omyl, kdy byly postupně manuálně odmazávány (nulovány) celé sekce v Boot sectoru, aby bylo zjištěno, bez jakých metadat je flash disk, případně paměťová karta stále funkční. Touto metodou bylo zjištěno, že se dají data v Boot sectoru rozdělit do tří základních skupin:

- Nepotřebná data
- Potřebná neobnovitelná data
- Potřebná obnovitelná data

Dobře viditelné je toto rozdělení na obrázku 3.3, kde zeleně ohraničené oblasti jsou potřebná neobnovitelná data, červeně ohraničené oblasti jsou potřebná obnovitelná data a zbytek jsou data nepotřebná.



Obr. 3.3: Zobrazení kategorií metadat v Boot sectoru

3.1 Nepotřebná data

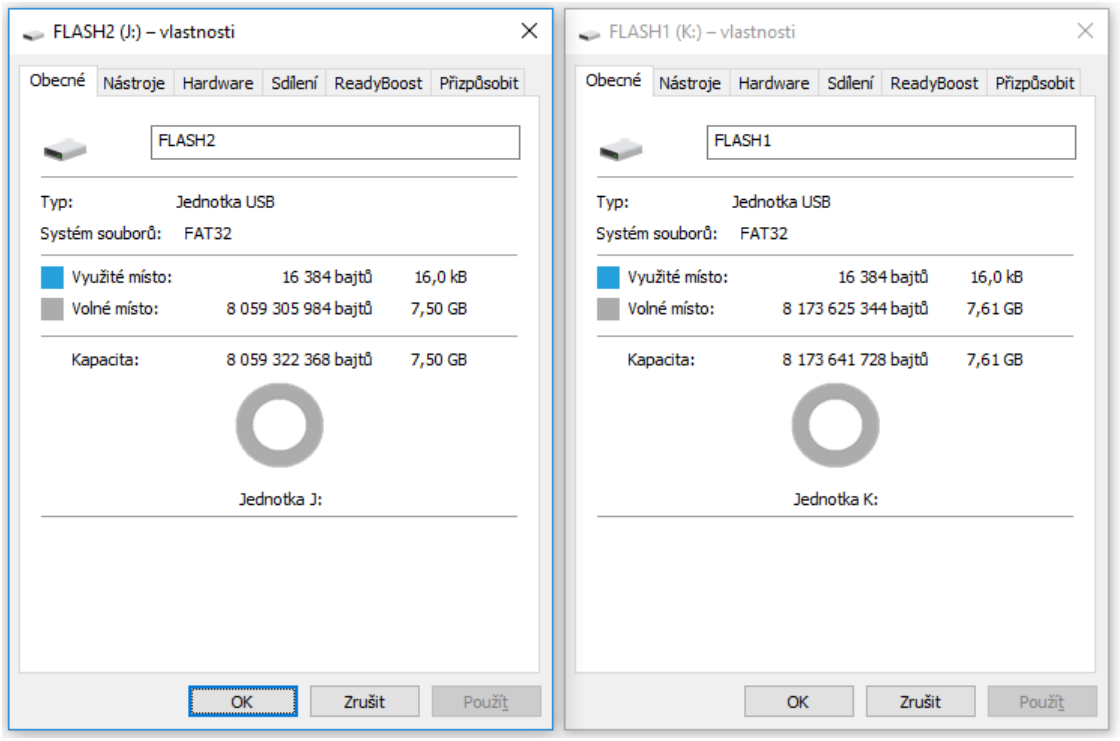
Nepotřebná data jsou taková, jejichž přítomnost, ani hodnota nemá vliv na správnou funkci flash disku, nebo paměťové karty. Pokud by při poškození media došlo i k poškození těchto dat, neměla by tedy jejich obnova žádný smysl. Výsledný program tedy na tato metadata nebere zřetel.

3.2 Potřebná neobnovitelná data

Potřebná neobnovitelná data jsou taková, která jsou pro správnou funkci daného media potřebná, nicméně jejich obnova není snadná z níže popsaných důvodů. Jedná se o následující sekce:

- Bajty 00_H, 01_H, 02_H (skoková instrukce k boot kódu)
- Bajty $0E_H$, $0F_H$ (rezervované sektory)
- Bajty 20_H, 21_H, 22_H, 23_H (počet sektorů v oddílu)
- Bajty 24_H, 25_H, 26_H, 27_H (počet sektorů ve FAT)

Skoková instrukce k boot kódu se sice nedá obnovit podle informací dle oficiální FAT32 specifikace, nicméně jsou tyto tři bajty, alespoň pro mou testovací skupinu disků, vždy stejné a je tedy možné tyto bajty obnovit. Tato sekce tří bajtů je umístěna mezi neobnovitelnými daty kvůli tomu, že obnova neprobíhá na základě specifikace, nýbrž na základě určité statistiky a bylo by tedy možné najít disky, kde by tyto bajty měly rozdílné hodnoty a obnova by se následně nepodařila.
Rezervované sektory je dvojice bajtů, která má podle specifikace typickou decimální hodnotu 32. Po zápisu této hodnoty na danou pozici se ale disk stává nečitelným, a proto i navzdory specifikaci Microsoftu nemá žádný smysl tyto bajty obnovovat [12].
Počet sektorů v oddílu a **počet sektorů ve FAT** jsou hodnoty přímo závislé na velikosti daného media. Tyto hodnoty se liší dokonce i pro disky, které mají stejnou kapacitu v gigabajtech. Z obrázku 3.4 je dobře patrné, že i přes to, že mají oba disky kapacitu 8 GB, tak se počet bajtů liší o 144303360. Z tohoto důvodu tedy není možné tyto bajty v Boot sectoru obnovovat, protože není možné určit, kolik bajtů, potažmo sektorů, dané medium obsahuje.



Obr. 3.4: Porovnání dvou různých 8GB flashdisků

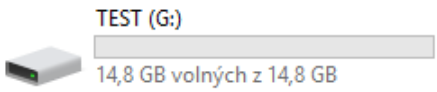
3.3 Potřebná obnovitelná data

Tato data jsou, jak už název napovídá, potřebná ke korektní funkci flash disku, případně paměťové karty. Jsou to data, která vytvořený program obnovuje. Objevují se zde i data, která nezpůsobují úplnou dysfunkci media (podrobněji popsáno níže), nicméně ve FAT32 specifikaci jsou konkrétně vyjádřeny hodnoty na těchto bajtových pozicích, a proto je program obnovuje rovněž. Výsledný efekt v závislosti na absenci, nebo změně hodnot těchto dat by se dal rozdělit do pěti skupin:

- Nefunkčnost - chyba č.1
- Nefunkčnost - chyba č.2
- Funkčnost - změna velikosti disku
- Funkčnost - změna názvu disku
- Funkčnost - žádná změna

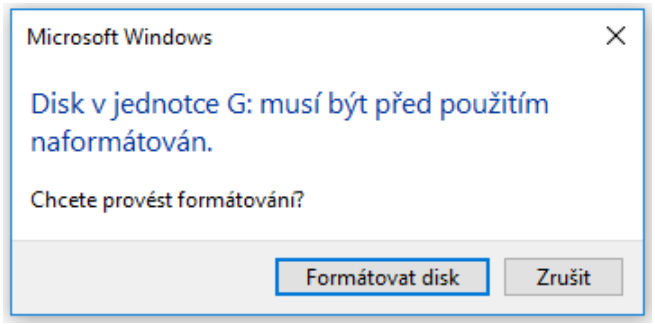
Nefunkčnost - chyba č.1 je způsobena změnou hodnoty bajtů na několika pozicích. První z těchto pozic je $0D_H$, která uchovává počet sektorů na cluster v mocninách dvou. Správná hodnota je závislá na kapacitě media, kdy pro disky o kapacitě menší než 16 GB je hodnota 8_D, pro kapacitu 16 GB je hodnota 16_D a pro kapacitu 32 GB je hodnota 32_D. Oddíly o vyšší kapacitě FAT32 oficiálně nepodporuje 2.1. Další pozice je 15_H, což představuje popis media ($F0_H$ pro vyměnitelná media). Poslední pozice jsou 16_H a 17_H. Tyto pozice určují počet sektorů na FAT. Nicméně FAT32 tato

data nepoužívá a dle specifikace jsou vždy 00_H. Při přepsání těchto pozic hodnotou jinou, než výše popsanou, se disk stává nepřístupným 3.6 a při pokusu o otevření obsahu se zobrazuje okno s chybovou hláškou 3.7.



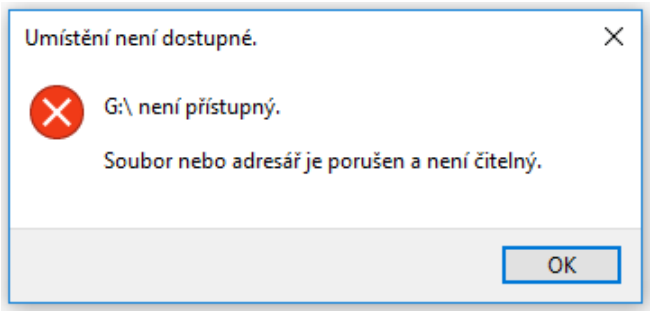
Obr. 3.5: Korektně fungující disk

Obr. 3.6: Nepřístupný disk při chybě v Boot sectoru



Obr. 3.7: Okno s chybovou hláškou č.1

Nefunkčnost - chyba č.2 je opět způsobena změnou hodnoty bajtů na několika pozicích. Jedná se o skupinu bajtů $2C_H$, $2D_H$, $2E_H$, $2F_H$. Tyto bajty představují číslo clusteru začátku kořenového adresáře. Jejich hodnota je 02000000_H, kořenový adresář tedy začíná v druhém clusteru. Při přepsání těchto bajtů jinou hodnotou se disk opět stává nepřístupným 3.6 a při pokusu o otevření obsahu disku se rovněž zobrazuje okno s chybovou hláškou, tentokrát jinou, než v předchozím případě 3.7. **Funkčnost - změna velikosti disku**. Při určité změně bajtů 13_H, 14_H z hodnot 0000_H na jinou se změní velikost využitelné kapacity disku. Například při změně těchto bajtů na $FFFF_H$ se změní kapacita ze 14,8 GB na 15,9 MB, nebo při změně na $AAAA_H$ se kapacita změní na 5,32 MB. Změna kapacity nicméně není pravidlem. Pokud se výše zmíněné bajty přepíší například na 1111_H, tak nedojde k žádné změně. Tyto bajty představují počet sektorů pro oddíly menší než 32MB. Souvislost mezi tímto a kapacitou disku tedy není zcela zřejmá. Souborový systém FAT32 tato data nepoužívá a dle specifikace jsou vždy 0000_H.

