VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

EXTRAKCE MULTIMEDIÁLNÍCH DAT Z OBRAZŮ PEVNÝCH DISKŮ

EXTRACTION OF MULTIMEDIA DATA FROM HARD DRIVE IMAGES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **LUKÁŠ CSÁDER**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAN PLUSKAL, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

154321

Ústav: Ústav informačních systémů (UIFS)
Student: **Csáder Lukáš**
Program: Informační technologie
Název: **Extrakce multimediálních dat z obrazů pevných disků**
Kategorie: Bezpečnost
Akademický rok: 2023/24

Zadání:

1. Nastudujte nástroj Autopsy a knihovnu The Sleuth Kit. Seznamte se s možnostmi interoperability v .NET a C/C++ nativní knihovnou pro Linux x86_64. Prozkoumejte možnosti obnovy smazaných/poškozených souborů za pomoci nástrojů/knihoven provozovatelných na platformě Linux x86_64. Seznamte se s nástrojem Maxtor.
2. Vytvořte dataset obsahující obrazy disků ve formátech dd a e01. Dataset musí obsahovat minimálně souborové systémy NTFS, FAT*, ExFAT, UFS 1, UFS 2, EXT2FS, EXT3FS, Ext4, HFS, APFS. Vytvořte reprezentativní soubor multimediálních dat, který uložíte do výše zmíněných souborových systémů a část vhodně poškodíte, aby bylo možné demonstrovat obnovu smazaných/poškozených souborů nástroji dle bodu 1.
3. Navrhněte integraci zvoleného nástroje či knihovny pro obnovu smazaných/poškozených souborů do nástroje Maxtor.
4. Implementujte integraci dle návrhu z bodu 2 a 3. Dbejte správných programátorských zásad dle zvolené metodologie (např. Clean Code) po konzultaci s vedoucím.
5. Otestujte Vaši implementaci na datasetu z bodu 2 a diskutujte dosažené výsledky.

Literatura:

- *SleuthKitWiki* [online] [vid. 2023a-09-27]. Dostupné z: https://wiki.sleuthkit.org/index.php?title=Main_Page
- *The Sleuth Kit: The Sleuth Kit (TSK) Library User's Guide and API Reference* [online] [vid. 2023b-09-27]. Dostupné z: http://sleuthkit.org/sleuthkit/docs/api-docs/4.12.1/

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2 a 3

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Pluskal Jan, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1.11.2023
Termín pro odevzdání: 9.5.2024
Datum schválení: 20.10.2023

Abstrakt

Táto práca sa zaoberá problematikou počítačovej forenznej analýzy a jej neodmysliteľnou úlohou pri detekcii, analýze a obnove digitálnych artefaktov. Hlavným cieľom je identifikácia, extrakcia a dokumentácia multimediálnych artefaktov s cieľom poskytnúť automatizovaný nástroj pre tento účel. Práca pojednáva o možnosti interoperability knižnice The Sleuth Kit v prostredí .NET a integrácie nástroja PhotoRec Carver do aplikácie Maxtor. Dôležitou súčasťou je vytvorenie vlastnej testovacej dátovej sady, ktorá zahŕňa rôzne súborové systémy a deformácie. Dosiahnuté výsledky testovania sú diskutované s ohľadom na ich účinnosť v navrhnutých scenároch. Zhodnocujú sa prínosy a možnosti tohoto nástroja oblasti digitálnej forenznej analýzy.

Abstract

This work addresses the issues of computer forensic analysis and its essential role in the detection, analysis, and recovery of digital artifacts. The main objective is the identification, extraction, and documentation of multimedia artifacts with the aim of providing an automated tool for this purpose. The paper discusses the possibility of interoperability of The Sleuth Kit library in the .NET environment and the integration of the PhotoRec Carver tool into the Maxtor application. An important part is the creation of a custom testing dataset, which includes various file systems and deformations. The achieved results of testing are discussed with respect to their effectiveness in the proposed scenarios. The benefits and possibilities of this tool in the field of digital forensic analysis are evaluated.

Kľúčové slová

extrakcia multimediálnych dát, .NET Framework, The Sleuth Kit, PhotoRec Carver, Obnova súborov, Data Carving (vyrezávanie dát), Forenzná analýza

Keywords

multimedia data extraction, .NET Framework, The Sleuth Kit, PhotoRec Carver, File recovery, Data Carving, Forensic Analysis

Citácia

CSÁDER, Lukáš. *Extrakce multimediálních dat z obrazů pevných disků*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Pluskal, Ph.D.

Extrakce multimediálních dat z obrazů pevných disků

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Jana Pluskala, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Lukáš Csáder
8. mája 2024

Poďakovanie

Touto cestou by som sa chcel poďakovať vedúcemu mojej záverečnej práce pánovi Ing. Janovi Pluskalovi, Ph.D. za vedenie práce, cenné rady a odbornú pomoc, ktorú mi poskytol.

Obsah

Kapitola 1

Úvod

Oblasť počítačovej forenznej analýzy je neodmysliteľnou súčasťou riešenia otázok týkajúcich sa detekcie, analýzy a obnovy digitálnych artefaktov. Hlavným cieľom tejto oblasti je identifikácia, extrakcia a dokumentácia digitálnych artefaktov s cieľom poskytnúť podporu pri vyšetrovaní trestných činov a iných právnych problémov. Forenzná analýza zahŕňa techniky ako je zber dát, ich analýza, rekonštrukcia udalostí a prípadné preskúmanie súborov a informácií v súlade s právnymi požiadavkami a normami. Je to kombinácia technickej odbornosti, právnej znalosti a etických zásad, ktorá sa vyžaduje pre efektívne vykonanie forenznej analýzy. Táto práca sa zameriava hlavne na oblasť zberu dát z obrazov pamäťových médii, aby bolo možné ich ďalej analyzovať. Pre tento účel existuje množstvo nástrojov a prostriedkov pomocou ktorých je možné tieto ciele dosiahnuť. Medzi tieto nástroje patrí aj nástroj Maxtor. Jedná sa o extrakčný nástroj napísaný v jazyku C#, ktorého základ tvorí knižnica *The Sleuth Kit*.

V tejto práci sa venujem analýze možností interoperability knižnice *The Sleuth Kit* v prostredí .NET, návrhu a integrácii nástroja *PhotoRec Carver* do nástroja Maxtor. Dôležitou súčasťou procesu vývoja forenzných nástrojov je ich testovanie, preto je v práci špecifikované vytvorenie vlastnej testovacej dátovej sady. Táto dátová sada zahŕňa často vyskytujúce súborové systémy, ako napríklad NTFS, skupina Ext2/3/4 a ďalšie. Na jednotlivých súborových systémoch sa nachádzajú rôzne druhy deformácii, príkladom sú zmazané či prepísané súbory, fragmentované súbory alebo súbory umiestnené mimo alokačný priestor súborového systému.

Práca systematicky posudzuje a vyhodnocuje efektívnosť knižnice *The Sleuth Kit* a nástroja *PhotoRec Carver* implementovaných v nástroji Maxtor, v kontexte obnovy dát na rôznych súborových systémoch. Ďalej využitie možnosti interoperability medzi TSK a prostredím .NET, ako aj vytvorenie vlastnej dátovej sady a následná analýza súborových systémov.

Kapitola 2 obsahuje popis jednotlivých nástrojov a technológii použitých v rámci práce, popis obsahuje základný princíp fungovania, vstupné a výstupné dáta s ktorými nástroje pracujú. Kapitola 3 odkazuje na popis štruktúry jednotlivých súborových systémov, ktoré sú súčasťou dátovej sady použitej pri testovaní. Tento popis poskytuje informácie o použitých súborových systémoch v dátovej sade, vrátane ich štruktúry, vlastností a charakteristík. Ďalej popisuje vytvorený skript, ktorý slúži na vytváranie jednotlivých obrazov diskov pre účely testovania. Tento skript je zodpovedný za vytvorenie obrazov diskov, ktoré sú naformátované na rôzne typy súborových systémov vrátane rôznych typov poškodení, ktoré sú následne použité pri testovaní funkcionality nástrojov na obnovu dát. Kapitola 4 obsahuje návrh integrácie nástroja *PhotoRec Carver* do nástroja Maxtor. Zvolený nástroj má za

úlohu rozšíriť možnosti nástroja Maxtor o podporu vyrezávania súborov z oblastí mimo alokačný priestor súborového systému. Kapitola 5 sa venuje spôsobu implementácie kľúčových štruktúr a funkcionality, ktorá je opísaná v návrhu.

Kapitola o testovaní 6 detailne popisuje navrhnuté testovacie scenáre a podrobne analyzuje výsledky testovania nástroja Maxtor na vytvorenej dátovej sade. Diskutuje sa o účinnosti nástroja v rôznych scenároch a hodnotí sa jeho schopnosť identifikovať, analyzovať a interpretovať digitálne dáta.

V záverečnej kapitole 7 sa sústredíme na zhodnotenie dosiahnutých výsledkov v kontexte celej práce. Vysvetľuje sa, ako tieto výsledky prispievajú k pochopeniu problematiky a riešia stanovené ciele. Okrem toho sa diskutuje o možnostiach budúceho využitia týchto výsledkov a ich prínosu pre oblasť digitálnej forenznej analýzy. Táto kapitola poskytuje komplexné zhrnutie práce a významných zistení, ktoré môžu slúžiť ako podklad pre ďalšie výskumy a aplikácie v tomto odbore.

Kapitola 2

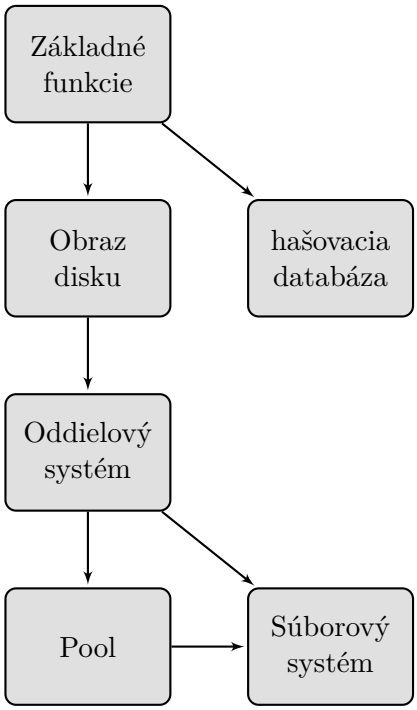
Nástroje použité pre obnovu dát

Táto kapitola obsahuje popis nástrojov použitých v rámci práce. Každý nástroj má, svoju špecifickú úlohu a význam v. Kapitola poskytuje základnú orientáciu a pochopenie funkcionality jednotlivých nástrojov. Ako prvá je predstavená sada nástrojov *The Sleuth Kit* v sekcii 2.1, ktorá tvorí základ pri analýze pamäťových médii a taktiež základ nástroja Maxtor. Popis nástroja *Autopsy* sa nachádza v sekcii 2.2, kde je uvedený popis základnej funkcionality nástroja, ktorý poskytuje grafické užívateľské rozhranie postavené nad knižnicou TSK. Ako ďalší dôležitý nástroj pri extrakcii dát nasleduje *PhotoRec Carver* v sekcii 2.3, ktorý som zvolil ako rozširujúci modul do nástroja Maxtor. Popis samotného nástroja Maxtor sa nachádza v sekcii 2.4. Posledná sekcia 2.5 ktorá sa v tejto kapitole nachádza, poukazuje na spôsob implementácie podpory pre natívnu knižnicu TSK (jazyk C/C++) v nástroji Maxtor, ktorý je implementovaný v jazyku C# a funguje na platforme .NET.

2.1 Knižnica The Sleuth Kit

Sada forenzných nástrojov *The Sleuth Kit* je navrhnutá pre komplexnú analýzu pamäťových médii [5]. Hlavnými prípadmi použitia sú napríklad získavanie metadát súborov zo štruktúr súborových systémov a ich následná evidencia, obnova zmazaných či poškodených súborov alebo vybudovanie časovej osi obsahujúcej sekvenciu udalostí na skúmanom médiu. TSK poskytuje podporu pre najčastejšie používané súborové systémy ako sú NTFS, FAT, Ext2/3/4, HFS, UFS1/2 a APFS. Samotná sada pozostáva z dvoch hlavných častí.

Prvou časťou je terminálové rozhranie poskytujúce nástroje, ktoré možno použiť pre analýzu obrazov disku priamo v termináli. Vstupom nástrojov, je pre väčšinu, obraz disku určený na analýzu. Obraz disku možno analyzovať na úrovniach od samotného obrazu cez diskové oddiely, až po súborový systém a súbory samotné. Druhou časťou je knižnica napísaná v jazyku C/C++. Knižnica umožňuje používateľom integráciu funkcionality do vlastných programov [7]. Aplikačné rozhranie knižnice je rozdelené do vrstiev podľa obrázku 2.1. Pre účely tejto práce sú najdôležitejšie funkcie z oblasti obrazov disku na úrovni celku, diskových oddielov, súborových systémov naformátovaných na jednotlivých oddieloch a samotných súborov, ktoré sa na disku nachádzajú. Proces analýzy samotného obrazu disku začína pomocou funkcii diskových obrazov, ktoré poskytujú dôležité funkcie, ktoré umožňujú manipuláciu so samotným obrazom. Tieto funkcie poskytujú rozhranie pre otváranie podporovaných typov obrazov disku, čítanie jeho obsahu a taktiež jeho korektné uzatvorenie v zmysle uvoľnenia alokovanej pamäte. Po úspešnom otvorení obrazu je možné



Obr. 2.1: Rozdelenie rozhrania knižnice do vrstiev. Prevzaté z [5].

pokračovať v analýze diskových oddielov pomocou funkcii z tejto skupiny. Podobne ako pri diskových obrazoch, sa jedná o funkcie, ktoré zaisťujú otváranie alebo čítanie pomocou priechodu jednotlivých oddielov v rámci jedného obrazu disku a jeho zatváranie. V prípade úspešného spracovania je ďalej možné otvoriť a analyzovať naformátovaný súborový systém. Rozhranie súborového systému ponúka funkcie, umožňujúce čítať metadáta samotného súborového systému a jednotlivých súborov, ako aj ich obsah. Tieto dáta predstavujú cenný zdroj informácii, ktoré možno získať z analyzovaného obrazu. V kontexte nástroja Maxtor, popísaného v sekcii 2.4, tvorí táto knižnica základ pre analýzu a proces extrakcie multimediálnych dát.

2.2 Autopsy

Jedná sa o voľne dostupný komplexný forenzný nástroj ktorý poskytuje grafické užívateľské rozhranie postavené nad knižnicou TSK. To umožňuje prístup k funkcionalitám TSK bez potreby používania príkazového riadku. Program poskytuje množstvo funkcii ktoré umožňujú podrobnú analýzu skúmaného média a taktiež funkcie umožňujúce používateľom vytvárať podrobné správy na dokumentovanie či prezentovanie nálezov v právnom prostredí. Rozhranie nástroja tiež umožňuje tímovú spoluprácu pomocou centrálneho úložiska [6]. Analýza v prostredí *Autopsy* prebieha pomocou modulov. Po úvodnej konfigurácii kde je možné vybrať konkrétne moduly a ich nastavenie, sú spustené nad dátovým zdrojom (napr. obraz disku) a pracujú na pozadí. Počas analýzy informujú užívateľa v reálnom čase o priebehu a nájdených artefaktoch. Tento program poskytuje širokú škálu funkcií, vrátane analytických nástrojov na preskúmanie súborových systémov, vyhľadávanie kľúčových slov, obnovu zmazaných súborov a množstvo ďalšieho. Jeho grafické rozhranie robí analýzu di-

gitálnych dát prístupnejšou a intuitívnejšou pre rôzne typy užívateľov, vrátane tých, ktorí sa nezaoberajú oblasťou digitálnej forenznej analýzy.

2.3 Výber nástroja PhotoRec Carver

Jedná sa o voľne dostupný a multiplatformový nástroj, špecializovaný na obnovu multimediálnych dát. PhotoRec je určený na obnovu súborov na základe ich signatúry a štruktúry dát, bez ohľadu na typ súborového systému, na ktorom boli pôvodne uložené [10]. Tento prístup umožňuje obnovu aj v prípade poškodenia samotného súborového systému. Keďže existuje viacero nástrojov umožňujúcich túto funkcionalitu, bolo treba vyhodnotiť ktorý nástroj je vhodný pre integráciu do nástroja Maxtor. Pri výbere som vychádzal z kritérií ako sú dostupnosť, multiplatformnosť a možnosť automatizovaného spúšťania aby bolo možné nástroj vhodne integrovať.

Tabuľka 2.1: Popis jednotlivých scenárov

Nástroj	**Licencia**	**Platforma**	**Možnosť automatizácie**
EnCase	Proprietárna	Windows	Áno
FTK	Proprietárna	Windows	Nie
PhotoRec	GPLV v2+	Windows, Linux	Áno
Scalpel	Apache 2,0	Windows, Linux	Nie
Foremost	Public Domain	Linux	Áno

V tabuľke 2.1 som zhrnul najpoužívanejšie nástroje, ktoré podporujú funkcionalitu vyrezávania dát. Problémom nástrojov EnCase a FTK pre integráciu do nástroja Maxtor je proprietárna licencia. V prípade nástroja Foremost, ktorý je vyvinutý pre platformu Linux, by nebolo možné zaistiť integráciu aj v prostredí Windows. Nástroj Scalpel je multiplatformná verzia nástroja Foremost avšak jej vývoj nie je ďalej aktívny. Výhodou nástroja PhotoRec je možnosť automatizovaného spustenia kde je možné špecifikovať konfiguráciu nástroja. Konfigurácia zahŕňa napríklad špecifikáciu diskového oddielu, čo bolo jedným z požadovaných prípadov použitia.

Analýza pomocou tohoto nástroja sa začína otvorením obrazu disku za účelom jeho čítania a zistenie základných parametrov obrazu ako je veľkosť alokačného bloku alebo rozloženie diskových oddielov. V prípade, že informácia o veľkosti alokačného bloku nie je známa priamo zo súborového systému (poškodením, formátovaním na iný), PhotoRec číta médium sektor po sektore, hľadá prvých desať súborov a na základe ich umiestnenia dokáže vypočítať veľkosť alokačného bloku. Po zistení veľkosti bloku PhotoRec postupne číta médium blok po bloku. Každý blok sa porovná s databázou signatúr súborov, ktorá je súčasťou programu. PhotoRec identifikuje typ súboru podľa charakteristickej signatúry. Napríklad súbor vo formáte JPEG je identifikovaný pomocou série bajtov:

- 0xff, 0xd8, 0xff, 0xe0
- alebo 0xff, 0xd8, 0xff, 0xe1
- alebo 0xff, 0xd8, 0xff, 0xfe

Počas obnovy sú nájdené súbory zapisované do cieľovej zložky, pre minimalizáciu strát je nutné aby cieľová zložka bola umiestnená mimo spracovávaného média aby nedošlo k prepísaniu doposiaľ neobnovených dát [9]. PhotoRec nie je schopný vo väčšine prípadov získať

názov súboru priamo z metadát keďže sa sústredí na súbory samotné, výnimkou môžu byť súborové formáty obsahujúce názov vo vlastných štruktúrach, ako napríklad dokumenty programu MS Word alebo PDF.

Názov výstupného súboru je generovaný na báze lokácie kde sa súbor nachádzal podla vzťahu 2.1. Názov obnovených súborov začína písmenom ktorého význam je nasledovný:

- **f:** súbor
- **b:** poškodený
- **t:** vložený náhľadový obrázok formátu `.jpeg`

Nasleduje číslo vypočítané nasledovným vzťahom:

$$\frac{O_f - O_v}{S} \tag{2.1}$$

kde:

- O_f — posun súboru na disku v bajtoch
- O_v — posun oddielu na disku v bajtoch
- S — veľkosť alokačného bloku v bajtoch

Pre niektoré súborové systémy, ako napríklad NTFS, exFAT, ext2/3/4, môže byť toto číslo identické s pôvodným číslom klastera/bloku, ak je veľkosť bloku rovnaká s veľkosťou sektora.

Extrahované súbory programom PhotoRec sú postupne ukladané do vytvorených adresárov `recup_dir.1`, kde každých 500 extrahovaných súborov je vytvorený nový adresár s rovnakým názvom a číselná prípona sa postupne inkrementuje. V adresári s číslom 1 sa nachádza aj súbor s názvom `report.xml` obsahujúci dôležité forenzné informácie o nájdených artefaktoch, tými sú veľkosti jednotlivých obnovených súborov posuny súborov v rámci celého disku a dĺžku samotných súborov v bajtoch.

2.4 Maxtor

Nástroj Maxtor predstavuje nástroj pre analýzu a obnovu multimediálnych dát z obrazov disku. Nástroj je vyvíjaný v jazyku C# a pracuje v prostredí .NET na platformách Linux aj Windows. Nástroj poskytuje konzolové rozhranie pre interakciu s užívateľom. Úlohou nástroja Maxtor je automatizovaná extrakcia dát a metadát zo vstupných obrazov pevných diskov a spočíva v integrácii knižnice TSK spomínanej v sekcii 2.1. Pôvodná verzia nástroja obsahovala knižnice s podporou operačného systému Linux na architektúre x64 a v prípade Windows bola podpora obmedzená na architektúru x86. V rámci práce som pridal novú dostupnú verziu knižnice, ktorá zahŕňa aj podporu pre architektúru x64 operačného systému Windows. Tento krok rozšíril dostupnosť nástroja a umožnil jeho použitie na širšom spektre platforiem, čo môže zvýšiť jeho praktickú hodnotu a využiteľnosť pre používateľov na oboch platformách.

Vstupom programu je obraz disku na ktorom sa nachádzajú súbory, ktoré je potrebné obnoviť napríklad po vymazaní, rôznych formách poškodenia súborového systému prípadne pokusu tieto dáta ukryť alebo iným spôsobom zamedziť ich obnove. Podporované formáty obrazov sú dané verziou použitej knižnice TSK. Výstupom programu sú extrahované súbory v špecifikovanom výstupnom priečinku a serializované dáta o jednotlivých nálezoch vo formáte JSON na štandardnom výstupe.

Proces spracovania obrazu disku sa uskutočňuje v niekoľkých krokoch, ktoré zahŕňajú komplexné operácie na úrovni obrazu disku, diskových oddielov a súborového systému. Vykonávanie daných operácii zaisťuje knižnica TSK integrovaná do nástroja Maxtor, spôsob integrácie je popísaný v sekcii 2.5. Keďže pôvodná verzia implementácie tejto knižnice obsahovala podporu rozhrania knižnice na najnutnejšie funkcie, pridal som podporu pre ďalšie funkcie pre prácu s konkrétnym diskovým oddielom a súborovými systémami. Všetky funkcie z rozhrania knižnice obsahujú jednotkové testy. Po načítaní obrazu disku nasleduje otvorenie diskového oddielu, kde sa analyzujú jeho štruktúry a charakteristiky. Následne je skúmaný naformátovaný súborový systém, pričom sa identifikujú jeho dôležité štruktúry a metadáta súborov. Po úspešnom získaní informácií o súborovom systéme sa vstupuje do fázy extrakcie súborov. V tejto etape sú z obrazu disku získavané konkrétne súbory, pričom sa venuje pozornosť ich umiestneniu, typu a dátam, ktoré obsahujú. Prechádzanie prebieha rekurzívne, pričom každý adresár dostupný v obrazu disku je systematicky analyzovaný. Počas tohto procesu sa súbor číta, analyzuje sa jeho typ a vytvára sa nový súbor s príslušným názvom a príponou na základe konfigurácie.

Po otvorení súboru na čítanie je najprv zistený jeho typ zo súborovej prípony a následne sa overuje formou signatúr rôznych podporovaných súborových typov. Identifikácia súboru prebieha tak, že sa porovnávajú sekvencie bajtov v rámci podporovaných súborov, čo umožňuje presnejšie rozpoznať o aký typ multimediálneho súboru sa jedná. Pôvodná verzia nástroja Maxtor obsahovala pomerne obmedzené množstvo signatúr a teda nedokázala presne rozpoznať značnú časť multimediálnych súborov. V rámci práce som sa teda zameral aj na rozpoznávanie rôznych formátov, čo značne rozšírilo schopnosť nástroja presnejšie rozpoznať väčšie množstvo nájdených artefaktov. Väčšinu signatúr som čerpal z projektu `File Signatures Table` [14] a projektu TrID [19]. Jednotlivé signatúry som následne porovnával napríklad so signatúrami získanými zo zdrojového kódu nástroja PhotoRec, prípadne priamo zo súbormi. Je nutné dodať, že kontrola signatúry súborov neposkytuje záruku o správnosti dát, ktoré sa v súbore nachádzajú. V prípade, ak sa aj napriek tomu nepodarí určiť typ súboru, súbor sa preskočí a informácie o ňom sa pridajú do záznamu o preskočených artefaktoch.

Ak je určený typ súboru, vytvára sa cieľový súbor na základe exportných nastavení. Nájdené súbory sú zapisované do výstupného adresára, ktorý je špecifikovaný v argumente pri spustení nástroja. Počas tohto procesu sú informácie o nájdených súboroch zaznamenávané a následne exportované do terminálu vo forme serializovaných dát v JSON formáte.

Výstupné dáta obsahujú nasledujúce informácie o každom nájdenom súbore:

- Cesta k súboru v rámci disku.
- Názov cieľového súboru.
- Prípona typu súboru.
- Názov typu súboru.
- Dĺžka súboru v bajtoch.
- MD5 heš súboru.
- SHA heš súboru.
- Čas vytvorenia súboru.
- Čas poslednej úpravy súboru.

- Čas posledného prístupu k súboru.

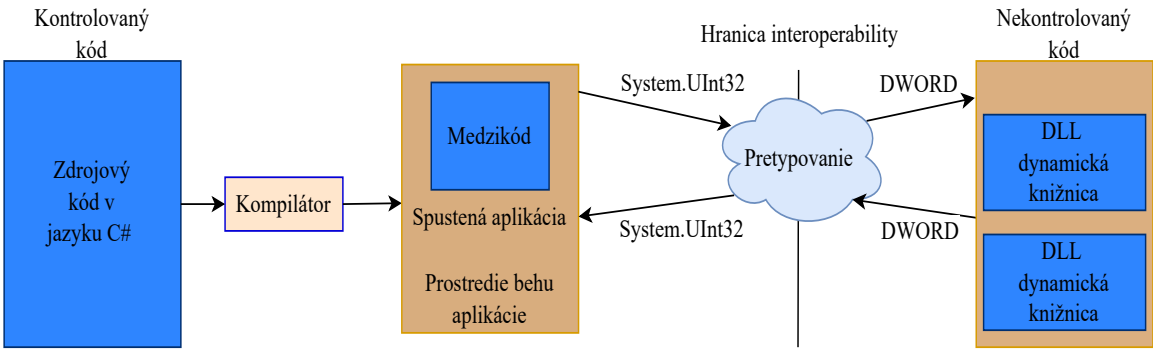
Tieto informácie sú cenné pre užívateľa, ktorý tak získa dôkladný prehľad o nájdených súboroch pri spracovaní obrazu disku.

2.5 Interoperabilita knižnice TSK v .NET

Keďže knižnica TSK je napísaná v C/C++, je nutné zaistiť správnu interoperabilitu medzi kódom knižnice ktorý nie je možné spravovať a spravovaným kódom programu v ktorom sa využívajú funkcie knižnice. Jednou z možností je využiť technológiu `P/Invoke` v .NET [15]. Táto technológia umožňuje obojstrannú komunikáciu medzi spravovaním a nespravovaným kódom. Mechanizmus je znázornený na obrázku 2.2. API pre `P/Invoke` je sústredené do dvoch modulov:

- `System`
- `System.Runtime.InteropServices`

Použitím týchto dvoch názvových priestorov sú sprístupnené nástroje na komunikáciu s nespravovanými komponentami. Prvým krokom je import knižnice TSK do .NET projektu.



Obr. 2.2: Diagram znázorňujúci mechanizmus P/Invoke. Prevzaté z [3]

`P/Invoke` poskytuje atribút `DllImport` na definovanie externých funkcií alebo tried z dynamickej knižnice. Ďalším krokom je definovanie ekvivalentných štruktúr k natívnym štruktúram použitej knižnice aby sa predišlo problémom s prístupom do pamäte a bol zaistený správny prenos dát. V nespravovaných jazykoch, ako je C/C++, je používanie ukazovateľov bežnou praxou pre predávanie argumentov funkciám. Pri interakcii s nespravovanými funkciami z C/C++ pomocou `P/Invoke` je dôležité zaistiť správne spracovanie ukazovateľov a ich hodnôt.

V prostredí .NET možno použiť triedu `Marshal`, ktorá poskytuje sadu metód na manipuláciu s nespravovanými dátovými typmi a podporuje operácie medzi spravovaným a nespravovaným kódom [21]. Táto trieda poskytuje funkcionalitu pre prácu s ukazovateľmi, konverziu dátových typov, kopírovaním pamäti a ďalšími úlohami spojenými s programovaním pomocou `P/Invoke`. V prípade potreby obojstrannej komunikácie, kedy sa funkciám predáva odkaz na inú funkciu, je možné využiť mechanizmus delegátov v .NET. Delegát je typ, ktorý reprezentuje metódu s konkrétnym predpisom. Následne môže byť použitý na vytvorenie a predanie referencie na metódu medzi spravovanými a nespravovanými časťami kódu. V prípade štruktúr musí byť zaistené správne zaradenie jednotlivých položiek

v pamäti. Správne zaradenie jednotlivých položiek v pamäti v štruktúre je kritické pre interoperabilitu s natívnymi knižnicami. Ak sa zoradenie v štruktúre v .NET kóde nezhoduje s tým, ktoré je použité v natívnej knižnici, môžu nastať závažné problémy s prístupom do pamäte a chyby programu. Ak sa položky v štruktúre správne nezaradia (napríklad ak by sa položky nezoradili v poradí, v ktorom sú v natívnej knižnici), môže dôjsť k chybám pri čítaní alebo zápise hodnôt, inkonzistencie dát alebo dokonca pádom programu. Abstraktná Trieda `Safehandler` poskytovaná práve modulom `System.Runtime.InteropServices`, je využívaná na správu zdrojov. Jej účelom je poskytnúť jednotné rozhranie pre prístup, manipuláciu a uvoľňovanie zdrojov, čo umožňuje bezpečné a efektívne používanie vo viacero kontextoch. v nástroji Maxtor sa nachádzajú implementácie tejto triedy, ktoré spravujú dôležité štruktúry poskytujúce informácie o obraze disku, oddieloch či súborovom systéme. Jednotlivé metódy poskytujúce rozhranie knižnice sú implementované formou obalu nad importovanými natívnymi funkciami knižnice. tieto metódy sú umiestnené v triedach podľa poskytovanej funkcionality a využívajú pri tom spomínané implementácie triedy `Safehandler`. Týmto spôsobom je bezpečne oddelený spravovaný kód od nespravovaného.

Kapitola 3

Návrh testovacej dátovej sady

V tejto kapitole sa nachádza prehľad súborových systémov použitých v rámci dátovej sady a taktiež popis jej vytvárania. Keďže oblasť digitálnej forenznej analýzy často čelí výzvam spojeným s forenznou analýzou či obnovou dát, je dôležitou súčasťou vývoja nástrojov ich dôkladné testovanie. Jedným z najvýznamnejších projektov v tejto oblasti je projekt testovania nástrojov počítačovej kriminalistiky (CFTT) Národného inštitútu pre štandardy a technológie (NIST) [12]. Od roku 2000 sa CFTT snaží vytvoriť robustné testovacie metodiky na overenie a hodnotenie výkonu nástrojov digitálnej forenznej analýzy. Tieto snahy majú poskytnúť praktikantom, výskumníkom a ďalším používateľom merateľné záruky o presnosti výsledkov pri počítačových kriminálnych vyšetrovaniach. Prínosom projektu CFTT je nielen poskytnutie dôležitého zdroja informácií, ale aj pohľad na komplexné problémy súvisiace s testovaním nástrojov digitálnej forenznej analýzy. Pri návrhu testovacích scenárov použitých v rámci testovania integrácie knižnice TSK do programu Maxtor som sa čerpal zo správy o testovaní samotnej knižnice TSK dostupnej na [18], ktorú poskytol spomínaný projekt CFTT. Táto správa mi poskytla základ pre vytvorenie vlastnej testovacej sady popísanej v sekcii 3.7, ktorá by mohla účinne overiť funkčnosť a spoľahlivosť implementácie. Testovaciu sadu je možné vytvoriť pomocou skriptu popísaného v 3.8. Testovacia sada obsahuje obrazy s rôznymi súborovými systémami, aby bolo možné overiť ich kompatibilitu s nástrojom Maxtor. Súborové systémy obsiahnuté v dátovej sade sú:

- NTFS (sekcia 3.1)
- FAT (sekcia 3.2)
- EXT2FS, EXT3FS a Ext4 (sekcia 3.3)
- UFS (sekcia 3.4)
- HFS (sekcia 3.5)
- APFS (sekcia 3.6)

Tieto sekcie poskytujú náhľad na dôležité vnútorné dátové štruktúry implementované v jednotlivých súborových systémoch. Pochopenie základov jednotlivých technológii uvedených súborových systémov umožňuje porozumieť spôsobom extrakcie za použitia knižnice TSK. Sekcia 6.1 popisuje scenáre zahrnuté v dátovej sade.