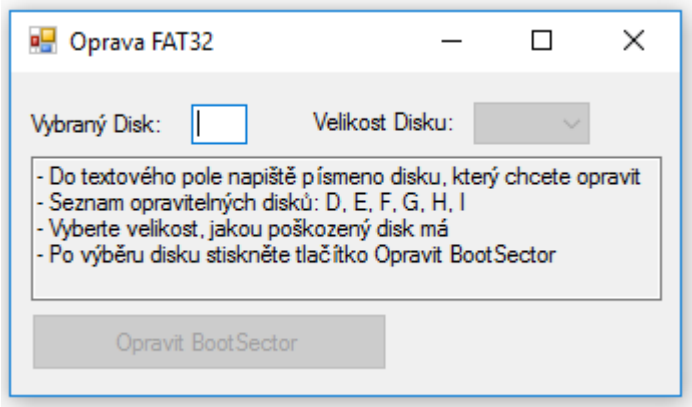


Obr. 3.8: Okno s chybovou hláškou č.2

Funkčnost - změna názvu disku. Modifikace bajtů na pozicích 10_H, 11_H, 12_H způsobuje změnu názvu disku na Beze jména, při zachování jeho funkce. Bajt 10_H vyjadřuje počet kopií FAT a jeho hodnota má být 02_H. Další dva výše zmíněné bajty reprezentují maximum zápisů kořenového adresáře a hodnota těchto bajtů má být 0000_H. Opět se jedná o dvojici bajtů, které FAT32 nepoužívá.
Funkčnost - žádná změna. Bajty, které spadají do této skupiny jsou takové, jejichž modifikace není nezbytná pro správnou funkci disku. První skupinou jsou bajty $1C_H$, $1D_H$, $1E_H$, $1F_H$, jejichž hodnota podle specifikace má být 00000000_H. Tato skupina představuje počet skrytých sektorů v oddílu. Další je bajt 41_H, s hodnotou 00_H. Tento bajt není ve FAT32 vůbec používán. Dalším bajtem je pak 42_H. Hodnota tohoto bajtu má být 29_H a ve specifikaci se píše jen, že se jedná o rozšířenou signaturu, která označuje přítomnost následujících tří bajtů. Poslední dvojice bajtů je na pozicích FE_H, FF_H. Společně tvoří bootovací signaturu, která musí mít dle specifikace podobu $AA55_H$.

4 POPIS PROGRAMU

Tato kapitola se zabývá popisem tvorby výsledného programu. Celá aplikace byla napsána v programovacím jazyku C++ s využitím grafického uživatelského rozhraní (GUI) Visual C++. Volba tohoto jazyka padla zejména proto, že obsahuje funkce, které jsou vytvořené přímo pro přístup k diskům. Koncept programu je směřován k jednoduchosti a uživatelské přívětivosti. Výsledné okno programu je zobrazeno na obrázku 4.1. Ve stručnosti je funkce aplikace taková, že uživatel musí napsat písmeno disku z níže nabízených k opravě. Je programově ošetřeno, že nezáleží, zda jsou zadány kapitálky či ne. Po té, co je zadán disk, aktivuje se rozvíjecí seznam, kde uživatel vybere kapacitu opravovaného disku. Jakmile jsou tyto parametry vyplněny, je aktivováno tlačítko, které opraví poškozený Boot sector vybraného disku. Funkci programu doplňují různá informační okna. Pro uživatele je přímo na okně programu připraven jednoduchý návod, pro správné provedení všech potřebných kroků. Veškerá funkcionalita popsaná v tomto odstavci je blíže vysvětlena dále v této kapitole.



Obr. 4.1: Okno hotového programu

Základem celého programu je funkce CreateFile [13].

Ukázka kódu 4.1: Použití funkce CreateFile

```
device = CreateFile(L"\\\\.\\D:",
                    GENERIC_READ | GENERIC_WRITE,
                    FILE_SHARE_READ | FILE_SHARE_WRITE,
                    NULL,
                    OPEN_EXISTING,
                    0,
                    NULL);
```

Proměnná `device` je typu handle, díky které program pracuje s obsahem disku. V tomto konkrétním případě se jedná o disk `D:`. Nepodařilo se mi bohužel do funkce `CreateFile` zakomponovat jakoukoliv proměnnou, která by v sobě měla uloženo písmeno disku k opravě, a proto je výběr disku k obsluze proveden pomocí příkazu `switch`. Toto řešení není příliš elegantní, nicméně funkční. Příkaz `switch` omezuje výběr disku jen na šest disků s písmeny D–I. Atributy funkce `CreatFile` mimo specifikaci disku jsou následující:

- **GENERIC_READ | GENERIC_WRITE:** Určuje požadovaný přístup k disku. Může být pouze pro čtení, pro zápis, kombinace obojího, jako v tomto případě, nebo zcela bez přístupu.
- **FILE_SHARE_READ | FILE_SHARE_WRITE:** Určuje požadované sdílení disku. Může být pro čtení, zápis, kombinace obojího, jako v tomto případě, mazání, všechny předešlé, nebo bez sdílení. Bez povolení sdílení je všem požadavkům ostatních procesů zakázáno disk číst, mazat, či přepisovat.
- **NULL:** Ukazatel na strukturu `SECURITY_ATTRIBUTES` obsahující dvě oddělené datové jednotky (`lpSecurityDescriptor` a `bInheritHandle`). Tento parametr může být nastaven na `NULL` s tím, že handle disku vytvořený funkcí `CreateFile` nemůže být zděděn žádným potomkem, který by aplikace mohla vytvořit a disk asociovaný s tímto handlem získá výchozí security descriptor.
- **OPEN_EXISTING:** Určuje akci, která se má se zařízením provést. Pro zařízení jiné, než soubory (například disky, jako v případě tohoto programu) má tento parametr běžně hodnotu `OPEN_EXISTING`.
- **0:** Tato hodnota určuje parametry, které má vytvářený soubor mít. Pokud je otevírán existující soubor, což v tomto případě platí, jsou atributy přebírány z tohoto souboru (disku).
- **NULL:** Handle souboru šablony s `GENERIC_READ` přístupovými právy. Šablona dodává vytvářenému souboru atributy a rozšířené atributy. Pokud se otevírá existující soubor, je tento parametr ignorován, a proto je jeho hodnota nastavena na `NULL`.

Po vytvoření handlu disku je provedeno jeho načtení do paměti spolu se základním ověřením správné funkce dosavadního kódu.

Ukázka kódu 4.2: Načtení sektoru do paměti a ověření funkce

```
SetFilePointer(device, numSector * 512, NULL, FILE_BEGIN);
ErrorFlag = ReadFile(device, sector, 512, &bytesRead, NULL);
if (FALSE == ErrorFlag)
 {
 MessageBox::Show("Disk se v~pocitaci nenachazi", "Chyba!");
 }
```

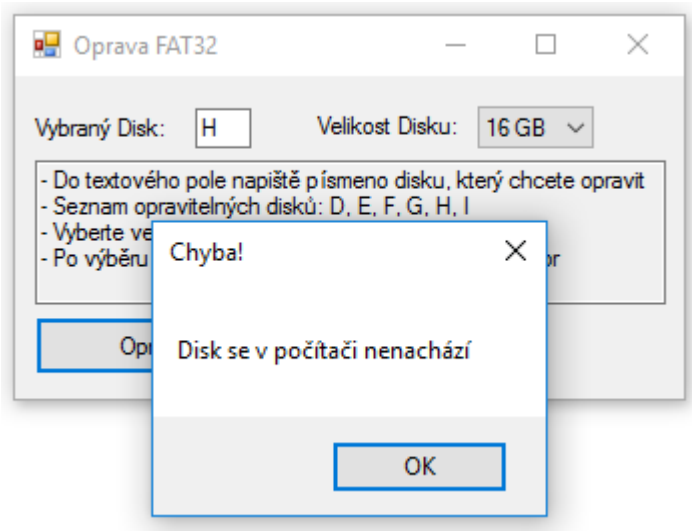
SetFilePointer je funkce, která uchovává ukazatel části disku, která se načte do paměti. Tohle se dá ovlivnit pomocí těchto parametrů [14]:

- **device:** Určuje handle disku, jehož ukazatel je uchováván.
- **numSector * 512:** Označuje počet bajtů, o které se ukazatel posune. V tomto případě jsou to celočíslené násobky 512 (`numSector` je proměnná typu integer, ve výchozím stavu nastavena na hodnotu 0), jelikož každý sektor FAT32 má právě 512 bajtů. Tak je dosaženo toho, že se ukazatel posune vždy na začátek sektoru, což zjednodušuje následné načítání do paměti.
- **NULL:** Rozšiřující parametr posunu ukazatele, v tomto případě není potřeba.
- **FILE_BEGIN:** Počáteční bod posunu ukazatele.