3.1 NTFS

Súborový systém NTFS (New Technology File System), vyvinutý spoločnosťou Microsoft, bol navrhnutý s dôrazom na spoľahlivosť, bezpečnosť a podporu veľkých úložných zariadení. Podporuje škálovateľnosť pre ktorú sa využívajú univerzálne dátové štruktúry, ktoré obklopujú štruktúry s konkrétnym obsahom [4]. Jedným z kľúčových konceptov súborového systému NTFS je asociácia dôležitých údajov so súbormi. Medzi tieto údaje patria administratívne údaje o súborovom systéme, ktoré bývajú v iných súborových systémoch často skryté. Celý súborový systém je považovaný za jednotnú dátovú oblasť, pričom každý sektor môže byť pridelený k jednotlivým súborom.

Kľúčovou štruktúrou súborového systému NTFS je tabuľka Master File Table (MFT). Tabuľka MFT obsahuje informácie o všetkých súboroch a adresároch. Samotná tabuľka je tiež súbor a obsahuje záznam sama o sebe. Tento záznam je nutné spracovať pre zistenie rozloženia MFT na disku. Jedná sa tiež o jediné miesto kde sa táto informácia nachádza a je umiestnené v štartovacom sektore, ktorý je prvým sektorom súborového systému. Jednotlivé záznamy MFT je možné podľa potreby pridávať. Každý záznam obsahuje fixné informácie, ako sú typ súboru alebo prístupové práva a rôzne atribúty, ktoré je možné do záznamu pridať a obsahujú ďalšie informácie o konkrétnom súbore príkladom môžu byť časové značky modifikácie. Metadáta obsahujúce informácie o samotnom systéme sú taktiež uložené v súboroch. Samotné dáta súboru sú uložené pod atribútom `$DATA`. V prípade NTFS, štandardný proces mazania dát, bez dodatočného šifrovania alebo niekoľkonásobného prepísania, proces mazania súboru vyvolá zmeny v záznamoch MFT. Keď je súbor odstránený, záznam v MFT je označený ako voľný a je pridaný do zoznamu voľných záznamov. Samotné dáta súboru zostávajú na disku, avšak iné súbory môžu neskôr využiť miesto, ktoré pôvodne obsadzoval tento súbor.

3.2 Súborové systémy FAT

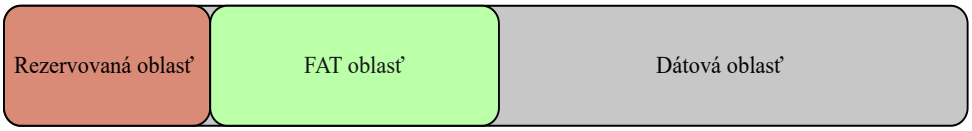
Súborový systém FAT (File Allocation Table) patrí medzi pomerne jednoduché súborové systémy. Hoci sa od verzií Windows NT používa ako predvolený nový súborový systém NTFS, FAT stále zohráva významnú úlohu v digitálnom svete. Často ho nájdeme v pamäťových kartách fotoaparátov a USB kľúčoch a podobných prenositeľných médiách.

[4] V súborovom systéme FAT nájdeme dve dôležité dátové štruktúry: Tabuľka Alokácie Súborov (FAT) a adresár (directory entry). Tieto štruktúry slúžia na viacero účelov a patria do viacerých kategórií modelu.

Osnovným konceptom súborového systému FAT je, že každému súboru a priečinku je pridelená dátová štruktúra nazývaná adresár. Táto štruktúra obsahuje názov súboru, veľkosť, začiatočnú adresu obsahu súboru a ďalšie metadáta. Obsah súboru a adresára je uložený v dátových jednotkách nazývaných zhluky. Ak je súboru alebo adresáru pridelený viac než jeden zhluk, ostatné zhluky sú nájdené pomocou štruktúry nazvanej FAT. FAT štruktúra sa používa na identifikáciu ďalšieho zhluku v súbore a tiež na určenie alokačného stavu zhlukov. Existujú tri rôzne verzie FAT: FAT12, FAT16 a FAT32. Hlavným rozdielom medzi nimi je veľkosť položiek v štruktúre FAT.

Jednou z prvých informácií, ktoré potrebujeme poznať pri analýze FAT súborového systému, je umiestnenie troch fyzických oblastí podľa obrázku 3.1. Rezervovaná oblasť začína v sektore 0 súborového systému, pričom jej veľkosť je uvedená v štartovacom sektore. Pre FAT12/16 je rezervovaná oblasť typicky len 1 sektor, ale FAT32 si môže rezervovať viacero sektorov. V súborovom systéme FAT neexistuje miesto, ktoré by jednoznačne identifiko-

valo typ súborového systému, a teda či sa jedná o systém FAT12, FAT16 alebo FAT32. Konkrétny typ môže byť určený len vykonaním výpočtov z údajov štartovacieho sektora.



Obr. 3.1: Rozloženie súborového systému FAT na disku. Prevzaté z [4].

FAT oblasť obsahuje jednu alebo viac FAT štruktúr a začína v sektore po rezervovanej oblasti. Jej veľkosť sa vypočíta násobením počtu FAT štruktúr veľkosťou každej FAT; obidve hodnoty sú uvedené v štartovacom sektore.

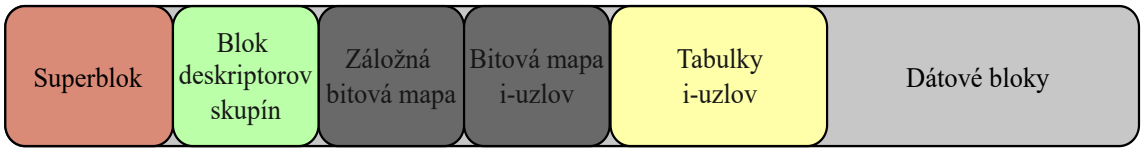
Dátová oblasť obsahuje zhluky, ktoré budú uchovávať obsah súborov a adresárov, a začína v sektore po FAT oblasti. Rozloženie dátovej oblasti je mierne odlišné v prípade FAT12/16 a FAT32. V FAT12/16 je začiatok dátovej oblasti rezervovaný pre koreňový adresár, ale v FAT32 môže byť koreňový adresár kdekoľvek v dátovej oblasti (hoci je zriedkavé, že by nebol na začiatku dátovej oblasti). Dynamická veľkosť a umiestnenie koreňového adresára umožňuje FAT32 prispôsobiť sa vadným sektorom na začiatku dátovej oblasti a umožňuje adresáru rásť podľa potreby. Koreňový adresár FAT12/16 má fixnú veľkosť uvedenú v štartovacom sektore. Počiatočná adresa pre koreňový adresár FAT32 je uvedená v štartovacom sektore a veľkosť sa určuje pomocou FAT štruktúry. Pri mazaní súborov pomocou príkazu sú v tabuľke alokácie súborov (FAT) príslušné záznamy označené ako voľné. Samotné dáta súborov stále zostávajú na disku, kým nie sú prepísané novými údajmi.

Štruktúra súborového systému exFAT sa líši od predchádzajúcich verzií FAT, ako je FAT16 a FAT32. exFAT používa iba jednu tabuľku alokácie súborových clustrov (FAT), ktorá je uložená na začiatku zóny systému pred samotným zhlukom clustrov. Na rozdiel od FAT32, ktorý používa len 28 z 32 bitov, exFAT využíva celý rozsah 32-bitových čísel. Tabuľka alokácie súborových clustrov už nie je používaná pre alokáciu, namiesto toho je zodpovedný súbor BITMAP [20]. Pri zmazaní súboru sa môžu vyskytnúť dva rôzne prípady. Prvým je, že súbor nevyužíval reťazenie v tabuľke FAT. V tomto prípade nie je obsah súboru roztrúsený a jeho obsah môže byť obnovený pomocou dostupných informácií o prvom zhluku a veľkosti súboru. Obnovenie sa potom uskutoční ukladaním všetkých bajtov začínajúcich od prvého zhluku podľa priradeného záznamu v adresári alokácie. Navrhovaná metodika oproti bežným funkciám vyrezávania zachováva súvisiace metadáta a zaručuje, že sa obnoví celý obsah súboru. Druhým prípadom je, že súbor využíval reťazenie v tabuľke FAT. Systém ExFAT podporuje tento typ ukladania na rozdiel od predchádzajúcich systémov FAT. Je to možné vďaka ovládaču súborového systému, ktorý aktualizuje iba priradené zhluky v súbore BITMAP a vyhýba sa aktualizácii FAT, aby minimalizoval zbytočné zápisy na zariadenie. V tomto prípade je potrebné znovu zostaviť reťazenie zhlukov a vymenovať všetky spojené obsahy zhlukov. Obsah posledného zhluku musí byť orezaný podľa veľkosti súboru.

3.3 Systémy EXT2FS, EXT3FS a EXT4

Súborové systémy Ext2 a Ext3 patria medzi predvolené súborové systémy mnohých distribúcií operačného systému Linux. Ext3 je novšou verziou Ext2 a pridáva žurnálovanie súborového systému, ale základná konštrukcia Ext2 zostáva rovnaká. Oba systémy vychádzajú zo súborového systému UFS (Unix File System) popísaného v sekcii 3.4, avšak mnohé zložky systému UFS, ktoré neboli naďalej potreba, boli odstránené. Týmto krokom došlo k celkovému zjednodušeniu štruktúry celého systému [4].

Rozloženie súborového systému začína voliteľnou rezervovanou oblasťou a zvyšok systému je rozdelený do častí, ktoré sa nazývajú blokové skupiny. Všetky blokové skupiny, okrem poslednej, obsahujú rovnaký počet blokov, ktoré slúžia na ukladanie názvov súborov, metadát a obsahu súboru. Rozloženie jednej blokovej skupiny popisuje obrázok popisuje obrázok 3.2. Základné informácie o rozložení oboch systémov sú uložené v dátovej štruktúre superblok, ktorá sa nachádza na začiatku súborového systému. Obsah súboru je uložený v blokoch. Metadáta pre každý súbor a adresár sú uložené v dátovej štruktúre zvanej `i--uzol`, ktorá má pevnú veľkosť a nachádza sa v tabuľke `i-uzlov`. V každej blokovej skupine sa nachádza jedna tabuľka `i-uzlov`. Názov súboru je uložený v dátovej štruktúre adresára, ktorá je umiestnená v blokoch pridelených rodičovskému adresáru súboru. Adresáre sú jednoduché štruktúry, ktoré obsahujú názov súboru a ukazovateľ na `i-uzol` súboru.



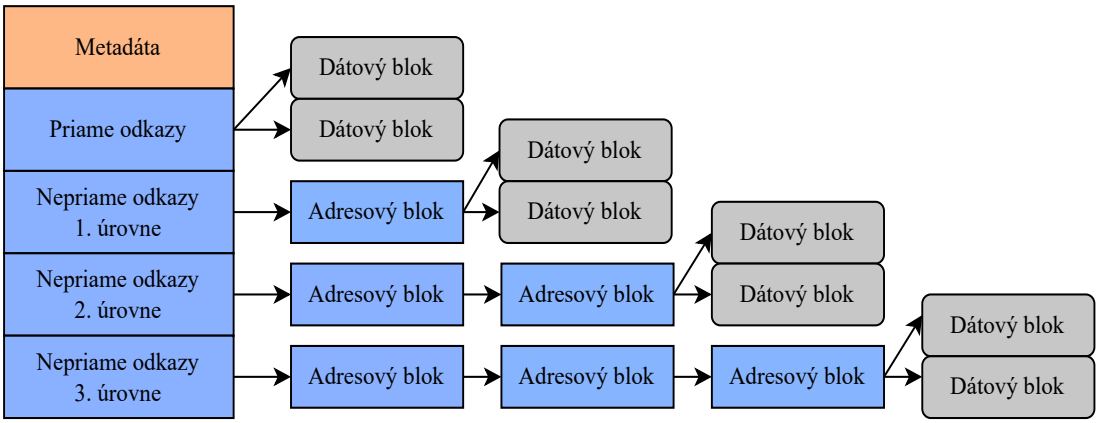
Obr. 3.2: Štruktúra rozloženia súborového systému na disku v Ext2 a Ext3. Prevzaté z [4].

Superblok Ext2/Ext3 sa nachádza 1 024 bajtov od začiatku súborového systému a má veľkosť 1 024 bajtov. Obsahuje základné informácie, ako veľkosť bloku, celkový počet blokov, počet blokov na blokovú skupinu a počet rezervovaných blokov pred prvou blokovou skupinou. Superblok obsahuje aj celkový počet `i-uzlov` a počet `i-uzlov` na blokovú skupinu.

Po super-bloku nasleduje blok pre tabuľku deskriptorov blokových skupín, ktorá obsahuje deskriptor skupiny pre každú blokovú skupinu v súborovom systéme. Záložné kópie super-bloku a tabuľky blokových skupín existujú v každej blokovej skupine, no túto funkciu je možné vypnúť príznakom `sparse_super` kedy sa pre úsporu miesta na disku, ukladajú kópie super-bloku len v niektorých skupinách. Okrem obsahu súborov blokové skupiny obsahujú administratívne údaje, ako sú super-bloky, tabuľky deskriptorov skupín, tabuľky `i-uzlov`, mapy `i-uzlov` a mapy blokov. Deskriptor skupiny popisuje, kde sa tieto údaje nachádzajú. Mapa blokov spravuje stav pridelenia blokov v skupine, a jeho adresa začiatku bloku je uvedená v deskriptore skupiny.

Všetky `i-uzly` majú rovnakú veľkosť, ktorá môže byť definovaná v super-bloku. Každému súboru a adresáru je pridelený jeden `i-uzol`. Obsahom sú informácie o veľkosti súboru, vlastníctve a časových údajoch. Informácie o vlastníctve sú uložené pomocou identifikátorov používateľa a skupiny. V Unixe je každému používateľovi pridelený identifikátor používateľa (UID), na preklad identifikátora používateľa na meno používateľa sa používa sú-

bor `/etc/passwd`. Podobne sa súbor `/etc/group` používa na preklad identifikátora skupiny na názov skupiny. Je dôležité poznamenať, že identifikátor používateľa v `i-uzle` neznamená nutne, že používateľ vytvoril súbor. Identifikátory používateľa a skupiny v `i-uzle` možno kedykoľvek zmeniť pomocou príkazov `chown` a `chgrp`. Pokiaľ ide o časové informácie, `i-uzol` obsahuje časy posledného prístupu, modifikácie, zmeny a odstránenia. Časy sú uložené ako počet sekúnd od 1. januára 1970 UTC. Okrem týchto informácii každý `i-uzol` uchováva adresy prvých 12 blokov, ktoré súbor alokoval podľa obrázku 3.3. Tieto sa nazývajú priame odkazy. Ak súbor potrebuje viac ako 12 blokov, alokuje sa blok na uchovávanie zostávajúcich adries.



Obr. 3.3: Štruktúra odkazovania na dátové bloky v Ext2 a Ext3. Prevzaté z [4].