Funkce `ReadFile` pak slouží k načtení potřebných dat do paměti. Výstup této operace se dá opět ovlivnit několika parametry [15]:

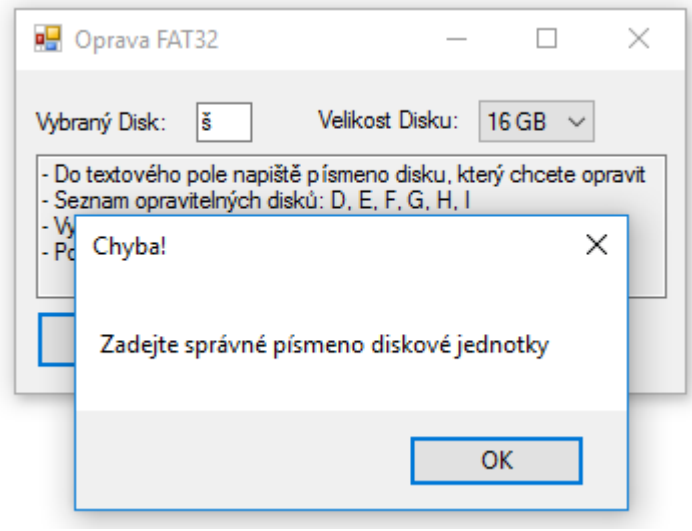
- **device:** Handle disku, ze kterého se data do paměti ukládají.
- **sector:** Místo v paměti, kam se data z disku ukládají. V tomto případě se jedná o pole typu byte o 512 prvcích.
- **512:** Počet bajtů, které se z disku do pole `sector` uloží. Počet 512 je zvolen kvůli velikosti sektoru ve FAT32, který má právě 512 bajtů. Tímto je zaručeno, že se v paměti nachází vždy přesně jeden sektor a není zasahováno do sektorů okolních. Boot sector, který je tímto programem editován obývá rovněž právě jeden sektor, konkrétně nultý.
- **&bytesRead:** Ukazatel na proměnnou, ve které je uloženo, kolik bajtů bylo ve skutečnosti načteno.
- **NULL:** Ukazatel na strukturu `OVERLAPPED`. Tento parametr je zapotřebí pouze pokud je handle disku otevřen s příznakem `FILE_FLAG_OVERLAPPED`. V opačném případě nabývá hodnoty `NULL`, jako tady.

Návratová hodnota funkce `ReadFile` je uložena do proměnné `ErrorFlag` a pomocí jednoduché podmínky je testováno, zda-li došlo k úspěšnému načtení dat do paměti



Obr. 4.2: Okno s první chybovou hláškou

(`ReadFile` při úspěšném čtení vrací hodnotu `true`). V tomto konkrétním případě je testováno, zda se zadaný disk skutečně nachází připojen k počítači, na kterém je opravný program spouštěn. Okno s chybovou hláškou je vidět na obrázku 4.2.
Jak již bylo zmíněno výše, výběr disku k opravě je prováděn skrze příkaz `switch`. Jednotlivé `case` sekce tohoto větvení jsou obsazeny funkcemi `CreateFile` pro všechny opravitelné disky. Výchozí sekce tohoto větvení `default` slouží pro ošetření případu, kdy uživatel zadá do textového pole jiný znak, než jsou podporované písmenka disků. (Tohle se nevztahuje na to, zda je disk zadán velkým, či malým písmenem). V takovém případě se otevře jiné okno s chybovou hláškou, které je zobrazeno na obrázku4.3.



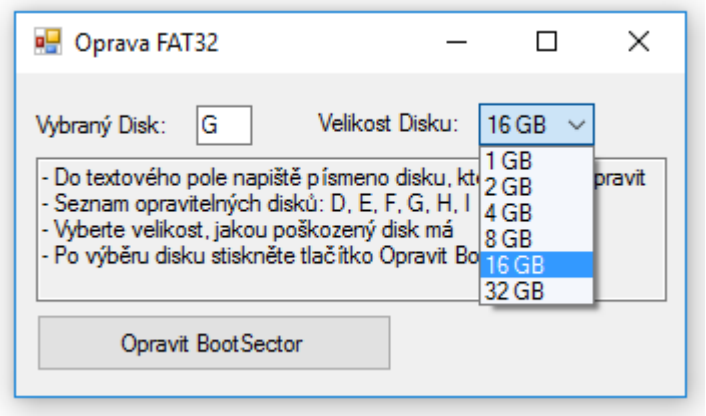
Obr. 4.3: Okno s druhou chybovou hláškou

Po opětovném volání funkce `SetFilePointer` (zapisování bajtů musí probíhat opět od začátku paměťového prostoru) jsou do bajtového pole `sector[512]` ukládány bajty popsány v předešlé kapitole 3 prostým přiřazováním správných hodnot na dané pozice v poli. Jediné větvení této procedury je v případě bajtu, který určuje kapacitu opravovaného disku. K tomu slouží rozvíjecí seznam (comboBox), který je obsluhován podmínkou 4.3. Tato podmínka se řídí specifikací dle 2.1.

Ukázka kódu 4.3: Větvení zápisu kapacity disku

```
if (kapacita == "16 GB")
{sector[13] = 16;}
else if (kapacita == "32 GB")
{sector[13] = 32;}
else
{sector[13] = 8;}
```

Výběr správné kapacity disku je pro uživatele realizován výše zmíněným rozvíjecím seznamem, jak je vidět na obrázku 4.4



Obr. 4.4: Zobrazení volby kapacity disku

Po zapsání správných hodnot dat bajtů do mezipaměti v podobě pole `sector` se opět zavolá funkce `SetFilePointer` ze stejného důvodu jako v předešlém případě a celý obsah pole `sector` se zapíše do nultého sektoru vybraného disku, tedy Boot sectoru. Funkce `WriteFile` [16] je v podstatě komplementární funkcí `ReadFile`, a proto jsou i jejich parametry takřka totožné:

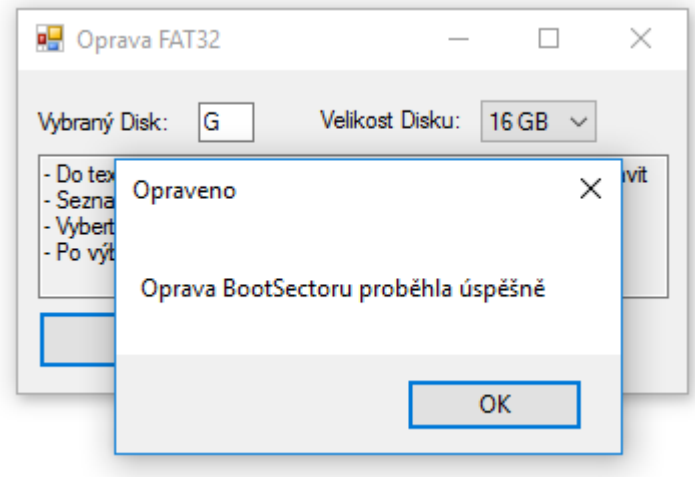
- **device:** Handle disku, na který se data z paměti ukládají.
- **sector:** Místo v paměti, odkud se data na disk ukládají. V tomto případě se jedná o pole `sector`

- **512:** Počet bajtů, které se na disk z pole `sector` uloží. Počet 512 je zvolen kvůli velikosti sektoru ve FAT32, který má právě 512 bajtů. Tímto je zaručeno, že se se na disk zapíše vždy přesně jeden sektor a není zasahováno do sektorů okolních.
- **&bytesWrite:** Ukazatel na proměnnou, ve které je uloženo, kolik bajtů bylo ve skutečnosti zapsáno.
- **NULL:** Ukazatel na strukturu `OVERLAPPED`. Tento parametr je zapotřebí pouze pokud je handle disku otevřen s příznakem `FILE_FLAG_OVERLAPPED`. V opačném případě nabývá hodnoty `NULL`, jako tady.

Ukázka kódu 4.4: Zápis dat na disk a ověření funkce

```
ErrorFlag2=WriteFile(device,sector,512,&bytesWrite,NULL);
if (TRUE == ErrorFlag2)
{
MessageBox::Show("Oprava byla uspesna", "Opraveno");
}
CloseHandle(device);
}
```

Návratová hodnota funkce WriteFile je uložena do proměnné ErrorFlag2 a pomocí jednoduché podmínky je testováno, zda-li došlo k úspěšnému zapsání dat z paměti na disk (WriteFile při úspěšném zápisu vrací hodnotu true). Při úspěšném zápisu dat na disk se zobrazí okno s informační hláškou, jak je to ukázáno na obrázku 4.5



Obr. 4.5: Zobrazení úspěšného zápisu na disk

Poslední zde zobrazené části kódu se starají o to, aby bylo použití programu odolné vůči chybám při zadávání vstupních parametrů (písmenko disku a kapacita). Funkce výsledného programu je taková, že jediný aktivní ovládací prvek v okně programu

je textové pole, do kterého se vpisuje písmeno disku k opravě. Jakmile je písmeno zadáno, zpřístupní se volba kapacity vybraného disku. Toho je docíleno procedurou 4.5.

Ukázka kódu 4.5: Podmínka pro aktivaci comboBoxu

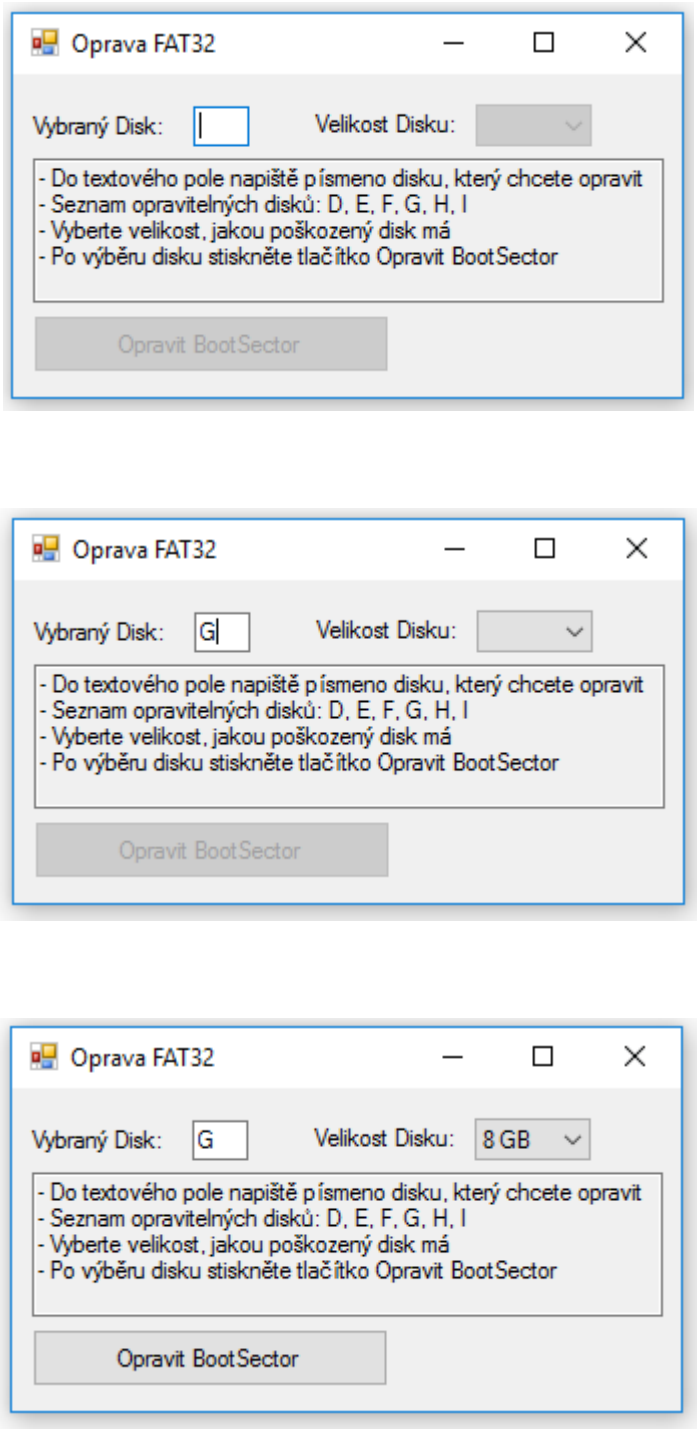
```
if (textBox1->Text != "")
        comboBox1->Enabled = true;
else comboBox1->Enabled = false;
```

V tomto stavu je tlačítko pro opravu disku stále neaktivní. Pro jeho aktivaci je zapotřebí vybrat požadovanou kapacitu z rozvíjecího seznamu. Jakmile je kapacita vybrána, aktivuje se tlačítko, které spouští opravnou sekvenci. Tohoto chování je dosaženo pomocí procedury 4.6.

Ukázka kódu 4.6: Podmínka pro aktivaci tlačítka

```
if (comboBox1->SelectedItem != NULL)
        button1->Enabled = true;
else button1->Enabled = false;
```

Tak je zajištěno, že program nezůstane viset v neznámém stavu, kdy nejsou určeny potřebné parametry disku. Tato funkcionalita je dobře vidět na porovnání 4.6.
Program byl vyzkoušen s několika flash disky a paměťovými kartami od různých výrobců o kapacitách v rozsahu 2 GB–16 GB. Ve všech případech se podařilo metadata Boot sectoru přepsat a obnovit tak správnou funkci media.



Obr. 4.6: Závislost zadaných parametrů na funkci programu

5 NÁVOD K PROGRAMU

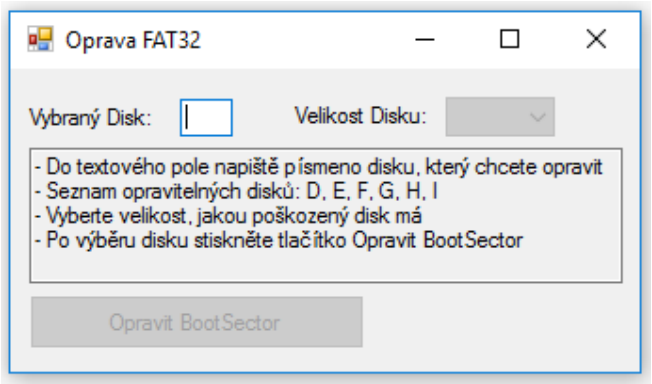
Následující kapitola se věnuje krátkému návodu k obsluze vytvořeného programu, jeho korektní funkci a popisu stavů, které mohou nastat, při nedodržení doporučeného použití.

5.1 Správná funkce programu

Tato sekce návodu slouží pro popis chování programu, pokud jsou dodrženy všechny zásady pro jeho správné používání. Případy, kdy nejsou tyto postupy přesně dodrženy, jsou popsány v další sekci.

5.1.1 Spuštění

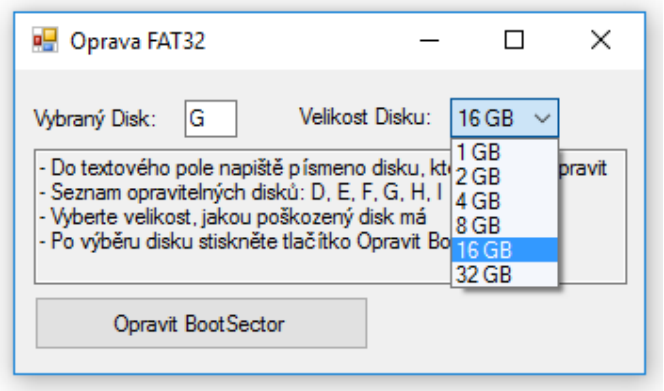
Po spuštění programu s názvem BootSector Oprava se zobrazí okno aplikace stejné, jako na obrázku 5.1. Samotné okno obsahuje jednoduché instrukce, podle kterých je potřeba se řídit.



Obr. 5.1: Okno spuštěné aplikace

5.1.2 Zadání parametrů media

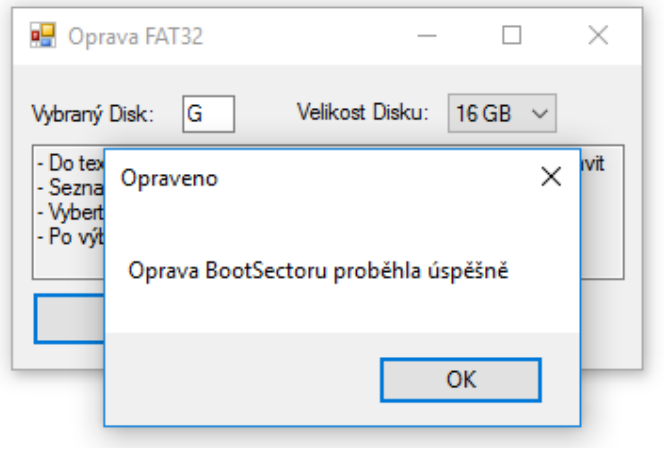
Následně je zapotřebí zadat validní informace o disku, který má být opravován. Tlačítko pro obnovu Boot sectoru nebude aktivní do doby, dokud nejsou oba parametry zadány. Písmeno jednotky je zadáváno ručně do textového pole. Seznam použitelných disků je: D, E, F, G, H, I. Nezáleží na tom, zda je disk zadán velkým, či malým písmenem. Po určení písmena disku je uživatel nucen vybrat kapacitu, kterou daný disk má. K tomuto účelu má program implementován rozvíjející seznam, viz obrázek 5.2



Obr. 5.2: Zadání parametrů disku

5.1.3 Obnova Boot sectoru

Po zadání parametrů se zpřístupní tlačítko pro opravu poškozeného disku. Po úspěšně provedené opravě se zobrazí okno s informační hláškou o úspěšné akci 5.3. Poškozený disk by nyní měl být opět čitelný a data, která se na něm nacházela před poškozením, opět přístupná. Pokud tomu tak není, bylo poškození rozsáhlejší, než jaké dokáže tento program opravit a z principu funkce nemá smysl pokoušet se o opravu tímto programem několikanásobně.



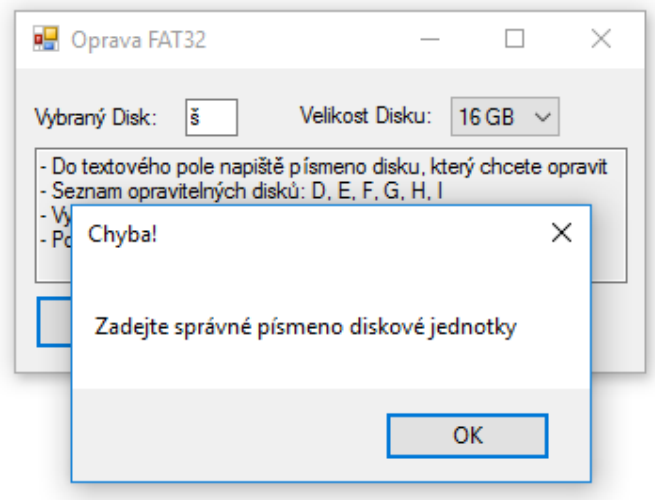
Obr. 5.3: Okno úspěšné obnovy

5.2 Chybové stavy programu

Tato sekce popisuje stavy programu, které nastanou při špatně zadaných vstupních parametrech. Při každé z těchto chyb stačí pouze odkliknout OK na vyskakovacím okně a zadat (tentokrát správné) parametry znovu.

5.2.1 Zadání písmenka mimo rozsah

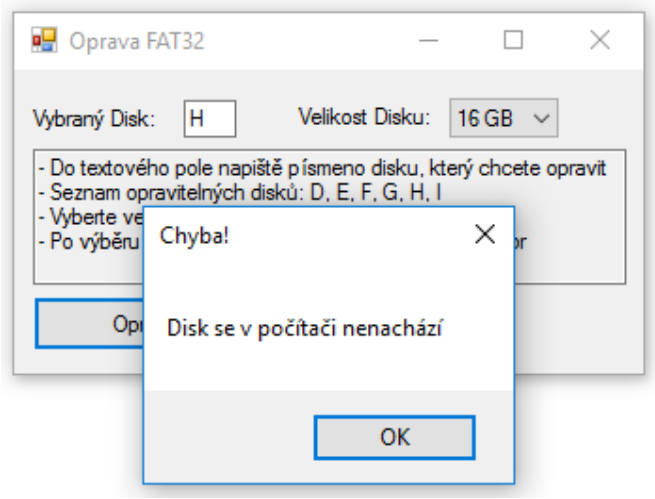
Pokud je při určování písmene poškozeného disku zadáno písmeno mimo povolený rozsah, zobrazí se okno s chybovou hláškou, zobrazené na obrázku 5.4. To zahrnuje veškeré znaky, které jsou různé od D, E, F, G, H, I, nebo jejich obdoby v malých písmenech.