Ukazovateľ na blok sa nazýva nepriamy odkaz 1. úrovne. Adresy v bloku majú veľkosť štyri bajty, a ich celkový počet v každom bloku závisí od veľkosti bloku. Nepriamy odkaz bloku je uložený v `i-uzle`. Ak má súbor potrebu alokovať viac blokov, než sa zmestí do 12 priamych odkazov a nepriameho odkazu 1. úrovne, použije sa nepriamy odkaz 2. úrovne. Napokon, ak súbor potrebuje ešte viac miesta, môže použiť nepriamy odkaz 3. úrovne. Pre obnovu vymazaných súborov je dôležité vedieť ako jednotlivé systémy pristupujú k mazaniu. Po odstránení súboru v systéme Ext2 je `i-uzol` priradený odstránenému súboru označený ako dostupný úpravou jeho bitových máp `i-uzlov` a dátových blokov. Tento prístup umožňuje obnoviť odstránený súbor, ak jeho obsah nebol prepísaný, pretože štruktúra `i-uzla` bude obsahovať odkazy na dátové bloky. V systéme Ext3 sa však vykonáva dodatočný krok prepisovaním ukazovateľov blokov v štruktúre `i-uzla` po odstránení. To zvyšuje obtiažnosť obnovy súboru a vyžaduje použitie techník známych ako *vyrezávanie dát* na rekonštrukciu súboru.

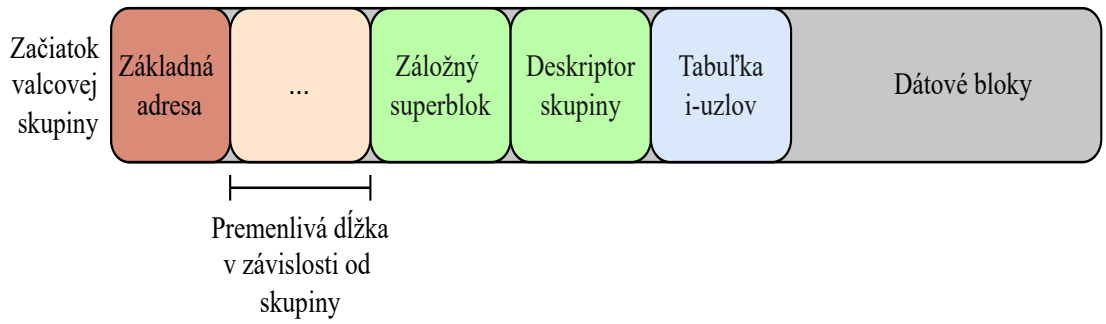
Súborový systém Ext3 zahŕňa žurnál súborového systému ktorý zaznamenáva aktualizácie súborového systému, aby sa mohol súborový systém rýchlejšie obnoviť v prípade nepredpokladanej udalosti, ktorá zapríčiní pád systému. Žurnál zaznamenáva, ktoré blokové aktualizácie sa budú vykonávať, a po aktualizácii označí, že aktualizácia je ukončená. Existujú dva režimy, v ktorých môže žurnál pracovať. V jednom režime sa v žurnály zaznamenávajú len aktualizácie metadát a v druhom režime sa zaznamenávajú všetky aktualizácie vrátane dátových blokov. Aktualizácie sa vykonávajú v transakciách a každá transakcia má číslo postupnosti. Bloky žurnálu obsahujú buď administratívne údaje o transakcii, alebo údaje o aktualizácii súborového systému. Každá transakcia začína blokom deskriptora, ktorý obsahuje sekvenčné číslo postupnosti transakcie a zoznam blokov súborového systému, ktoré sa aktualizujú. Za blokom deskriptora nasledujú aktualizované bloky, ktoré boli opísané

v deskriptore. Keď sú aktualizácie zapísané na disk, nasleduje blok potvrdenia s rovnakým sekvenčným číslom. Deskriptor pre novú transakciu môže existovať za blokom potvrdenia.

Ďalším pokračovaním jeho predchodcov Ext2 a Ext3 je systém Ext4 [17]. Hlavnou zmenou oproti predošlým systémom je použitie extentových stromov pri adresácii dátových blokov. Tento prístup efektívne mapuje logické na fyzické bloky pre veľké súvislé súbory. Maximálna veľkosť jedného súvislého súboru v systéme Ext4 je 16 TB oproti 2 TB pri systémoch Ext2 a Ext3 [8]. Extent je jednoduchý deskriptor, ktorý reprezentuje rozsah súvislých fyzických blokov. Štyri extenty môžu byť uložené priamo v štruktúre `i-uzla`. To je všeobecne dostačujúce pre uloženie malých alebo súvislých súborov. Pre veľmi veľké, silne fragmentované alebo riedke súbory sú potrebné ďalšie extenty. V takom prípade sa používa strom extenov s konštantnou hĺbkou na ukladanie mapy extenov súboru. Koreň tohto stromu je uložený v štruktúre `i-uzla` a extenty sú uložené v listových uzloch stromu.

3.4 UFS

Unix File System (UFS) je súborový systém, ktorý sa vyskytuje v rôznych variantoch v UNIXových systémoch, napríklad FreeBSD alebo NetBSD. Jeho základné koncepty sú rovnaké, ale existujú dva hlavné typy: UFS1 a UFS2. UFS2 umožňuje prácu s väčšími diskami a časovými pečiatkami. Často sa môžme stretnúť s UFS pri analýze Unixových systémov, predovšetkým serverov. [4]Kópie dôležitých štruktúr dát sú uložené po celom súborovom systéme a dáta sú lokalizované tak, aby sa hlavy pevného disku nemuseli veľa pohybovať pri čítaní súboru. UFS je organizovaný do sekcií nazývaných valcové skupiny a veľkosť každej skupiny závisí od geometrie pevného disku. Valcová skupina má štruktúru podľa obrázku 3.4.



Obr. 3.4: Štruktúra súborového systému UFS v rámci jednej valcovej skupiny.
Prevzaté z [4].

Z pohľadu rozloženia UFS využíva superblok na začiatku súborového systému na uchovávanie základných informácií o jeho štruktúre. Obsah každého súboru je ukladaný do blokov, ktoré sú skupinou súvislých sektorov, pričom bloky môžu byť ďalej rozdelené na fragmenty. Metadáta sú uložené v štruktúre `i-uzla`, ktorá obsahuje informácie o súbore alebo adresári. Názvy súborov sú zaznamenané v štruktúrach adresára (directory entry). Každá valcová skupina obsahuje vlastnú tabuľku `i-uzlov`, bitové mapy alokačného stavu fragmentov a kópie superbloku.

[4]UFS využíva tri typy dátových štruktúr pre ukladanie kategórie dát súborového systému: superblok, súhrn valcovej skupiny a deskriptor skupiny. Superblok sa nachádza na začiatku súborového systému a obsahuje základné informácie o veľkosti a konfigurácii. Odkazuje na oblasť súhrnu valcovej skupiny, ktorá poskytuje prehľad o využití valcovej skupiny. Každá valcová skupina má svoj deskriptor skupiny poskytujúci podrobnosti o danej skupine.

Superblok UFS obsahuje základné informácie, ako veľkosť fragmentu a počet fragmentov v každom bloku ďalej tiež informácie o veľkosti valcovej skupiny a mieste uloženia potrebných dátových štruktúr skupiny. Pomocou týchto informácii môžeme zistiť rozloženie súborového systému. Môže obsahovať aj názov zväzku a čas posledného pripojenia súborového systému. Nachádza sa na začiatku súborového systému, UFS1 typicky 8 kB od začiatku, UFS2 typicky 64 kB od začiatku, no nie je to s istotou dané. Kópiu superbloku obsahuje tiež každá valcová skupina. Dátové štruktúry UFS1 a UFS2 použité v superbloku sú mierne odlišné. Superblok obsahuje taktiež odkazy na vstupy do jednotlivých valcových skupín nachádzajúcich sa v oblasti súhrnu valcových skupín.

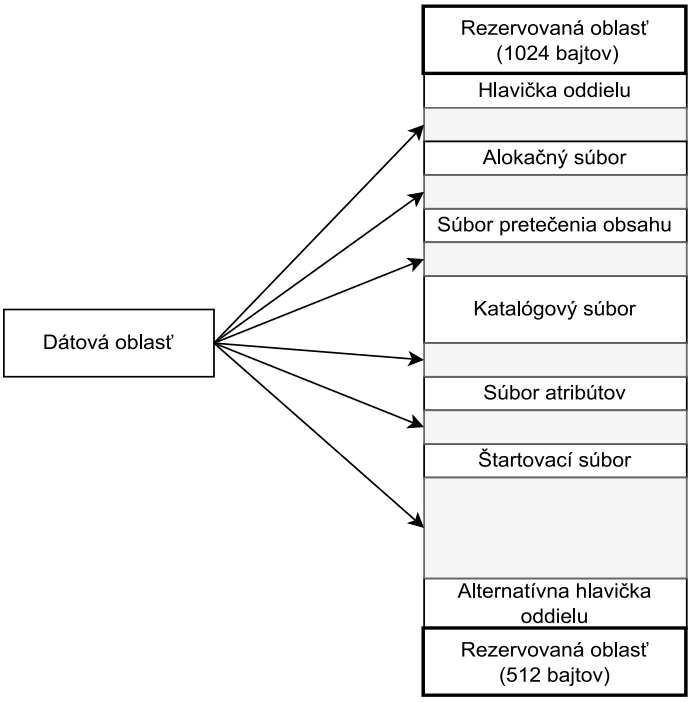
Deskriptor valcovej skupiny UFS obsahuje kombináciu štandardných polí a rozsiahlej oblasti, ktorá sa môže použiť na rôzne tabuľky. Štandardné polia poskytujú informácie o riadení a opisujú organizáciu poslednej časti bloku. Informácie o počte voľných blokov, fragmentov a `i-uzlov` sú tiež uvedené, ale mali by byť zhodné s hodnotami v oblasti súhrnu valcovej skupiny. Deskriptor skupiny obsahuje aj čas posledného zápisu do skupiny.

Záverečná časť deskriptora skupiny obsahuje bitové mapy pre `i-uzly`, bloky a fragmenty v skupine. Obsahuje tiež tabuľky, ktoré pomáhajú nájsť súvislé fragmenty a bloky danej veľkosti. Začiatočná poloha každej z týchto dátových štruktúr je uvedená ako bajtový posun relatívny k začiatku deskriptora skupiny, a veľkosť dátovej štruktúry sa zvyčajne musí vypočítať.

Ak obsahuje súborový systém UFS operačné jadro OS, potrebuje zavádzacie kódy. Tieto kódy sú umiestnené v prvom sektore súborového systému a môžu pokračovať až po superblok alebo dátovú štruktúru disku.

3.5 HFS

HFS (Hierarchical File System) a jeho nástupca HFS Plus (HFS+) sú súborové systémy vyvinuté spoločnosťou Apple pre použitie v jej operačných systémoch [1]. Tieto súborové systémy sa používajú na organizáciu a ukladanie dát na pevných diskoch a iných úložných médiách v zariadeniach od spoločnosti Apple. HFS bol pôvodným súborovým systémom používaným na starších verziách operačného systému Macintosh a používal hierarchickú štruktúru adresárov, čo znamená, že súbory a priečinky sú organizované do stromovej štruktúry. HFS+ bol vylepšením pôvodného HFS a je stále používaný v niektorých verziách macOS. HFS+ priniesol viacero vylepšení vrátane podpory pre dlhšie názvy súborov, lepšiu správu voľného miesta na disku a rozšírenie maximálnej veľkosti súboru. Rozloženie na disku popisuje obrázok 3.5.



Obr. 3.5: Rozloženie súborového systému HFS na disku. Prevzaté z [1]

Hlavné štruktúry ktoré sa nachádzajú v týchto súborových systémoch sú:

- **B-stromy:** HFS/HFS+ využíva B-stromy na organizáciu dát v katalógovom súbore. Tieto štruktúry umožňujú efektívne vyhľadávanie a manipuláciu so súbormi a priečinkami.
- **Katalógový súbor:** Hlavnou dátovou štruktúrou je katalógový súbor, ktorý obsahuje informácie o hierarchii súborov a priečinkov na disku. Je organizovaný ako B-strom a uchováva záznamy o jednotlivých objektoch v súborovom systéme.
- **Bitová mapa súborovej alokácie:** Identifikuje použité a voľné alokačné bloky na disku. Táto dátová štruktúra je kritická pre správu priestoru na disku.
- **Štartovací súbor:** V HFS Plus existuje špeciálny súbor nazývaný štartovací súbor, ktorý zjednodušuje štartovanie z iných operačných systémov než je macOS.

- **Súbor atribútov:** Súbor atribútov obsahuje dodatočné údaje o súboroch a priečinkoch, a je tiež organizovaný ako B-strom.
- **Súbor pretečenia rozsahu:** Súbor pretečenia rozsahu slúži pre uloženie dodatočných rozsahov pre špeciálne súbory, ktoré nedokážu zmestiť všetky svoje údaje do katalógového súboru.

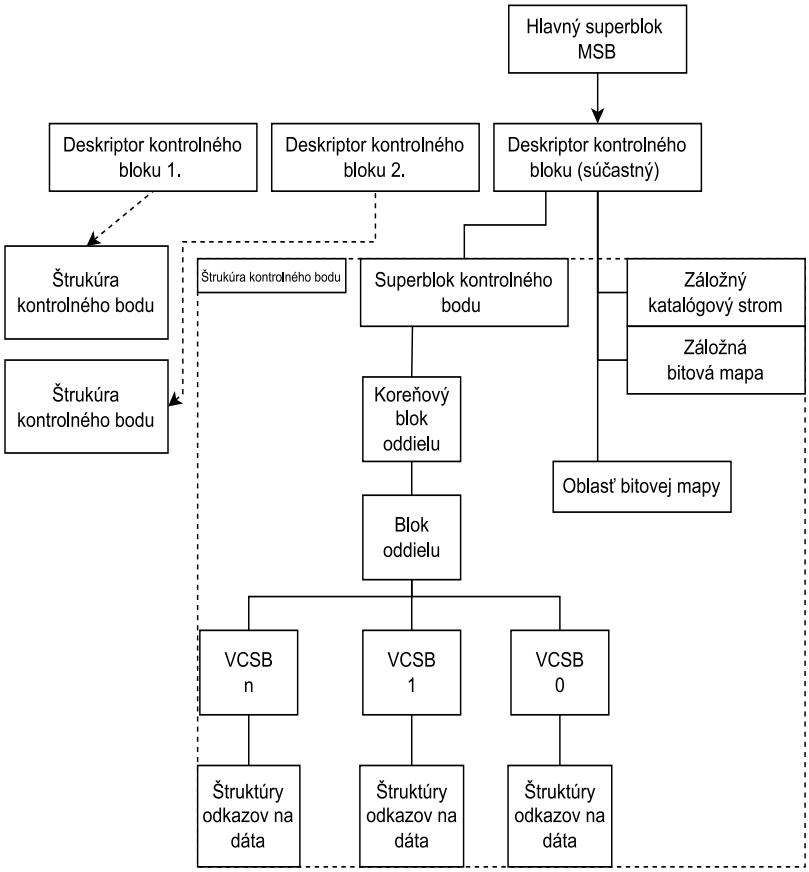
Pri obnove zmazaných súborov je možné využiť implementáciu žurnálovania v tomto súborovom systéme na identifikáciu záznamov katalógových súborov, ktoré reprezentujú súbory, ktoré boli vymazané zo súborového systému. V prípade ak sa nepodarí obnoviť súbor pomocou metadát súborového systému, je potreba použiť vyrezávanie.