Obr. 5.4: Chybová hláška pro disk mimo rozsah

5.2.2 Zadání disku, který se nenachází v počítači

Pokud je při určování písmene poškozeného disku (z povoleného rozsahu) zadáno takové, jehož přidružený disk se v počítači nenachází, zobrazí se okno s chybovou hláškou viz obrázek 5.5.



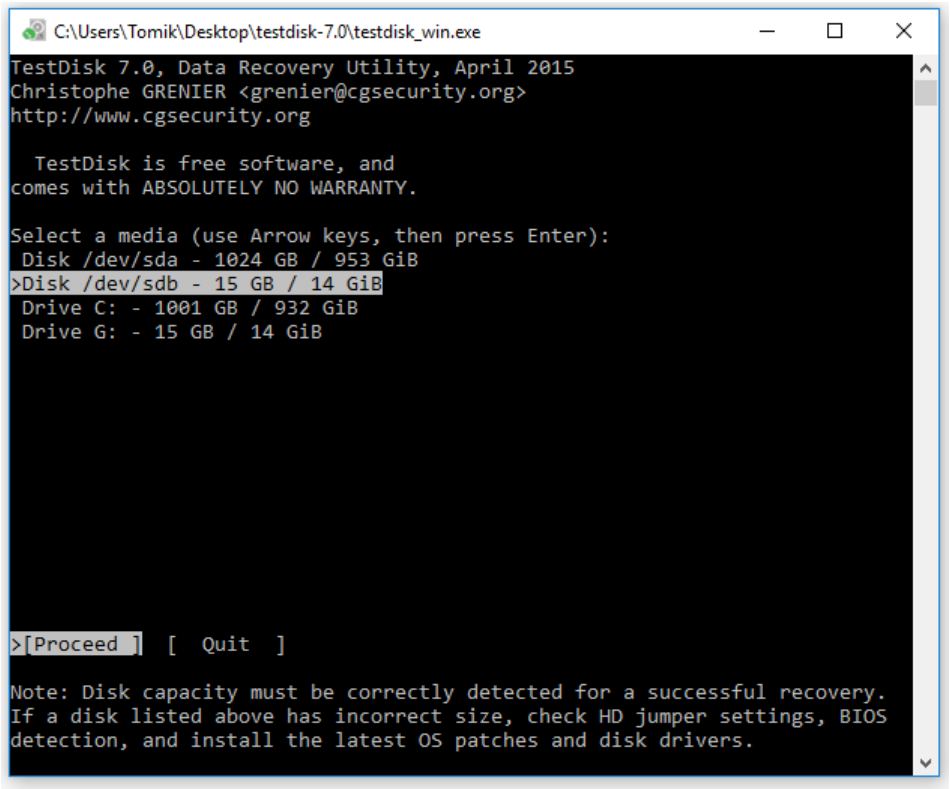
Obr. 5.5: Chybová hláška pro disk nepřipojen k počítači

6 TESTDISK

Následující kapitola se zabývá programem pro práci s diskovými oddíly s názvem TestDisk a jeho porovnáním s aplikací, která byla vytvořena v rámci této diplomové práce.

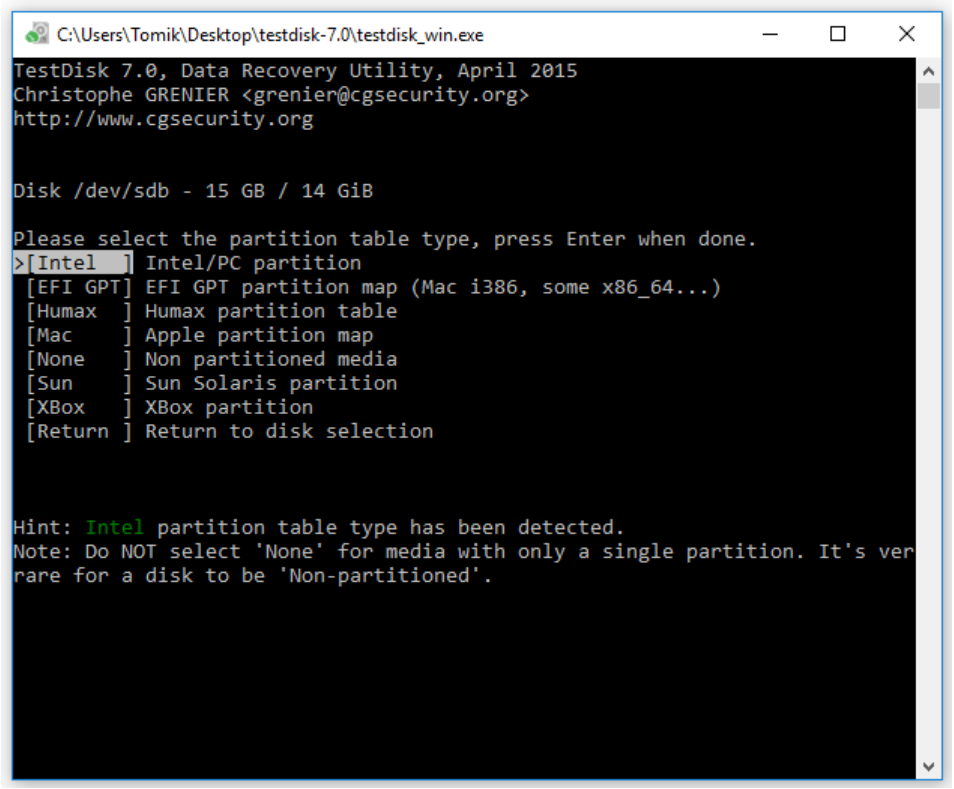
Program TestDisk je komplexní software s licencí opensource, pro obnovu diskových oddílů, jejich částí, nebo celých disků. Celý program pracuje v textovém režimu, nicméně pro jeho používání není potřeba zadávat jednotlivé příkazy. Ovládání programu funguje na principu výběru možností z výběrových nabídek v několika úrovních, v závislosti na požadované proceduře, kterou má program vykonat. Funkcionalita tohoto programu je mnohem rozšířenější, než mnou napsaná aplikace. Přímé porovnání těchto dvou programů je tedy na místě pouze pro funkci obnovení Boot sectoru [17]. Postup pro tuto akci je v TeskDisku následující:

Nejdříve proběhne výběr disku k obnově.

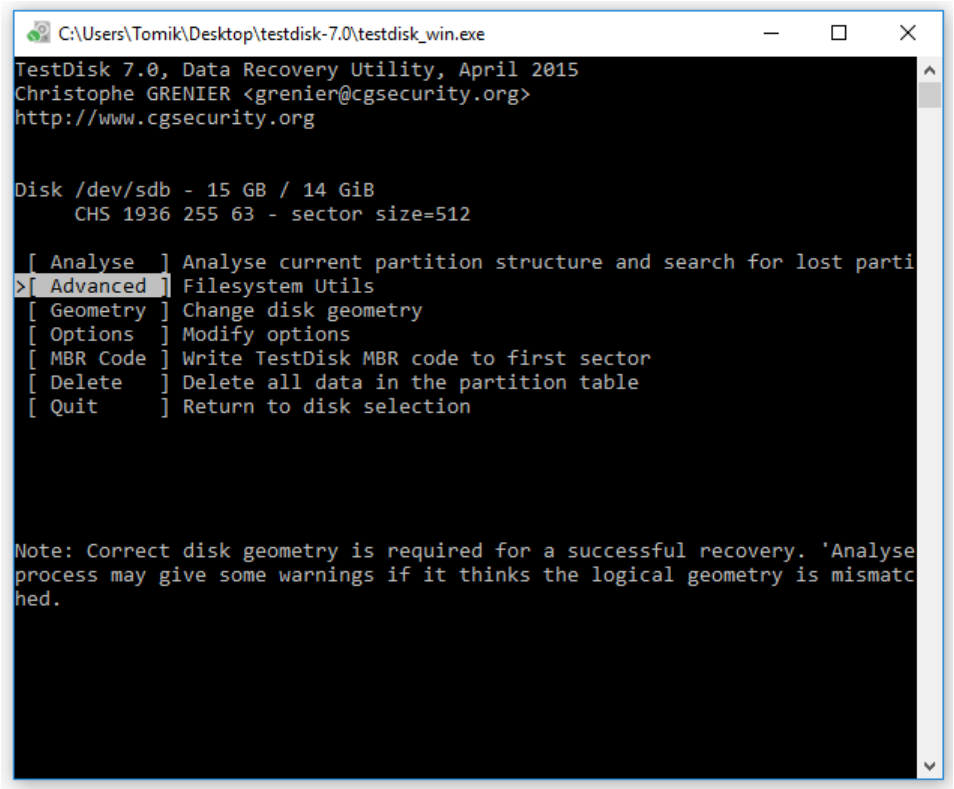


Obr. 6.1: Výběr disku k obnově

Následuje výběr typu tabulky oddílů. TestDisk v tomto dokáže uživateli napovědět správnou volbu. V tomto případě bylo doporučena i zvolena položka Intel/PC partition 6.2. Mezi dalšími nástroji, které TestDisk nabízí je potřeba vybrat možnost Advance, stejně, jako je ukázáno na obrázku 6.3, aby se program dostal k možnosti obnovy Boot sectoru.

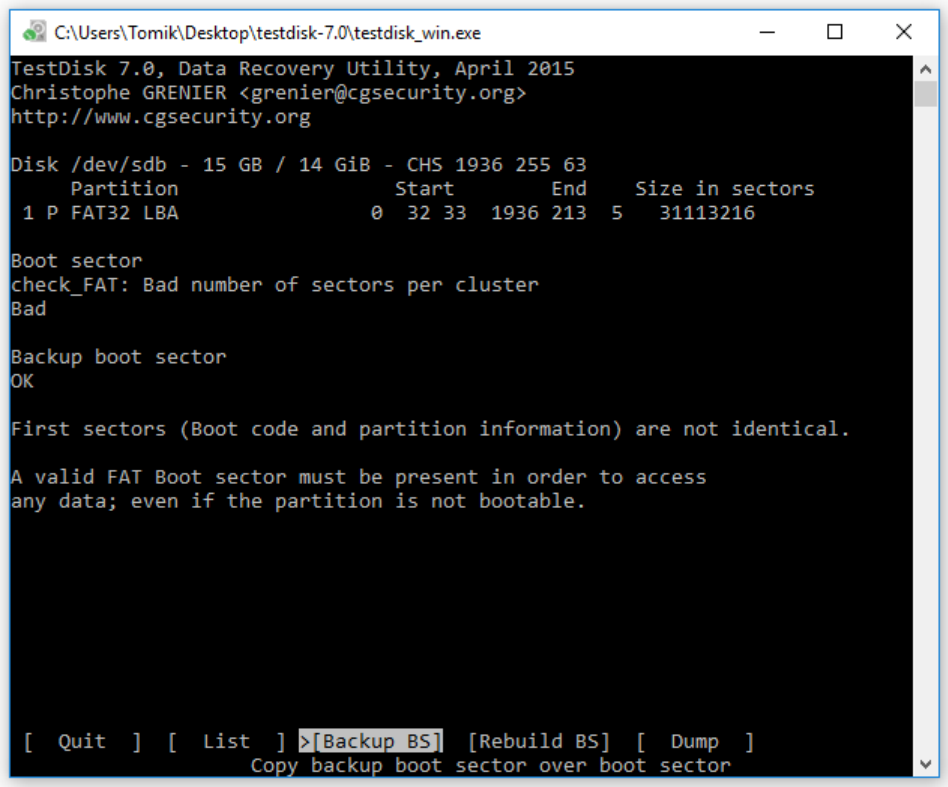


Obr. 6.2: Výběr typu tabulky oddílů



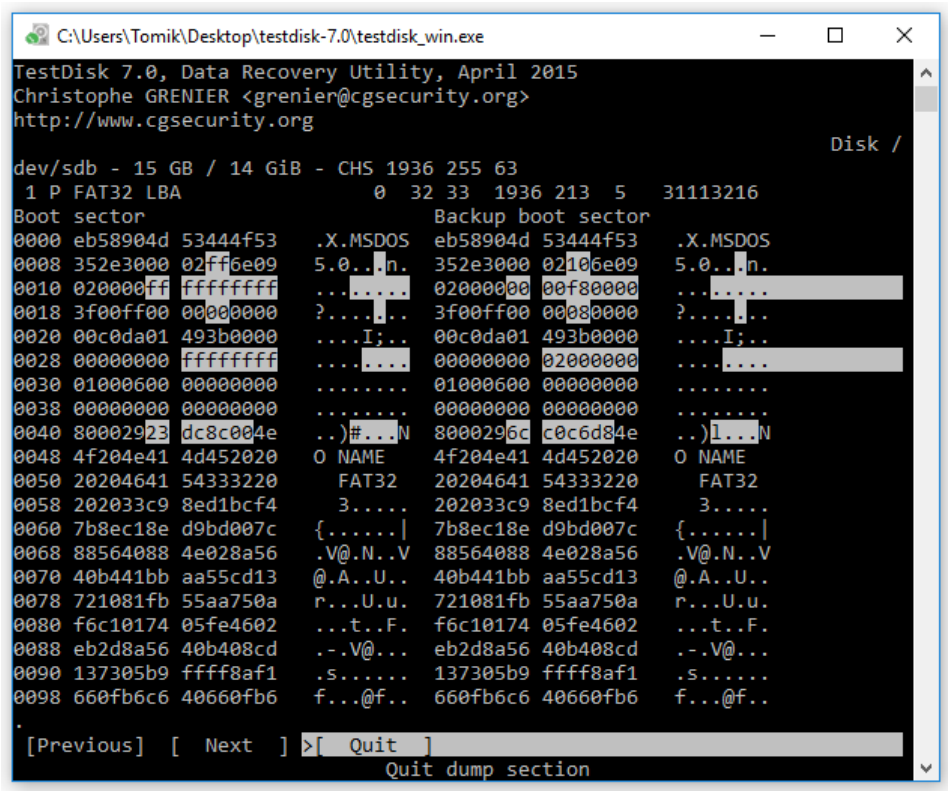
Obr. 6.3: Výběr správné funkce programu

V posledních krocích se už jen vybere požadovaný oddíl disku (v tomto případě má disk jen jeden oddíl), v následné nabídce je zapotřebí vybrat možnost Backup BS.

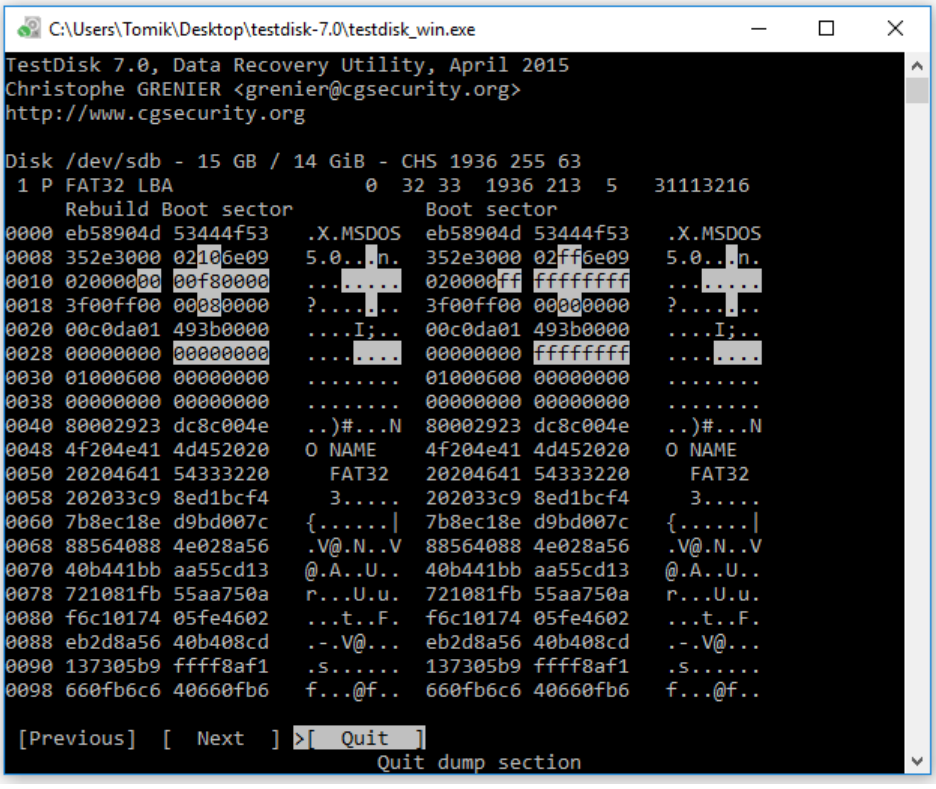


Obr. 6.4: Obnova Boot sectoru

Pro ověření funkce obnovy pomocí TestDisku byly za použití hexeditoru HxD [10] manuálně přepsány bajty na pozicích, které mnou napsaný program opravuje, na hodnoty $\mathtt{FF}_{\mathtt{H}}$. Výjimkou byly první tři bajty určující skokovou instrukci k boot kódu, jejichž hodnoty jsou obnovovány na základě statistického výskytu, nikoliv specifikace 3.2. Systém FAT32 ukládá kopii Boot sectoru zpravidla do šestého sektoru. Volba Backup BS tedy nedělá nic jiného, než že zkopíruje obsah šestého sektoru a vloží jej do sektoru nultého. Metoda obnovy Boot sectoru s využitím kopie zálohy je logicky závislá na tom, že tato záloha není sama poškozena. Pokud by byl šestý sektor rovněž poškozen, nabízí TestDisk možnost vytvořit Boot sector zcela nový. K tomuto účelu slouží volba Rebuild BS (stejná nabídka, ve které se nachází Backup BS 6.4). Tento proces trvá pro 16GB paměťovou kartu přibližně 25 minut. I přes to, že program proběhne bezchybně až do konce, není obnova Boot sectoru účinná. Disk je stále nečitelný a při pokusu o přístup se zobrazí stejné okno s chybovou hláškou, jako na obrázku 3.8. Tento výsledek se projevuje pro vícero různých medií (16GB paměťová karta, 8GB flash disk). Zvláštní věcí je ta, že důležitý obsah nultého sektoru se pro obnovu pomocí kopie zálohy a obnovu metodou vytvoření téměř shoduje.