3.6 APFS

Apple File System (APFS) je súborový systém od spoločnosti Apple, navrhnutý aby nahradil systém HFS+ avšak nejedná sa o jeho rozšírenie. Z HFS+ poznáme špeciálne súbory, ako je katalógový súbor, súbor atribútov, alokačný súbor a súbor pretečenia rozsahu. Tieto súbory už neexistujú, rovnako ako neexistuje ani žurnál v pôvodnej forme. APFS používa stratégiu atomických operácii pričom stav súborového systému na disku je založený na predchádzajúcom stave na disku [11]. Tento systém definuje kontajner ako jeho základnú jednotku, obsahujúcu metadáta a obsah dát pre súbory, priečinky a ďalšie štruktúry. Kontajner je rozdelený do zväzkov, ktoré sú logickou súčasťou kontajnera. Obrázok 3.6. Voľné dátové bloky kontajnera sú medzi jednotlivými zväzkami zdieľané. Hlavnými komponentami súborového systému APFS sú:

- **Superblok Kontajnera:** Obsahuje informácie o celom kontajneri APFS, ako sú obmedzenia veľkosti blokov, celkový počet blokov a predchádzajúce kontroly. Je najvyššou úrovňou v súborovom systéme.
- **Deskriptor kontrolného bloku:** Obsahuje informácie o metadátach v APFS a je predchádzajúcim kontrolným blokom (okrem hlavného superbloku).
- **Bitová mapa:** Identifikuje použité a voľné bloky. Je spoločná pre všetky zväzky kontajnera.
- **Superblok zväzku:** Obsahuje informácie o danom zväzku.
- **B-strom:** pre súbory a priečiny Zaznamenáva všetky súbory a priečinky zväzku.
- **B-strom extentov:** Obsahuje extenty pre daný zväzok odkazujúce na samotný obsah súboru a jeho dĺžke v blokoch.
- **Snímka súčastného stavu:** Obsahuje užívateľom vytvorený stav zväzku v čase.
- **Kontrolný bod:** Zahŕňa metadáta kontajnera a zväzku v čase. Na začiatku sa nachádza superblok kontrolného bodu. Aktuálny stav je zvyčajne reprezentovaný posledným kontrolným bodom.

Pri zápise dát do kontajnera je vytvorený nový kontrolný bod ktorému je pridelený vlastný superblok. Novo vytvorený kontrolný superblok je označený ako hlavný a obsahuje informáciu o predchádzajúcom.



Obr. 3.6: Štruktúra súborového systému APFS v rámci jedného zväzku. Prevzaté z [11]

APFS poskytuje väčšie možnosti pre obnovu súborov v porovnaní s predchádzajúcimi, vďaka rozsiahlemu používaniu kontrolných bodov. Posunutím počiatočného bodu z hlavného super-bloku na kontrolný bod z predchádzajúcej fázy, je možné rekonštruovať predchádzajúce verzie súborového systému. Pre lokalizáciu kontrolného bodu je možné z hlavného super-bloku nájsť deskriptor súčastného kontrolného bodu za ktorým sa nachádza samotný kontrolný bod obsahujúci informáciu o deskriptore predchádzajúceho kontrolného bodu. Takto sa dajú zmapovať všetky kontrolné body. Pomocou týchto bodov je možné rekonštruovať predchádzajúce stavy systému.

3.7 Popis dátovej sady

Testovacia dátová sada, ktorá je nevyhnutnou súčasťou procesu testovania forenzných nástrojov, pozostáva z obrazov diskov na ktorých sú naformátované vyššie spomínané súborové systémy a obsahujú rôzne multimediálne súbory. Pri vytváraní tejto dátovej sady som uplatnil prístup využívajúci virtuálne stroje s rôznymi operačnými systémami, ktoré sú schopné formátovať jednotlivé obrazy diskov a manipulovať s konkrétnymi typmi súborových systémov. Pre formátovanie súborových systémov som použil nasledovné kombinácie:

- **FreeBSD** — pre formátovanie súborových systémov UFS1/2.
- **MacOS** — pre formátovanie súborového systému APFS.
- **Linux** — pre formátovanie skupiny súborových systémov Ext2/3/4, FAT*, NTFS a HFS.

Jednotlivé operačné systémy poskytujú vhodné nástroje pre formátovanie požadovaných súborových systémov. Testovacia dátová sada je kľúčovým prvkom v procese vývoja a overovania funkčnosti extrakčného nástroja. Je dôležité, aby bola táto dátová sada čo najviac reprezentatívna a obsahovala rôznorodé typy súborov, ktoré môžu byť nájdené v reálnych prípadoch použitia. Do dátovej sady som zahrnul rôzne typy poškodení a situácie, ktoré sú bežné v reálnom svete, ako sú fragmentované súbory a iné typy poškodení súborových systémov. Tieto prípady sú dôležité, pretože extrakčný nástroj by mal byť schopný efektívne pracovať aj s takýmito problémami a obnoviť čo najviac údajov. Prítomnosť týchto prípadov v dátovej sade umožňuje overiť, ako sa extrakčný nástroj správa v rôznych problematických situáciách a ako efektívne dokáže obnoviť údaje aj v prípade poškodených alebo fragmentovaných súborov. Vytvorená dátová sada obsahuje:

- obrázky vo formáte `.jpg`
- dokumenty formátu `.pdf` jeden lineárny a druhý nelineárny (komplexnejšia štruktúra)
- súbor vo formáte JSON
- audio a video súbory vo formátoch `.wav`, `.mov` a `.wmv`
- prezentáciu programu MS Powerpoint
- dokument programu MS Word
- hárok MS Excel
- komprimovaný priečinok `.zip` a `.rar`

Túto dátovú sadu bude možné reprodukovať pomocou vytvoreného skriptu. Tento skript vytvorí obrazy diskov a naformátuje vyššie spomínané súborové systémy. Keďže sada obsahuje súborové systémy, ktoré sú podporované rôznymi operačnými systémami je nutné použiť virtualizáciu daných systémov.

3.8 Návrh skriptu na vytváranie testovacích obrazov

Hlavným cieľom navrhnutého skriptu, je podpora testovacích procesov extrakčných nástrojov, ako je napríklad nástroj Maxtor. Tento skript je zameraný na zefektívnenie procesu testovania a overovania funkčnosti extrakčných nástrojov prostredníctvom automatizácie vytvárania testovacie obrazy diskov.

Pre dosiahnutie tohto cieľa, skript využíva nástroj QEMU, pomocou ktorého zabezpečuje virtualizáciu operačných systémov. Týmto spôsobom sa zabezpečuje, že testovacie obrazy diskov sú vytvorené v prostredí, ktoré simuluje reálne operačné systémy.

Navrhnutý skript poskytuje prostriedok na rýchle a opakovateľné vytváranie testovacích obrazov diskov, ktoré budú slúžiť ako vstupné dáta pre testovacie scenáre extrakčných nástrojov, v tomto prípade nástroja Maxtor. Hlavnou motiváciou vytvorenia skriptu bolo automatizovať proces vytvárania testovacích obrazov diskov tak, aby bol čo najmenej náročný na manuálnu intervenciu. Pomocou nej možno zabezpečiť konzistentné a opakovateľné vytváranie testovacej sady pre extrakčné nástroje, čo umožní spoľahlivé porovnávanie výsledkov testov. Skript tiež umožňuje prispôsobenie parametrov vytvárania testovacích obrazov diskov podľa potrieb konkrétnych testovacích scenárov a požiadaviek na testovanie extrakčných nástrojov.

Prvým krokom je nastavenie správnej konfigurácie virtuálnych strojov pomocou nástroja QEMU. V skripte je definovaná konfigurácia pre jednotlivé virtuálne operačné systémy. Podproces každého operačného systému sa spúšťa pomocou modulu *Pexpect*, ktorý poskytuje rozhranie pre interakciu so spusteným procesom.

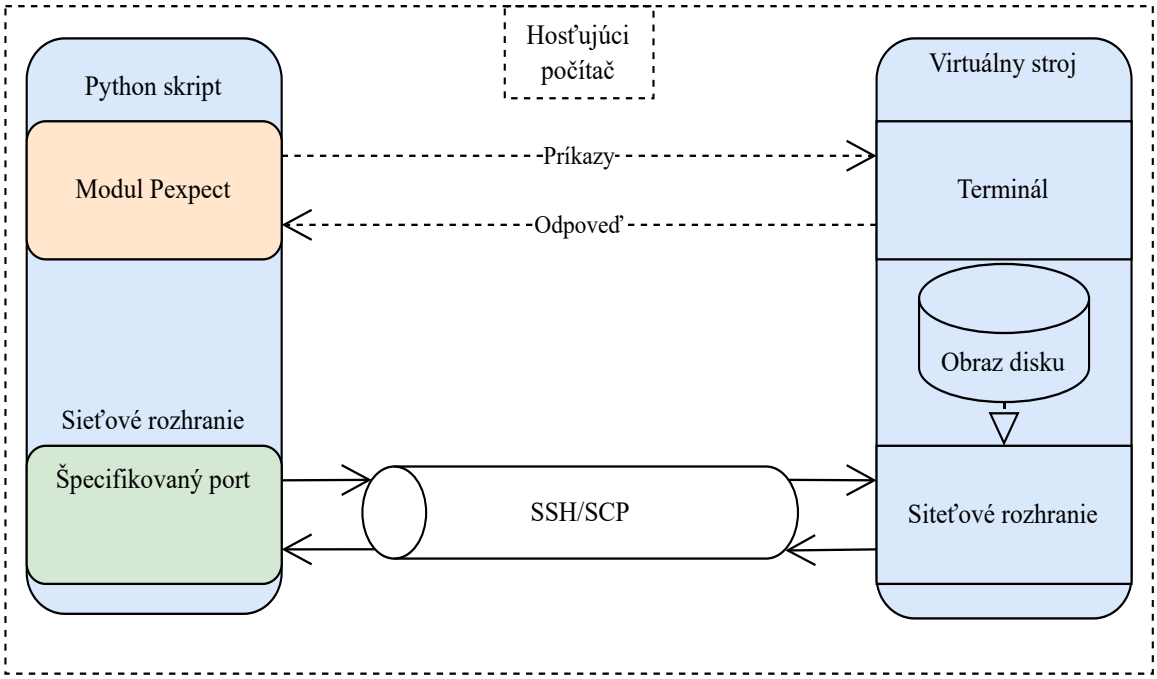
Príkazy na spustenie virtuálnych strojov zahŕňajú nastavenia veľkosti pamäte, počtu jadier CPU a ďalšie. Dôležité je správne nastavenie sieťového rozhrania. Virtuálny stroj je pripojený na virtuálnu sieť, na ktorú sa pristupuje pomocou lokálneho rozhrania a unikátnom porte pre každý virtuálny stroj. Mimo modul *Pexpect*, prebieha komunikácia pomocou protokolu SSH a výmena súborov pomocou protokolu SCP. Pri spúšťaní virtuálneho stroja je nadstavené presmerovanie komunikácie na zvolených portoch hostiteľského stroja, na port virtuálneho stroja na ktorom sa nachádza SSH server. To zaistí možnosť pripojenia pomocou tohoto protokolu na virtuálny stroj.

Prístup pomocou protokolu SSH a SCP je dôležitou súčasťou skriptu a zabezpečuje riadenie prístupu k virtuálnym strojom. Túto komunikáciu zaisťuje trieda zodpovedná za riadenie SSH operácií medzi hostiteľským systémom a virtuálnymi strojmi spustenými pomocou prostredia QEMU. Poskytuje metódy na generovanie SSH kľúčov, výmenu kľúčov medzi hostiteľom a hostiteľským strojom a medzi hostiteľom a virtuálnymi strojmi, ako aj prenos súborov medzi nimi. Pomocou kľúčov je možné vykonávať všetky potrebné operácie bez potreby zadávať prihlasovacie údaje pri každom dotaze tohoto typu. Výmenu kľúčov je nutné vykonať v prípade ak na danom počítači nebola pred tým vykonaná, táto operácia je trvalá a stačí ju na danom počítači vykonať raz.

V inicializačnej metóde triedy sa nastavuje meno hostiteľského stroja podľa užívateľa ktorý program spustil a jeho IP adresa. Tieto údaje sú nevyhnutné v momente prenosu vytvorených obrazov naspäť na hostiteľský stroj.

Skript taktiež obsahuje metódy na výmenu SSH kľúčov medzi hostiteľom a všetkými virtuálnymi strojmi. Celý proces výmeny som navrhol interaktívne. Hlavným dôvodom je nutnosť zadávania hesiel v momente prihlasovania na virtuálny stroj a aj na hostiteľský.

Každá metóda zabezpečujúca výmenu kľúčov spúšťa virtuálny stroj príslušného operačného systému s trvalými zmenami. Po dobu spúšťania virtuálneho stroja skript opakovane vykonáva pokus o ssh spojenie, až po úspešnom spojení sa vykonáva výmena SSH kľúčov



Obr. 3.7: Diagram znázorňujúci komunikáciu medzi modulmi skriptu

s daným strojom. Po úspešnej výmene sa virtuálny stroj ukončí. Súčasťou tejto triedy sú aj metódy zaisťujúce výmenu a prenos súborov.

Parameter `-snapshot` umožňuje určiť, či sa má použiť virtuálny stroj bez možnosti perzistentných zmien alebo sa majú vykonať akcie vyžadujúce trvalé zmeny v danom systéme. Každá metóda spúšťa proces virtuálneho stroja s príslušnými parametrami podľa definícií v atribútoch triedy. V prípade použitia parametra `-snapshot` sa spustí snímok virtuálneho stroja, ktorý obsahuje zmeny vykonané pri poslednom spustení bez tohto parametra a žiadna z vykonaných zmien sa po skončení neuloží. To zaisťuje konzistentné vytváranie nových obrazov pri každom spustení keďže operačný systém neobsahuje zmeny z predchádzajúcich spustení.

Riadenie a koordináciu všetkých činností súvisiacich s tvorbou obrazov disku zabezpečuje trieda `SubProcessManager`. Jej hlavným cieľom je organizovať a spúšťať podprocesy potrebné na vytvorenie obrazov pre rôzne operačné systémy a súborové systémy. Táto trieda poskytuje rozhranie na spracovanie argumentov, ktoré určujú, aké operácie majú byť vykonané, ako aj na nastavenie parametrov pre spúšťanie virtuálnych strojov a inicializáciu potrebných inštancií tried pre manipuláciu s konkrétnymi operačnými a súborovými systémami.

Hlavná metóda `Run()` tejto triedy spúšťa celý proces tvorby obrazov disku. Najprv kontroluje, ktoré súborové systémy majú byť naformátované na jednotlivých obrazoch. Nasleduje spúšťanie virtuálnych strojov a po úspešnom spustení operačného systému, formátovanie súborových systémov pre každý operačný systém. Postup krokov vytvárania obrazov je nasledovný:

1. **Vytvorenie obrazu disku:** Prvým krokom je vytvorenie obrazu disku o zvolenej veľkosti pomocou nástroja `dd`. Tento príkaz zaslaný cez konzolu vytvára obraz disku, ktorý je inicializovaný na nulové bity.

2. **Pripojenie obrazu disku:** Po vytvorení obrazu je potrebné pripojiť ho ako nové zariadenie k rozhraniu pomocou príkazu `losetup`, aby bolo možné operovať s diskovým obrazom ako s pripojeným zariadením.

3. **Formátovanie súborového systému:** Nasleduje formátovanie súborového systému na predchádzajúci obraz disku. Na tento účel som využil programy poskytnuté samotnými operačnými systémami, na Linuxe som použil program `mkfs`, na FreeBSD `newfs` a na MacOS `diskutil`.

4. **Pripojenie a kopírovanie súborov:** Po vytvorení súborového systému sa obraz disku pripája k určenému priečinku pomocou príkazu `mount`, čo umožňuje kopírovanie súborov z hostiteľa na daný obraz disku. Následne sa kopírujú požadované súbory pomocou príkazu `cp`.

5. **Spôsobenie poškodenia:** Ak je špecifikovaný typ poškodenia, vykoná sa príslušná operácia nad obrazom disku.

6. **Odpojenie obrazu disku:** Po dokončení kopírovania a prípadného poškodenia sa súborový systém odpojí pomocou príkazu `umount` a obraz disku sa odpojí pomocou príkazu `losetup`.

7. **Odoslanie obrazu disku:** Nakoniec sa dokončený obraz disku odošle, pomocou protokolu SCP, z virtuálneho stroja na hostiteľský stroj.

Kapitola 4

Návrh rozšírenia nástroja Maxtor

Kapitola o návrhu je kľúčovou súčasťou akéhokoľvek softvérového projektu, pretože poskytuje detailný plán a usmernenie pre implementáciu. V návrhu sa nachádza popis spracovania a validácie vstupov v sekcií 4.1. Táto časť sa zameriava na zabezpečenie správnej syntaxe vstupných argumentov a ich validácia. Kostra modulu je popísaná v sekcií 4.2, kde sú definované hlavné časti nevyhnutné pre integráciu modulu *PhotoRec Carver* do nástroja Maxtor. V tejto časti návrhu som zahrnul základné usmernenia pre štruktúru tried, ako aj spôsob spracovania a riadenia behu modulu počas integrácie.