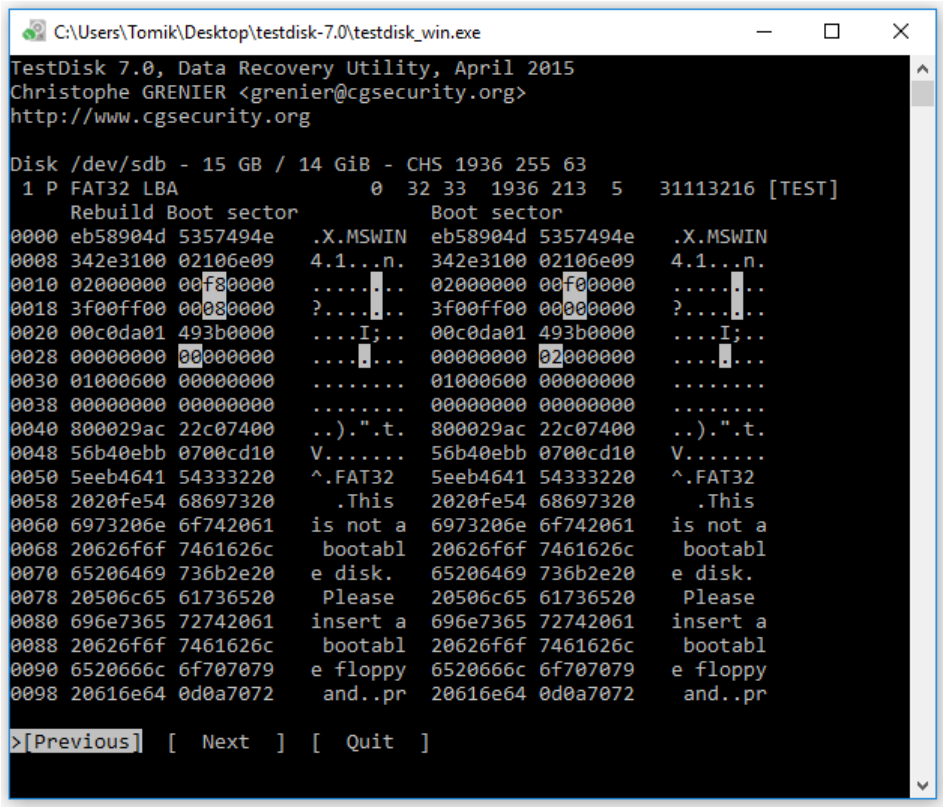
(a) Obnova Boot sectoru pomocí zálohy



(b) Obnova Boot sectoru pomocí jeho vytvoření

Obr. 6.5: Závislost zadaných parametrů na funkci programu

Jak je dobře vidět na porovnání 6.5, jsou sektory pro opravu téměř totožné. U prvního z obrázků je podstatný sloupec s názvem Backup boot sector, u druhého obrázku pak sloupec Rebuild Boot sector. TestDisk rozdíly mezi Boot sectory označuje šedým zvýrazněním, je tedy snadné určit odlišnosti. Jediný rozdíl je v bajtech 43_H - 47_H. Tyto bajty reprezentují sériové číslo oddílu a jejich vliv na funkci media je nulový. Tento fakt tedy znamená, že pokud dojde k poškození media, které zasáhne jak nultý, tak i šestý sektor disku, je mnou napsaný program schopen opravit to, co aplikace TestDisk nedokáže.
Bylo rovněž zjištěno, že pokud je Boot sector přemazán nulami, následně znovu vytvořen pomocí TestDisku a poté mnou napsanou aplikací, tak disk opět funguje bez problémů. Z toho tedy plyne, že TestDisk dokáže opravit i data, která moje aplikace opravit nedokáže 3.2, nicméně pro obnovu správné funkce media je zapotřebí i proběhnutí mého programu. Rozdíl mezi Boot sectory před zásahem mé aplikace je znázorněn na obrázku 6.6, kde sloupec Rebuild Boot sector představuje Boot sector, který vytváří TestDisk a sloupec Boot sector je po zásahu mojí aplikace.



Obr. 6.6: Porovnání Boot sectoru vytvořeného TestDiskem a po zásahu mou aplikací

Je patrné, že změny mezi těmito sektory jsou celkem tři. První z nich je bajt 15_H, který popisuje druh media, kdy hodnota $F0_H$ platí pro vyměnitelná media, nicméně hodnota $F8_H$ je rovněž validní, neboť bývá na této pozici zapsána po formátování

disku a dle specifikace odpovídá pevným diskům [18]. Další bajt je $1D_H$, představující jeden ze skupiny bajtů, které určují počet skrytých sektorů v oddílu. Poslední změnou je bajt na pozici $2C_H$, který je opět součástí skupiny bajtů, která v tomto případě určuje začátek kořenového adresáře. Toto je ta hodnota, která způsobuje chybu po vytvoření Boot sectoru pomocí TestDisku. Dle specifikace Microsoftu by měla být hodnota bajtu na pozici $2C_H$ typicky 02_H. TestDisk toto doporučení z nějakého důvodu nerespektuje a díky tomu se obnova Boot sectoru v podání tohoto programu stává nefunkční.

7 GNU DDRESCUE

Program ddrescue je nástroj určený pro operační systémy Linux, který dokáže číst a obnovovat data i z poškozených souborů a medií. Pokud je při procesu záchrany použit speciální soubor, nazývaný mapfile, je obnova dat velmi účinná. Program umožňuje obnovu dat kdykoliv přerušit a pokračovat v ní později ve stejném bodě. Ddrescue nezapisuje nuly na místa vadných sektorů a výsledný soubor nijak nezkracuje, pokud tak není nastaveno. Pokud máme dvě a více poškozených kopií jednoho souboru a spustíme na nich jednotlivě ddrescue s výstupním souborem shodným pro všechny vstupní, je tu možnost, že výstupem bude obnovený soubor bez jakýchkoliv chyb, protože pravděpodobnost výskytu chyby na stejných místech je velmi malá. Díky použití mapfilu se tak dílčí obnovené soubory spojují do jednoho a každá další iterace má snahu obnovit pouze chybějící části souboru (nehrozí tak nebezpečí přepsání již zachráněných dat nějakými poškozenými).[19]

Doména obnovy (Rescue domain)

Jedná se o blok, nebo bloky dat, na kterých se má provádět daná akce (obnova dat). Dá se definovat pomocí několika parametrů, jako například počáteční pozice, nebo velikost. Ve výchozím nastavení obsahuje doména obnovy celý vstupní soubor. Ddrescue se nikdy nesnaží číst jakákoliv data mimo tuto doménu s výjimkou použití módu přímého přístupu k disku (Direct disc access).[19]

7.1 Algoritmus

Ddrescue se nesnaží číst data sekvenčně, ale v první řadě se snaží číst data, která jsou v pořádku a nevykazují žádné chyby. Oblasti, které se jeví jako špatné, poškozené, nebo jen pomalé jsou přeskočeny a ponechány na pozdější dobu. Tímto je maximalizován objem zachráněných dat. Pokud by se například jednalo o pevný disk, který vykazuje hardwarové chyby, mohly by pokusy o pomalé čtení vadných sektorů jen uspíšit jeho úplně selhání. Samotný algoritmus je pak rozdělen do šesti bodů: [19]

1. Volitelné načtení mapfilu, popisující stav předešle přerušené, nebo na více částí rozdělené obnovy. Pokud není žádný mapfile specifikován, je prázdný, nebo neexistuje, pak se celá doména obnovy označí jako nevyzkoušená (non-tried)
2. Copying. Čtení non-tried úseků vstupního souboru. Při tomto procesu jsou všechny chybné bloky označeny jako neupravené (non-trimmed) a jsou přeskočeny. Přeskakují se také oblasti, kde je čtení příliš pomalé. Přeskočené oblasti jsou vyzkoušeny později ve dvou dalších průchodech (před úpravou), kdy každý

z průchodů má opačný směr, než ten předchozí. Třetí průchod se nazývá čistící (sweeping), se zakázaným přeskakováním. (Cílem je vymezit velké chyby rychle, udržet mapfile malý a vytvořit vhodný počáteční bod pro úpravy-/trimming). Ve velkých blocích jsou čteny pouze non-tried oblasti. Trimming (úpravy), scraping (šrotování) a retrying (přezkoušení) je realizováno sektor po sektoru. Každý sektor je zkoušen maximálně dvakrát. Poprvé to je součástí tohoto kroku při čtení velkých bloků a podruhé v jednom z níže popsaných kroků při čtení jednotlivých sektorů.

3. Trimming. Tento proces je proveden v jednom průchodu. Čtení probíhá pro každý non-trimmed blok od jeho začátku sektor po sektoru, dokud není nalezen vadný sektor. Poté se tentýž blok čte od jeho konce sektor po sektoru, dokud není nalezen vadný sektor. Poté se označí nalezené vadné sektory jako vadné a zbytek bloku jako non-scraped (nesešrotovaný) bez snahy o jeho čtení.
4. Scraping. Dává dohromady data neobnoveny v částech copying a trimmung. Scraping je prováděn v jednom průchodu. Každý blok, označený jako non-scraped je čten od jeho začátku, jeden sektor po druhém. Jakýkoliv vadný sektor je označen jako vadný.
5. Retying. Volitelné pokusy o čtení vadných sektorů, dokud není vyčerpán specifikovaný počet opětovných průchodů souborem. Směr čtení je po každém průchodu obrácen. Každý vadný sektor je v každém průchodu vyzkoušen pouze jednou. Ddrescue nemůže nijak zjistit, zda daný sektor nelze obnovit, nebo je obnova možná po určitém počtu pokusů o jeho přečtení.
6. Volitelný zápis mapfilu pro pozdější použití.

Celková velikost chyb (errsize) je součet velikostí všech bloků sektorů s označením vadný sektor. Tato velikost narůstá během procesů trimming a scrapping. Ke zmenšení může naopak dojít při fázi retrying. Non-trimmed a non-scrapped bloky nejsou považovány za chybné. Je potřeba zmínit, že při obnově dat z poškozených bloků může dojít k rozdělení těchto bloků do několika menších (podaří se obnovit část dat uvnitř vadného bloku, načež je tento blok rozdělen ve dví). Toto má za následek snížení celkové velikosti chyb, ale zvýšení jejich počtu. [19]

7.2 Struktura mapfilu

Mapfile je textový soubor, který se dá snadno číst a editovat. Je tvořen třemi částmi komentářů, stavovými řádky a seznamem datových bloků. Každý řádek obsahující znak # je komentářem. Komentáře obsahují verzi programu ddrescue (nebo ddrescuelog), který mapfile vytvořil, verzi příkazového řádku a čas, kdy byl program

spuštěn. Pokud byl mapfile vytvořen pomocí ddrescue, bude obsahovat i čas, kdy byl mapfile uložen a kopii stavové zprávy z obrazovky, popisující prováděné operace (copying, trimming, finished, atd). Tyto komentáře jsem zamýšleny jako informativní pro uživatele. Prvním řádkem, který není komentář, je stavový řádek. Ten obsahuje čísla větší, nebo rovna nule a stavový znak. Číslo značí pozici ve vstupním souboru, na které se program snaží provádět akce. Stavový znak je jeden z následujících: [19]

- **?** Copying non-tried bloků
- ***** Trimming non-trimmed bloků
- **/** Scraping non-scraped bloků
- **-** Opětovné čtení vadných sektorů
- **F** Plnění specifikovaných bloků
- **G** Generování přibližného mapfilu
- **+** Dokončeno

Bloky v seznamu datových bloků musí být sousedící a nepřekrývající se. Každý řádek v seznamu datových bloků popisuje určitý blok dat. Obsahuje dvě čísla větší, nebo rovna nule a jeden stavový znak. První číslo značí počáteční pozici datového bloku ve vstupním souboru, druhé číslo pak představuje velikost tohoto bloku v bajtech. [19] Stavový znak je jeden z následujících:

- **?** Non-tried blok
- ***** Chybný blok, non-trimmed
- **/** Chybný blok, non-scraped
- **-** Chybný blok, vadný sektor (sektory)
- **+** Dokončený blok

Příklad mapfilu:

```
# Mapfile. Created by GNU ddrescue version 1.21
# Command line: ddrescue -d -c18 /dev/fd0 fdimage mapfile
# Start time: 2015-07-21 09:37:44
# Current time: 2015-07-21 09:38:19
# Copying non-tried blocks... Pass 1 (forwards)
# current_pos  current_status
0x00120000     ?
#      pos        size  status
0x00000000 0x00117000  +
0x00117000 0x00000200  -
0x00117200 0x00001000  /
0x00118200 0x00007E00  *
0x00120000 0x00048000  ?
```

Z výše popsaného je patrné, že nástroj ddrescue je software velmi dobře použitelný, pokud dojde k poškození media. Tato funkcionalita může představovat vhodné doplnění pro aplikaci, která vznikla v rámci této diplomové práce. Tento vytvořený program by mohl být použit k obnově Boot sectoru a následně tak přístupu k datům ze zálohy, vytvořené pomocí ddrescue, aby bylo zamezeno dalšímu případnému poškozování čteného media.

8 ZÁVĚR

V rámci diplomové práce bylo potřeba zpracovat hlavně teorii potřebnou k pochopení souborového systému FAT32. Tato teorie popisuje zejména strukturu jednotlivých logických bloků uložených na přenosovém mediu v rámci souborového systému FAT32. Dále bylo potřeba prostudovat teorii samotného principu zápisu na paměťové medium, jeho adresaci a případné čtení. Následný návrh programu, který by měl splňovat zadání, tedy obnovovat poškozená metadata tohoto souborového systému, byl založen zejména na specifikacích vydaným přímo autorem FAT32 - společností Microsoft. Původním plánem bylo obnovovat metadata MBR (Master boot record) a Boot sectoru. Bylo ale zjištěno, že přenosná media MBR vůbec nemají, a proto se návrh programu zaměřil pouze na Boot sector. Pro co nejlepší výsledky bylo hojně využito i experimentální metody, při které byly jednotlivé bajty, či jejich logické celky přepisovány různými hodnotami a podle efektu, který tyto přepisy měly na medium, byly zařazeny do mnou napsané aplikace tak, aby pokryla co nejširší oblast případného poškození. Výsledkem je program, který obnovuje naprostou většinu potřebných bajtů, kromě bajtů $0E_H$, $0F_H$ (rezervované sektory), 20_H, 21_H, 22_H, 23_H (počet sektorů v oddílu) a 24_H, 25_H, 26_H, 27_H (počet sektorů ve FAT). Korektní hodnoty těchto bajtů bohužel nejsou snadno zjistitelné ze specifikace. Zejména pak poslední dvě zmiňované skupiny jsou závislé na konkrétním disku a jeho kapacitě, protože ani dva disky shodné nominální kapacity nedisponují stejným počtem bajtů, tedy ani sektorů. Tato aplikace provádí obnovu bez časové prodlevy a Boot sector obnovuje lépe, než nástroj TestDisk, se kterým byla tato funkcionalita porovnávána. TestDisk sice dokáže obnovit i data, která moje aplikace nedokáže, nicméně je opravované medium i po vytvoření nového Boot sectoru stále nečitelné, kvůli chybě v bajtu $2C_H$, kdy místo typické a specifikací doporučované hodnoty 02_H zapisuje 00_H. Výše popsané skutečnosti tedy splňují zadání diplomové práce, které zní Oprava metadat souborového systému FAT32. Výsledný program je schopný nejen přepisovat chybné bajty, ale i obnovit uživatelský přístup k jinak nečitelnému mediu.

Koncept mnou napsané aplikace by pak po malých úpravách mohl sloužit jako doplněk pro nástroj ddrescue, který slouží pro práci s poškozenými medii, jejichž další čtení by mohlo způsobovat další poškozování. Aplikace vytvořená v rámci této diplomové práce by tak mohla být použita pro obnovu Boot sectoru a v důsledku pak i dat ze zálohy vytvořené právě nástrojem ddrescue.

Výsledný program by tedy mohl být rozšířen o přidanou funkcionalitu v podobě dalších studentských prací. Nabízí se například rozšíření opravy Boot sectoru pro další souborové systému, konkrétně pak velmi hojně používaný NTFS, který je využíván zejména u disků vyšších kapacit, pro jeho prakticky nekonečnou maximální velikost jednoho souboru. Program by také mohl představovat základ pro komplexní projekt,

který by byl schopný opravovat i Master boot record systémových disků pro různé operační systémy.

LITERATURA

[1] KOMOSNÝ, D. a kolektiv *Síťové operační systémy*. Brno: Vysoké učení technické v Brně, 2015. ISBN: 978-80-214-4446. [cit. 16. 10. 2016].

[2] *Souborový systém.*, Wikipedia [online]. 2013, poslední aktualizace 23. 6. 2016 [cit. 16. 10. 2016]. Dostupné z URL: <https://cs.wikipedia.org/wiki/Souborov%C3%BD_syst%C3%A9m>.

[3] *Žurnálovací systém souborů.*, Wikipedia [online]. 2014, poslední aktualizace 29. 5. 2014 [cit. 16. 10. 2016]. Dostupné z URL: <https://cs.wikipedia.org/wiki/%C5%BDurn%C3%A1lovac%C3%AD_syst%C3%A9m_soubor%C5%AF>.

[4] *Compact flash memory card driver technical manual.*, [online]. 2013 [cit. 16. 10. 2016]. Dostupné z URL: <goo.gl/n7qL4k>.

[5] *Cylindr-Hlava-Sektor*, Wikipedia [online]. 2001, poslední aktualizace 23. 12. 2015 [cit. 16. 10. 2016]. Dostupné z URL: <https://cs.wikipedia.org/wiki/Cylindr-Hlava-Sektor>.

[6] *Logical Block Addressing.*, Wikipedia [online]. 2014, poslední aktualizace 17. 2. 2014 [cit. 16. 10. 2016]. Dostupné z URL: <https://cs.wikipedia.org/wiki/Logical_Block_Addressing>.

[7] *File Allocation Table.*, Wikipedia [online]. 2016, poslední aktualizace 17. 2. 2014 [cit. 16. 10. 2016]. Dostupné z URL: <https://en.wikipedia.org/wiki/File_Allocation_Table#FAT32>.

[8] *Microsoft Extensible Firmware Initiative FAT32 File System Specification.*, [online]. 2000 [cit. 16. 10. 2016]. Dostupné z URL: <http://download.microsoft.com/download/1/6/1/161ba512-40e2-4cc9-843a-923143f3456c/fatgen103.doc>.

[9] *Výchozí velikost clusteru pro systém souborů FAT32.*, [online]. 2015 [cit. 16. 10. 2016]. Dostupné z URL: <https://support.microsoft.com/cs-cz/kb/140365>.

[10] *Hexeditor HxD*, [online]. 2015 [cit. 28. 4. 2017]. Dostupné z URL: <https://mh-nexus.de/en/hxd/>.

[11] *MBR FAT32*, [online]. 2017, poslední aktualizace 15. 3. 2017 [cit. 2. 4. 2017]. Dostupné z URL: <https://www.contextis.com/resources/blog/overcoming-problems-within-forensic-analysis/>.

[12] *Master Boot Record*, [online]. 2017 [cit. 2. 4. 2017]. Dostupné z URL: <https://technet.microsoft.com/en-us/library/cc976786.aspx/>.

[13] *CreateFile*, [online]. 2017 [cit. 2. 4. 2017]. Dostupné z URL: <https://msdn.microsoft.com/en-us/library/windows/desktop/aa363858(v=vs.85).aspx/>.

[14] *SetFilePointer*, [online]. 2017 [cit. 2. 4. 2017]. Dostupné z URL: <https://msdn.microsoft.com/en-us/library/windows/desktop/aa365541(v=vs.85).aspx/>.

[15] *ReadFile*, [online]. 2017 [cit. 2. 4. 2017]. Dostupné z URL: <https://msdn.microsoft.com/en-us/library/windows/desktop/aa365467(v=vs.85).aspx/>.

[16] *WriteFile*, [online]. 2017 [cit. 2. 4. 2017]. Dostupné z URL: <https://msdn.microsoft.com/en-us/library/windows/desktop/aa365747(v=vs.85).aspx/>.

[17] *TestDisk*, [online]. 2011, poslední aktualizace 9. 10. 2011 [cit. 2. 4. 2017]. Dostupné z URL: <http://www.cgsecurity.org/wiki/Advanced_FAT_Repair/>.

[18] *Boot sector*, [online]. 2017 [cit. 2. 4. 2017]. Dostupné z URL: <https://technet.microsoft.com/en-us/library/cc976796.aspx/>.

[19] *GNU ddrescue Manual.*, [online]. 2016 [cit. 30. 11. 2016]. Dostupné z URL: <<goo.gl/jLc4Yh>>.

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÉHO CD

Přiložené CD obsahuje celý projekt vytvořený v MS Visual Studio 2015. Tento projekt obsahuje mimo jiné jak zdrojové kódy programu, tak i zkompilovanou aplikaci, připravenou k použití.