4.1 Spracovanie vstupných argumentov

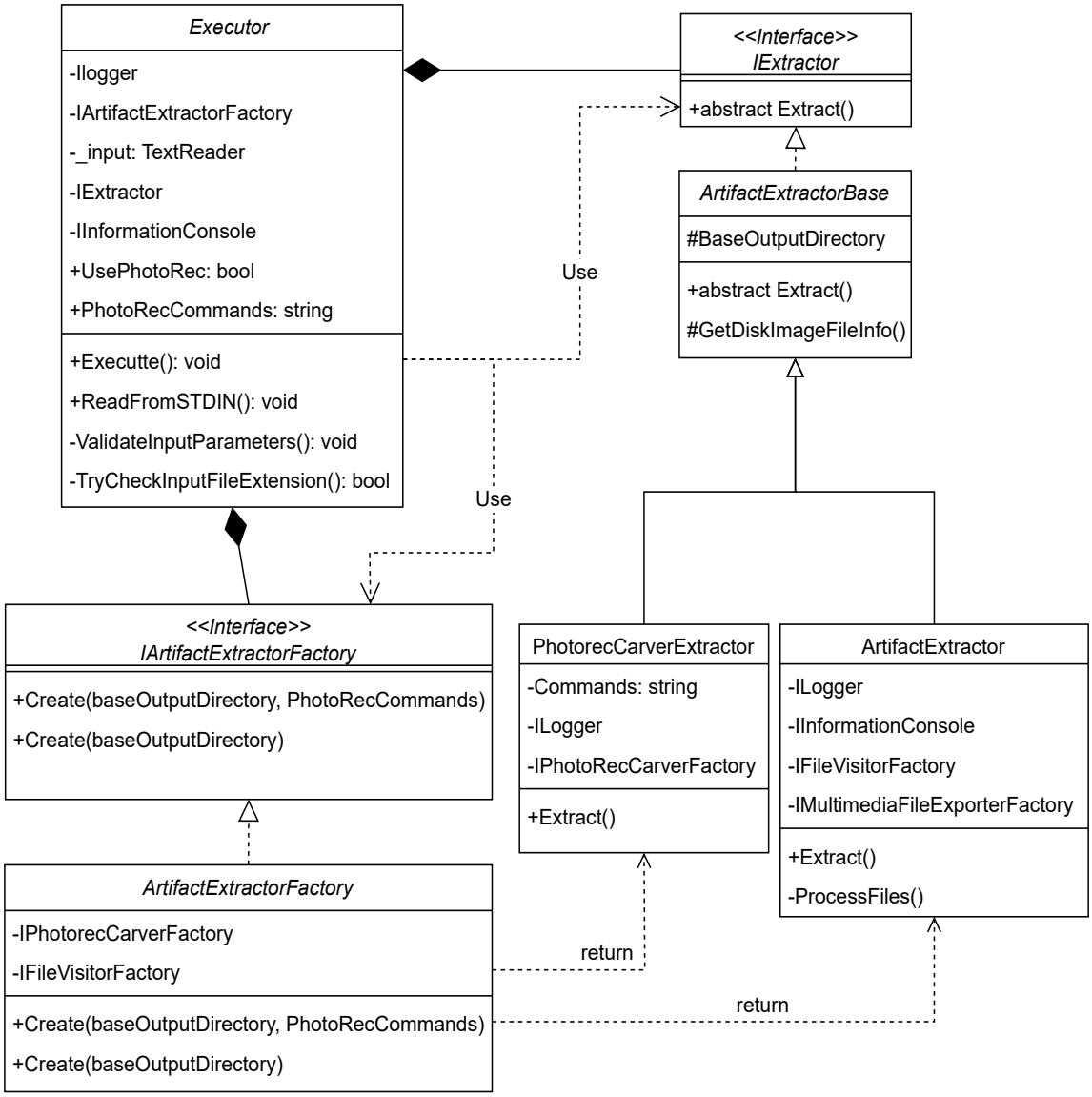
Rozhranie príkazového riadku som navrhol s použitím balíčku McMaster na spracovanie argumentov. Základný návrh využíva štruktúru založenú na bázovej triede, ktorá obsahuje definíciu základných argumentov spoločných pre triedy, ktoré sa starajú o spúšťanie programu. Tieto argumenty sú potom využívané dedičnými triedami pre extrakciu pomocou knižnice TSK alebo nástroja PhotoRec.

Bázová trieda obsahuje základné argumenty, ktoré sú spoločné pre oba extrakčné nástroje. Základným argumentom sú vstupy, ktoré sú navrhnuté tak, aby umožnili spracovanie vstupu pomocou zadaných ciest. Tieto cesty môžu smerovať k jednému vstupnému obrazu alebo k adresáru, ktorý obsahuje dané vstupy, prípadne aj obraz rozdelený na viacero samostatných súborov. Tento návrh umožňuje flexibilné a rozšíriteľné spracovanie vstupov podľa požiadaviek a špecifík.

Výstupný adresár poskytuje primárnu štruktúru, ktorá slúži ako základný bod pre uloženie výsledkov extrakcie. Ak je vykonaná extrakcia pomocou knižnice TSK, do tohto adresára sa postupne vytvorí obraz extrahovaného súborového systému. Dôležitým aspektom je, že táto operácia je realizovaná s ohľadom na úspešnosť extrakcie. To znamená, že v prípade úspešnej extrakcie budú súbory vhodne usporiadané podľa usporiadania na skúmanom obraze. V prípade nástroja PhotoRec sú postupne vytvárané adresáre ako je uvedené v 2.3. Podľa dokumentácie programu PhotoRec je potrebné definovať príkazy a nastavenia, ktoré sa majú odoslať do modulu. Tieto príkazy môžu obsahovať rôzne možnosti konfigurácie typov súborov na obnovu a ďalšie parametre. Každá z tried implementuje metódou na spustenie extrakcie, ktorá dodržiava zadané parametre a vykonáva požadovanú funkcionalitu extrakčného nástroja (buď TSK alebo PhotoRec).

4.2 Návrh implementácie modulu PhotoRec

Pre zobrazenie navrhnutej štruktúry spúšťania extrakcie nad jednotlivými vstupmi slúži nasledujúci diagram tried na obrázku 4.1. Diagram tried a vzťahov je dôležitým nástrojom v procese vývoja softvéru. Poskytuje prehľad o štruktúre programu a interakciách medzi jednotlivými komponentami. Pomáha v identifikácii hlavných častí systému a ich vzájomných vzťahov, čím znižuje zložitosť implementácie a zvyšuje prehľadnosť.



Obr. 4.1: Diagram tried extraktora

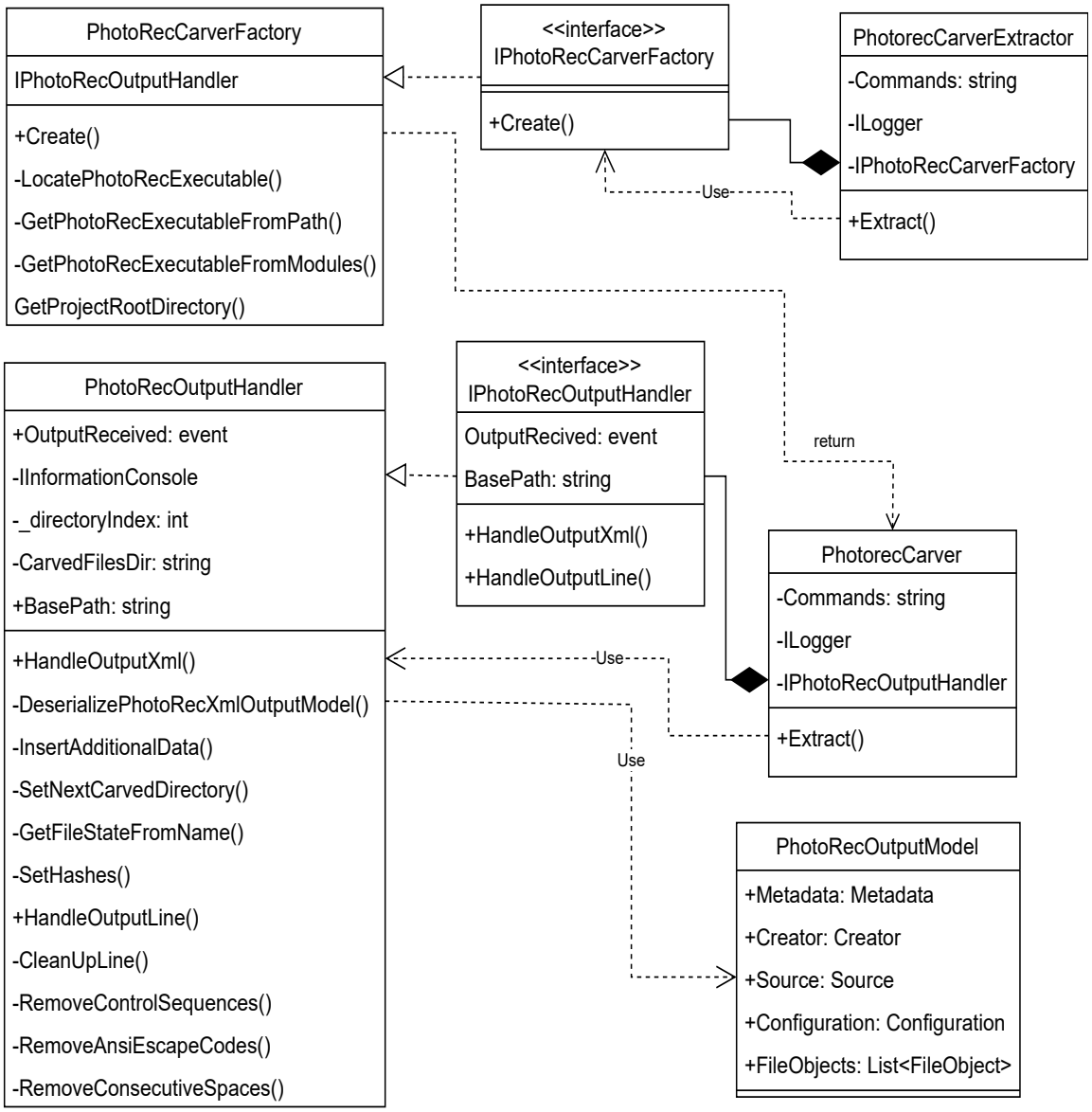
V návrhu je zobrazená štruktúra tried a vzťahy medzi nimi, ktoré sú potrebné na implementáciu v nástroji Maxtor. Diagram obsahuje nasledovné komponenty:

- `ArtifactExtractorBase` (Abstraktná trieda) — Obsahuje základnú implementáciu pre extrakciu artefaktov.Poskytuje metódu `Extract()`, ktorú musia implementovať odvodené triedy.
- `ArtifactsExtractor` (Konkrétna trieda) — Implementuje extrakciu artefaktov pomocou knižnice TSK a rozhraní. Obsahuje metódu `Extract()`, ktorá riadi extrakciu artefaktov.
- `PhotoRecCarverExtractor` (Konkrétna trieda) — Špecifická trieda pre extrakciu artefaktov pomocou modulu PhotoRec. Implementuje metódu `Extract()`, ktorá vykonáva extrakciu s využitím nástroja PhotoRec.
- `Executor` (Hlavná trieda) — Riadi celý proces extrakcie artefaktov. Využíva továrňu `ArtifactsExtractorFactory` na vytvorenie konkrétnych extraktorov. Obsahuje metódu `Execute()`, ktorá spúšťa extrakciu artefaktov.
- `IArtifactsExtractorFactory` (Rozhranie) — Sprístupňuje metódy pre vytvorenie konkrétnych inštancii extraktorov artefaktov.
- `ArtifactsExtractorFactory` (Továreň) — Implementuje rozhranie `IArtifactsExtractorFactory`. Slúži na vytvorenie konkrétnych inštancii extraktorov artefaktov.

Návrh programovej implementácie modulu PhotoRec poskytuje základné usmernenia pre organizáciu tried a komponentov, ktoré budú použité pri integrácii modulu do nástroja Maxtor. Štruktúra a organizácia je znázornená diagramom tried na obrázku 4.2.

Prvým kľúčovým komponentom je trieda `PhotoRecCarverFactory`, ktorá plní funkciu továrne pre vytváranie inštancií objektov. Táto trieda abstrahuje proces vytvárania inštancií a zabezpečuje ich základnú konfiguráciu. Jedna z jej metód je nutná pre lokalizáciu spustiteľného súboru programu PhotoRec. Nájdená cesta sa následne predá inštancii pri vytváraní. Použitie továrne je v súlade s princípom injekcie závislostí a po spustení programu sa jej inštancia zaregistruje do kontajnera pre správu závislostí, čo umožňuje prístup k nej počas behu programu.

Ďalším dôležitým komponentom je trieda `PhotoRecCarver`, ktorá je určená na riadenie samotného behu modulu PhotoRec. Táto trieda vytvára premostenie medzi nástrojom Maxtor a modulom PhotoRec a spravuje proces extrakcie. Extrakcia artefaktov prebieha v samostatne vytvorenom procese, pričom výstupné informácie o počte nájdených a extrahovaných súborov sú priebežne zaznamenávané a zobrazované na štandardný výstup procesu. Cieľom týchto informácií je užívateľa pravidelne informovať o priebehu extrakcie. Po dokončení extrakcie je potrebné dodatočne spracovať výstupný súbor obsahujúci serializované dáta, ktoré sa prevádzajú na formát JSON s cieľom zachovať konzistenciu výstupu nástroja Maxtor.



Obr. 4.2: Diagram tried modulu PhotoRec

Kapitola 5

Implementácia

Pri implementácii softvérových systémov je kľúčové zabezpečiť, aby bol kód ľahko udržateľný a čitateľný. Jedným z hlavných cieľov je dosiahnutie jasného a zrozumiteľného pomenovania, jednoduchých funkcií a oddelenia zodpovedností. Tieto aspekty zabezpečujú, že kód je nielen ľahko pochopiteľný, ale aj menej náchylný na chyby a jednoduchší na úpravy v budúcnosti. Preto som sa pri implementácii riadil zásadami podľa knihy *Clean Code* [16].

Prvá sekcia 5.1 tejto kapitoly sa zameriava na návrh a implementáciu vstupného rozhrania nástroja Maxtor. Toto rozhranie je navrhnuté tak, aby bolo modulárne a rozšíriteľné, umožňujúc jednoduché pridávanie nových funkcií a modulov. Ďalšia sekcia 5.2 sa zaoberá implementáciou triedy `Executor` a jeho rozhrania. `Executor` je kľúčovým komponentom nástroja Maxtor, ktorý riadi proces extrakcie artefaktov. Táto časť podrobne popisuje štruktúru triedy `Executor`, jeho rozhranie a implementáciu. Posledna sekcia 5.3 tejto kapitoly sa zameriava na implementáciu triedy `Extractor` a jej rozhraní. Táto trieda poskytuje spôsob definovania extrakčných funkcií a ich implementácie. Rozhranie `IExtractor` umožňuje jednotný spôsob definovania extrakcie artefaktov, ktorý je nezávislý na konkrétnej implementácii extraktora.

5.1 Vstupné rozhranie nástroja Maxtor

Vstupné rozhranie nástroja Maxtor je implementované formou rozhrania príkazového riadka poskytujúceho užívateľom prístup k rôznym funkcionalitám nástroja prostredníctvom príkazov v konzole. Toto rozhranie je navrhnuté tak, aby bolo modulárne a jednoducho rozšíriteľné, čo umožní pridávanie nových funkcií a modulov bez nutnosti zásahu do existujúcich štruktúr. Rozhranie som implementoval s dôrazom na a rozšíriteľnosť architektúry, ktorá má za úlohu poskytnúť automatizovanú extrakciu artefaktov z obrazov pevného disku alebo konkrétneho diskového oddielu. Táto časť vysvetľuje štruktúru a funkcie implementovaného rozhrania. Spracovanie argumentov príkazového riadka prebieha pomocou NuGet balíčku `McMaster.Extensions.CommandLineUtils`, ktorý je optimalizovaný pre spracovanie argumentov príkazového riadka. Tento balíček poskytuje vhodné rozhranie na definovanie príkazov a parametrov, manažment vstupných argumentov a ich spracovanie. S podporou tejto modulárnej a rozšíriteľnej architektúry je možné jednoducho rozširovať funkcionalitu nástroja Maxtor a pridávať nové príkazy a parametre podľa potreby. Rozhranie som implementoval podľa návrhu s využitím abstraktnej triedy `CommandBase` ktorá obsahuje základné parametre a funkcionalitu pre spúšťanie nástroja. Od abstraktnej triedy `CommandBase` dedia triedy `Cli` a `PhotoRecSubCommand`.

Trieda `CommandBase` zapúzdruje nasledujúce kľúčové artibúty a metódy:

- **Výstupný adresár** — Konfigurovateľný parameter (argument `--OutputDirectory`), ktorý špecifikuje cestu k adresáru v ktorom sú uložené extrahované artefakty. Štandardne sa artefakty ukladajú do adresára `/tmp/maxtor`.
- **Vstupné cesty** — Pole reťazcov (vstupov) reprezentujúcich cestu k obrazom diskov určených na extrakciu artefaktov alebo štandardný vstup (STDIN).
- **Metóda `HandleInputAndExecute()`** — riadi tok vykonávania na základe typu vstupu. Deleguje vykonávanie na príslušnú metódu rozhrania `Executor`, buď čítanie zo štandardného vstupu alebo vykonávanie so špecifikovanými vstupnými cestami.
- **Metóda `IsSTDINinput()`** — určuje, či je vstupný zdroj štandardný vstup, čo umožňuje dynamickú kontrolu nad extrakciou artefaktov.

Trieda `Cli` plní úlohu vstupného bodu pre interakciu užívateľa s nástrojom za použitia knižnice TSK. Trieda `PhotoRecSubCommand` zaisťuje spúšťanie a konfiguráciu modulu PhotoRec. Pri implementácii som kládol dôraz hlavne na konzistenciu s rozhraním samotného nástroja PhotoRec ako je uvedené v návrhu 2.3.

- **Konfiguračné príkazy** — Konfigurovateľný parameter (argument `--cmd`) umožňuje užívateľom špecifikovať vlastné príkazy a nastavenie pre PhotoRec, čím sa uľahčí flexibilita pri extrakcii artefaktov.
- **Metóda `OnExecute()`** — nakonfiguruje triedu `Executor` na používanie nástroja PhotoRec a nastaví akékoľvek špecifikované príkazy. Následne deleguje vykonávanie na spoločnú obsluhu vykonávania zdedenú z triedy `CommandBase`.

Rozhranie poskytované touto implementáciou ponúka jednoduchú platformu na automatizovanú extrakciu artefaktov zo súborov s obrazom disku. Prostredníctvom zapuzdrenia funkcií v rámci abstraktných tried a špecializovaných tried príkazov dosahuje implementácia ľahšiu rozšíriteľnosť o prípadné nové moduly.

5.2 Implementácia triedy `Executor` a jej rozhrania

Rozhranie `Executor` spolu s triedou `Executor` tvorí základnú štruktúru v rámci nástroja Maxtor, ktorá sa zaoberá riadením procesu extrakcie artefaktov. Rozhranie deklaruje metódy pomocou ktorých prebieha komunikácia s triedou `Executor`. Jeho použitie je dôležité z hľadiska oddelenia závislosti na konkrétnej triede a umožňuje registráciu do kontajnera závislostí. Trieda `Executor` sprístupňuje nasledujúce metódy:

- `UsePhotoRec` — metódy pre nastavenie a získanie hodnoty tohoto atribútu, ktorý určí či sa má použiť na extrakciu PhotoRec.
- `PhotoRecCommands` — metódy pre nastavenie a získanie hodnoty tohoto atribútu, ktorý obsahuje konkrétne príkazy pre PhotoRec
- `Execute()` — hlavná metóda zaisťujúca spúšťanie modulov na extrakciu.
- `ReadFromSTDIN()` — metóda zaisťujúca čítanie vstupných dát zo štandardného vstupu.

Metóda `Execute` na vstupe berie ako parameter pole vstupov a výstupný adresár. Na začiatku prebehne validácia vstupov pomocou volania metódy `ValidateInputParameter()`. Nasleduje selekcia podľa typu vstupu pre každý zo vstupov. Implementácia podporuje načítavanie zo štandardného vstupu, konkrétneho súboru obrazu disku alebo spracovanie celého priečinka obsahujúceho obrazy disku. Metóda taktiež podporuje obraz disku rozdelený na viacero súborov. O samotnú extrakciu sa starajú triedy, ktoré implementujú rozhranie `IExtractor`. Inštancie týchto tried sa vytvárajú použitím továrenských tried pred samotnou extrakciou. Inštancie sú zvolené podľa toho či sa má na extrakciu použiť knižnica TSK alebo nástroj PhotoRec. Po vytvorení konkrétnej inštancie je možné zavolať jej metódu `Extract` ktorá sa postará o samotnú extrakciu.

Pre čítanie zo štandardného vstupu slúži metóda `ReadFromSTDIN()`. Metóda prijíma jediný parameter `outputDirectory`, do ktorého budú uložené extrahované artefakty. V metóde sa vytvára inštancia serializéru pre formát JSON. Následne sa vytvára `JsonTextReader`, ktorý číta zo vstupného textu poskytnutého pomocou atribútu `_input` a nadstavením viacero obsahov.

Následne sa cyklicky číta zo vstupu, pričom sa deserializuje každá časť ako objekt typu `MaxtorToolJobBatchInput`. Ak sa nepodarí deserializovať tento objekt, vyvolá sa výnimka s informáciou o chybe.

Následne sa volá metóda `Execute` s cestami na vstupné súbory poskytnuté zo vstupných serializovaných dát a zadaným výstupným adresárom. Na konci každého cyklu sa do informačnej konzole pridávajú informácie o spracovanom dávkovom spracovaní.

V metóde `ValidateInputParameter` sa validujú vstupné parametre `inputs`. Pre každý vstupný parameter sa kontroluje, či ide o štandardný vstup, súbor alebo adresár. Ak ide o štandardný vstup, vyvolá sa chybové hlásenie pretože parameter štandardný vstup nie je povolené zadávať priamo parametrom. Pre súbory sa overuje ich prípona a pre adresáre sa kontroluje, či neobsahujú pod-adresáre. V prípade nesprávneho parametra sa vyvolá chybové hlásenie. Ak validácia zlyhá pre niektorý zo vstupov, nastaví sa príznak zlyhania validácie a na konci sa vyvolá výnimka s informáciou o zlyhaní validácie. Metóda `TryCheckInputFileExtension` kontroluje, či prípona súboru patrí medzi podporované prípony.

Rozhranie`IExecutor` a jeho príslušná implementácia predstavujú komponenty nástroja Maxtor, ktoré zabezpečujú efektívnu a všestrannú extrakciu artefaktov z obrazových súborov disku alebo zväzkov. Zapuzdrenie extrakčnej logiky do štruktúrovaného rozhrania a implementáciou príslušnej funkcionality na riadenie operácií extrakcie artefaktov zaisťuje implementácia triedy `Executor`.

5.3 Implementácia triedy Extractor a jej rozhrania

Rovnako ako pri predchádzajúcom rozhraní `IExecutor`, aj v tomto prípade som použil rozhranie, konkrétne s názvom `IExtractor`, ktoré zapúzdruje štruktúru tried zaisťujúcich extrakciu. Rozhranie deklaruje jedinú metódu `Extract()`, ktorá je následne implementovaná v jednotlivých triedach. Logickú štruktúru implementácie tried som implementoval podľa návrhu 4.2.

Toto rozhranie poskytuje jednotný spôsob, ako definovať funkcionalitu extrakcie bez toho, aby bolo viazané na konkrétnu implementáciu alebo typ extraktora. O vytváranie inštancii samotných extraktorov sa postará továrenská trieda `ArtifactsExtractorFactory`. Obsahuje dve preťažené metódy `Create()`, ktoré vrátia konkrétnu inštanciu extraktora implementujúceho rozhranie `IExtractor` na základe poskytnutých parametrov metód.

`ArtifactsExtractorBase` trieda slúži ako základná abstraktná implementácia extraktora artefaktov, ktorá poskytuje spoločnú funkcionalitu pre všetky konkrétne extraktory. Obsahuje abstraktnú metódu `Extract()`, ktorá deklaruje rozhranie metódy určenej k extrakcii artefaktov z obrazov disku alebo zväzkov. Samotná implementácia sa nachádza v triedach, ktoré dedia od triedy `ArtifactsExtractorBase` a je rozdielna pre jednotlivé typy extrakcie. Trieda tiež obsahuje pomocnú metódu na získanie informácií o vstupných diskových obrazoch.

`ArtifactsExtractor` trieda je konkrétnou implementáciou extraktora artefaktov, ktorá rozširuje základnú funkcionalitu definovanú v triede `ArtifactsExtractorBase`. Implementuje metódu `Extract`, ktorá zahŕňa logiku pre spracovanie jednotlivých obrazov disku a ich extrakciu artefaktov. Táto metóda má na starosti vykonanie procesu extrakcie artefaktov z obrazu disku na základe knižnice TSK. Najprv sa zaznamenáva informácia o spustení extrakcie pomocou rozhrania `ILogger`, kde sa zobrazuje zoznam vstupných súborov z ktorých sa má extrakcia vykonať. Ďalej sa overuje existencia vstupného súboru. Následne sa vytvorí inštancia `DiskImage` na základe zadaných vstupných súborov. Ak sa pokúšame otvoriť systém objemu pomocou metódy `TryOpenVolumeSystem`, prechádzame všetkými diskovými oddielmi v systéme, získavame informácie o každom objeme a prechádzame do spracovania súborov prostredníctvom metódy `ProcessFiles`. V prípade neúspechu otvorenia oddielu, sa pokúšame priamo otvoriť súborový systém obrazu disku s autodetekciou typu súborového systému. Ak sa otvorenie úspešne vykoná, prechádzame na spracovanie súborov rovnakým spôsobom. Na samotné spracovanie súboru slúži metóda `ProcessFile`, ktorá pomocou rozhrania a štruktúr knižnice TSK spracováva jednotlivé nájdené súbory. Aby bolo možné určiť typ súboru, vykonáva sa validácia prípon pomocou signatúr jednotlivých typov súborov.

`PhotoRecCarverExtractor` trieda je špecifickou implementáciou extraktora artefaktov, ktorá využíva nástroj PhotoRec na vyhľadávanie a obnovu multimediálnych súborov z obrazov disku. Táto trieda tiež dedí funkcionalitu z triedy `ArtifactsExtractorBase` a implementuje metódu `Extract`, ktorá spúšťa proces vyhľadávania pomocou nástroja PhotoRec pre každý vstupný súbor. Pomocou továrenskej triedy sa vytvára inštancia objektu `PhotoRecCarver` ktorej sa predá konfigurácia, čiže cesta k vstupnému súboru, príkazy a filtre pre PhotoRec a výstupný adresár. Pri vytváraní inštancie sa vyhľadá cesta k spustiteľnému súboru nástroja. Pre Operačný systém Windows nástroj Maxtor obsahuje balíček so spustiteľným súborom a v prípade platformy Linux sa predpokladá existencia odkazu v systémovej premennej PATH. Implementácia sa nachádza v triede `PhotoRecCarver`. Pred spustením procesu sa inicializujú všetky potrebné parametre a presmeruje sa štandardný a chybový výstup zo spúšťaného procesu. Spracovanie štandardného výstupu prebieha pomocou obslužnej metódy filtrujúcej výstupné reťazce.

5.4 Spracovanie výstupu modulu PhotoRec

Spracovanie výstupu prebieha v triede `PhotoRecOutputHandler`, kde sa asynchrónne vykonáva v reakcii na udalosti vyvolané procesom v prípade, že na štandardný výstup nástroja sú poslané výstupné dáta. Užívateľ je tak informovaný o nájdených artefaktoch počas priebehu extrakcie artefaktov. Keďže výstup programu PhotoRec obsahuje množstvo znakov, ktoré zabezpečujú jeho konzolové rozhranie, musí byť pre ďalšie použite očistené. Očistenie od zbytočných znakov bolo možné implementovať pomocou jednoduchých regulárnych výrazov. Užívateľ je následne upozornený o aktuálnom priebehu extrakcie. Po úspešnom priebehu, PhotoRec exportuje nájdené artefakty do výstupných adresárov. V prvom z nich vytvorí súbor s názvom `report.xml` obsahujúci dôležité forenzné informácie o nájdených

artefaktoch. Tento súbor je následne deserializovaný a pridávajú sa dodatočné informácie o každom nájdenom súbore. Každý záznam výstupu reprezentuje objekt typu `FileObject`, ktorý obsahuje nasledujúce informácie:

- **FileName** — Názov súboru, ktorý bol nájdený.
- **FileExt** — Prípona súboru.
- **FileStatus** — Stav súboru, či bol obnovený, poškodený alebo obsahuje vložený náhľad v JPEG formáte.
- **FileSize** — Veľkosť súboru v bajtoch.
- **Md5Hash** — MD5 heš súboru.
- **ShaHash** — SHA256 heš súboru.
- **ByteRun** — Objekt, ktorý obsahuje informácie popísané v sekcii 2.3

Celá štruktúra je ďalej serializovaná do formátu JSON a vypísaná do výstupnej konzole. Tieto informácie poskytujú podrobný pohľad na nájdené artefakty a umožňujú ďalšiu analýzu a spracovanie digitálnych dát získaných pomocou nástroja PhotoRec.

Kapitola 6

Testovanie

V tejto kapitole sa nachádza návrh testovacích scenárov popísaných v sekcii 6.1, ktoré som vytvoril pomocou nástrojov dostupných na platformách FreeBSD, Linux a MacOS. Tieto nástroje sú spúšťané pomocou skriptu popísaného v sekcii 3.8. Za návrhom testovacích scenárov nasledujú sekcie obsahujúce analýzu výstupov z nástroja Maxtor spúšťaného nad vytvorenej dátovej sade. Analýzu som rozdelil do dvoch logických častí podľa typu extraktora, ktorý som spúšťal nad testovacími dátami. V sekcii 6.2 sa venujem výstupom testovania nástroja Maxtor s konfiguráciou TSK a výstupom, ktoré vznikli použitím tejto knižnice. Popísané sú jednotlivé scenáre, chovanie nástroja počas extrakcie a množstvo extrahovaných súborov. V sekcii 6.3 analyzujem výsledky testovacích scenárov s použitím extraktoru za použitia modulu *PhotoRec Carver*.

6.1 Spracovanie poškodení súborových systémov

V tejto sekcii sa nachádza popis jednotlivých scenárov použitých pri testovaní obnovy dát z rôznych súborových systémov. Návrh testovacích scenárov je kritickým krokom pri overovaní efektivity forenzných nástrojov a techník na získanie dát z digitálnych zariadení. Pri ich tvorbe som čerpal z viacerých zdrojov, ktoré sa zaoberali najmä problematikou anti-forenzných postupov. Jedným z týchto zdrojov bol článok [2], ktorý sa venoval testovaniu forenzných nástrojov na dátových sadách vytvorených s použitím anti-forenzných techník. Hoci tento článok neponúkol konkrétne postupy na tvorbu testovacích scenárov, poskytol základné informácie a podnety o možných situáciách a scenároch, ktoré sa dajú simulovať.

Ďalším zdrojom informácií bol článok zaoberajúci sa oblasťou anti-forenzných techník [13]. Tento článok poskytol hlbší pohľad na rôzne metódy a stratégie, ktoré sa používajú na skrytie alebo manipuláciu s digitálnymi dátami tak, aby bolo obtiažne ich získať pomocou rôznych forenzných nástrojov. Tieto informácie som využil pri navrhovaní scenárov, ktoré by simulovali rôzne anti-forenzné postupy a situácie, ktoré môžu nastať pri analýze digitálnych médií vrátane ukrývania dát mimo dosah súborového systému alebo niekoľko násobného prepísania. Každý scenár som navrhol tak, aby simuloval konkrétnu situáciu, ktorá môže nastať pri mazaní alebo manipulácii so súbormi v súborovom systéme. Popisuje sa postup a očakávaný výsledok po vykonaní každého scenáru. V prípade knižnice TSK som pre referenciu použil vyššie popísaný nástroj *Autopsy*, ktorý túto knižnicu používa. Výstup modulu PhotoRec by mal byť v každom scenári rovnaký ako pri použití nástroja samotného.

Prvým scenárom je štandardné mazanie súborov bez akéhokoľvek prepísania náhodnými znakmi. Scenár predstavuje postup mazania súborov pomocou nástroja `rm` v prostredí použitého virtuálneho stroja.

V tomto scenári sa najprv kopírujú testovacie súbory zo zdrojového adresára na pripojený obraz disku v priečinku `/mnt/img`. Ako cieľové miesto pre presunutie súborov určených na odstránenie slúži priečinok `delete`. Následne je všetok obsah priečinku vymazaný pomocou príkazu `rm -rf`. Tento scenár predstavuje bežný spôsob odstránenia súborov zo súborového systému. Očakávaným výstupom po vykonaní tohoto scenára by v oboch prípadoch (PhotoRec aj TSK) nástroje mali úspešne lokalizovať a obnoviť tieto zmazané súbory.

Druhým scenárom je bezpečné mazanie súborov. Bezpečné mazanie zahŕňa dodatočné prepísanie obsahu súboru. Na tento účel som zvolil nástroj `shred` dostupný na unixových platformách. Podobne, ako v predchádzajúcom scenári sa najprv kopírujú súbory určené na odstránenie na obraz disku. Následne sú pomocou príkazu `shred -u -n /mnt/img/*` prepísané (základný počet prepísaní je 3) a bezpečne odstránené. V tomto prípade neočakávam úplné obnovenie súborov v oboch prípadoch použitých nástrojov, avšak v prípade knižnice TSK očakávam výskyt určitej formy stopy po odstránení.

Tretím scenárom je jednoduchá fragmentácia súboru. Scenár simuluje vytvorenie fragmentovaného súboru z existujúceho súboru uloženého v súborovom systéme. Počas vykonávania tejto akcie sa rozdelí pôvodný súbor na dva fragmenty pomocou programu `split`, ktoré sa následne uložia do dvoch rôznych zložiek. Nakoniec sa tieto fragmenty zjednotia čo vytvorí fragmentovaný súbor. Po vykonaní tohto scenáru očakávam, že po obnove bude vytvorený fragmentovaný súbor obsahovať údaje z pôvodného súboru, ktorý bol rozdelený na dva fragmenty a následne zjednotený spolu s jednotlivými fragmentami.

Štvrtý scenár podobne simuluje vytvorenie fragmentovaného súboru z existujúceho súboru v systéme. Avšak pôvodný súbor je rozdelený na viacero fragmentov, ktoré sú následne umiestnené do rôznych adresárov a pod-adresárov. Nakoniec sú tieto fragmenty, podobne ako v predchádzajúcom scenári, zjednotené do jedného fragmentovaného súboru. Scenár začína výberom prvého súboru zo zoznamu súborov v aktuálnej zložke. Následne sa súbor rozdelí na desať fragmentov pomocou príkazu `split`. Pre prvých 5 fragmentov sa v aktuálnom adresári vytvoria ďalšie adresáre, pričom niektoré fragmenty sa presunú do týchto adresárov. Súčasne sú v jednom z nich vytvorené aj pod-adresáre, kam sa presúvajú zostávajúce fragmenty. Nakoniec sa všetky fragmenty zjednotia do jedného fragmentovaného. Po vykonaní tohto scenáru sa očakáva, že vytvorený fragmentovaný súbor bude obsahovať údaje z pôvodného súboru, ktorý bol rozdelený na desať fragmentov a následne zjednotený.

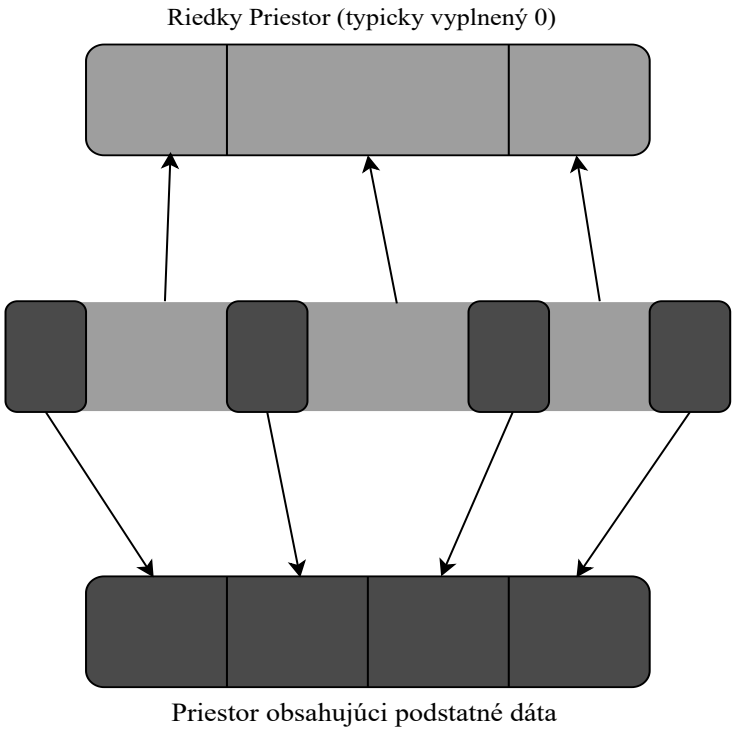
Ďalší scenár tvorí vytvorenie hlbokého adresára a následné vymazanie celého adresára. Počas tohto procesu sa vytvára hlboká štruktúra adresárov, vytvára sa súbor v najhlbšom adresári a potom sa celý adresárový strom odstráni. Postup tvorí vytvorenie hlbokého adresárového stromu pomocou príkazu `mkdir -p`. Tento príkaz vytvára adresárovú štruktúru s viacerými úrovňami hĺbky. Následne sa presunú súbory do najhlbšieho adresára pomocou príkazu `mv`, ktorý presunie všetky súbory zo súčasného adresára do zadaného adresára. Nakoniec sa celý adresárový strom odstráni pomocou príkazu `rm -rf`. Predpokladaným výsledkom po obnove pomocou nástrojov Maxtor a PhotoRec je, že bude možné obnoviť súbory, ktorý boli vytvorené v najhlbšom adresári. Napriek tomu, že adresárová štruktúra bola odstránená, nástroje na obnovu údajov by mali byť schopné nájsť a obnoviť tieto

súbory. V prípade nástroja PhotoRec sa očakáva obnovenie všetkých nástrojom podporovaných súborov bez zachovania adresárovej štruktúry.

Predposledný scenár tvorí obnova zmazaného súbory ktorého obsah bol prepísaný iným súborom. Postup vytvorenia súboru sa skladá z vybrania dvoch súborov, kde sa obsahom jedného prepíše obsah druhého pomocou príkazu `cat`. a následne sa tento súbor odstráni.

Posledný scenár simuluje vytvorenie veľkého riedkeho súboru, ktorý má veľký logický rozsah no na disku zaberá pomerne málo miesta keďže väčšina alokovaných štruktúr odkazuje na nulové hodnoty. Do súboru bol zapísaný opakujúci sa vzor núl. Scenár sa týka najmä súborových systémov, ktoré podporujú tento typ súboru. z použitých sú to systémy skupiny Ext 3.3, UFS 3.4, ExFAT 3.2.

Scenár začína vytvorením veľkého súboru s názvom `large_file.txt` pomocou príkazu



Obr. 6.1: Štruktúra riedkeho súboru

`truncate -s`. Veľkosť tohto súboru je hodnota nachádzajúca sa pod hranicou maximálnej veľkosti súboru, ktorý vedia jednotlivé súborové systémy alokovať pre jeden súbor. Štruktúru riedkeho súboru možno vidieť na obrázku 6.1. Systému ExFAT som ponechal veľkosť ako pri predchádzajúcich aj napriek tomu, že je možné vytvoriť omnoho väčší súbor. Logické veľkosti sú nasledovné:

- Ext2, Ext3, UFS1 — 1TB
- Ext4, UFS2 — 15TB
- ExFAT — 15TB

Nasleduje zápis opakujúceho sa vzoru núl do vytvoreného súboru pomocou príkazu `dd` a `tr`. V prípade tohoto scenáru bude testovaná schopnosť nástrojov pracovať s veľkými logickými súbormi, ktoré dosahujú veľkosti niekoľkých terabajtov, čo môže predstavovať výzvu pre nástroje na spracovanie tohoto typu súborov. Tieto veľké logické súbory môžu viesť k nadmernému zaťaženiu pamäte a zdrojov procesora, čo môže spôsobiť pád nástrojov alebo výrazné spomalenie ich výkonu. V takom prípade sa stáva analýza daného média zložitejšia a môže nastať situácia kedy nebudú obnovené všetky dáta čo predstavuje problém v oblasti digitálnej forenznej analýzy. Predpokladám, že v prípade knižnice TSK môžu nastať problémy pri otváraní súboru keďže sa spolieha na metadáta daného súboru. Nástroj PhotoRec by mal byť schopný sa s daným súborom vysporiadať bez problémov keďže z podstaty uvedenej v sekcii 2.3, sa nespolieha na štruktúry súborového systému.

Tabuľka 6.1: Popis jednotlivých scenárov

Názov scenára	**Popis**	**Predpokladaný Výsledok**
DeletedFiles	Bežné Odstránenie súborov z obrazu	Úspešná obnova súborov
ShrededFiles	Bezpečné odstránenie súborov z obrazu	Nástroj naznačí možné mazanie
FragmentedFile	Fragmentácia súboru	Úspešná extrakcia fragmentovaného súboru
DeepDirectoryStructure	Odstránenie hlbokého adresára.	Úspešná obnova súborov z hlbokého adresára
LargeFile	Vytvorenie veľkého logického súboru.	Nástroj PhotoRec sa vysporiada s veľkým logickým súborom, TSK môže mať problém.

V závere tejto časti sa nachádza tabuľka 6.1 s prehľadom jednotlivých scenárov a očakávaných výsledkov.

6.2 Testovanie nástroja Maxtor s konfiguráciou TSK

Táto časť prezentuje výsledky integračného testovania nástroja na vytvorenej dátovej sade. Nástroj bol konfigurovaný na použitie extraktora založeného na knižnici TSK (*The Sleuth Kit*). V prípade systému APFS bol nástroj schopný tento systém načítať, no nebol schopný sa dostať k samotnej adresátovej štruktúre a teda aj k zmazaným súborom. Súborový systém HFS sa nepodarilo načítať. Toto chovanie pri súborových systémoch APFS a HFS som zaznamenal vo všetkých scenároch. Nasleduje rozbor výsledkov testovania jednotlivých scenárov na rôznych súborových systémoch, ktoré sa podarilo úspešne spracovať.

Prvým testovacím scenárom boli štandardne zmazané súbory. Nástroj dokázal úspešne obnoviť všetky súbory v systémoch NTFS a všetkých súborových systémoch skupiny FAT. Pri súborových systémoch Ext2/3/4 program korektne ukončil svoju činnosť avšak neboli obnovené žiadne zo zmazaných súborov. Výsledok som porovnával s nástrojom *Autopsy* ktorý tiež nebol schopný obnoviť tieto súbory. Rovnako to bolo v prípade UFS1/2.

V prípade bezpečného mazania pomocou programu `shred` sa nepodarilo obnoviť zmazané súbory. Podľa výstupu však program narazil na objekty, ktoré môžu naznačovať, že k podobnému mazaniu došlo. V súborových systémoch NTFS a ExFat boli v štruktúrach zachované informácie o tom, že sa tam v minulosti nachádzali zmazané súbory (počet týchto objektov odpovedal počtu testovacích súborov). Nepodarilo sa obnoviť ani cestu k nim a ich

veľkosť bola nadstavená na nulu, tým pádom boli programom preskočené. V súborových systémoch Ext2/3/4, boli rovnako objavené podobné súbory. Tieto súbory boli zaradené do skupiny súborov bez vlastníka. Sú to zmazané súbory, ktoré stále obsahujú metadáta súborového systému, ale ku ktorým nie je možné pristupovať z koreňového adresára. Vo väčšine súborových systémov sú metadáta súboru (napríklad časy a pridelené bloky) uložené na odlišnom mieste než názov súboru. Názov odkazuje na miesto metadát [5]. Je možné, že názov zmazaného súboru sa vymaže alebo znovu použije, ale metadáta súboru stále existujú. Od verzie TSK 3 sú tieto súbory exportované do adresára `$OrphanFiles`. Treba poznamenať, že tento adresár sa na obraze disku v skutočnosti neexistuje, je to iba virtuálny spôsob, ako TSK poskytuje prístup k metadátam, ktoré existujú.Nástroj Maxtor tieto súbory zaradil do preskočených a teda neboli obnovené do výstupného adresára.

Nasledoval scenár obsahujúci ľahko fragmentovaný súbor typu `.jpg`. V priebehu extrakcie bol úspešne obnovený celý fragmentovaný súbor, vrátane jeho prvej časti. Avšak, ďalšie oddelené fragmenty neprešli validáciou ich signatúry, čo malo za následok ich preskočenie a teda neboli obnovené. V prípade situácie s viac fragmentovaným súborom sa nástroj zachoval obdobným spôsobom kedy opäť bola extrahovaná aj prvá časť súboru. V oboch scenároch bol výsledok extrahovaných fragmentovaných súborov rovnaký pre všetky podporované súborové systémy.

Scenár kde sa nachádzal súbor prepísaný iným súborom sa v prípade súborového systému NTFS podarilo obnoviť súbor avšak iba s obsahom ktorým bol pôvodný súbor prepísaný a s nesprávnou príponou. V prípade Ext2/3/4 sa nepodarilo pôvodný súbor obnoviť, rovnako sa nepodarilo obnoviť ani prepísaný súbor. V prípade systémov UFS1 a UFS2 neboli obnovené ani pôvodné ani prepísané súbory. Scenár kde sa súbory nachádzali hlbšie v adresárovej štruktúre boli súbory obnovené v prípadoch NTFS, skupiny súborových systémov FAT a ExFAT. Naopak v prípade Ext2/3/4 a UFS1/2 sa nepodarilo obnoviť tieto zmazané súbory

V rámci priebehu scenára obsahujúceho veľký riedky súbor, program úspešne extrahoval súbory na obraze disku až do momentu kedy narazil na daný súbor. V momente keď sa nástroj pokúsil tento súbor spracovať sa extrakcia zastavila a program sa javil ako zamrznutý v procese spracovania tohto súboru. Výsledkom tohto scenára je, že extrakcia ďalších súborov z obrazu disku bola pozastavená kvôli problémom s veľkým súborom. Problém s týmto typom súboru nastal pri všetkých typoch súborových systémov ktoré ho obsahovali. Tento incident naznačuje potenciálnu oblasť vylepšenia v programe, najmä pokiaľ ide o riešenie a spracovanie veľkých a riedkych súborov, aby sa minimalizovalo riziko zastavenia extrakcie dát.

6.3 Testovanie nástroja Maxtor s konfiguráciou PhotoRec

Nástroj PhotoRec bol schopný úspešne obnoviť bežne zmazané súbory. Tento výsledok v porovnaní s extraktorom založeným na knižnici TSK dopadol lepšie, pretože nástroj PhotoRec bol úspešný pri všetkých testovaných súborových systémoch a dokázal úspešne obnoviť bežne zmazané súbory. Tento výsledok potvrdzuje efektívnosť nástroja pri obnove zmazaných dát.

Napriek tomu súbory, ktoré boli zmazané pomocou programu `shred`, nebolo možné úspešne obnoviť, čo je v súlade s očakávaným výsledkom. Pri súboroch, ktoré boli prepísané náhodným obsahom, sa nenachádzajú identifikačné hodnoty, ktoré program PhotoRec zachytáva, čo vysvetľuje neúspech obnovy týchto súborov.

Fragmentované súbory sa nástroju podarilo obnoviť v celku, ale jednotlivé fragmenty neboli obnovené. Tento výsledok je dôležitý pre pochopenie obmedzení nástroja pri obnove fragmentovaných súborov.

V scenári s prepísaným súborom sa vyskytla situácia, kde nástroj úspešne obnovil iba prepísaný obsah súboru, pričom pôvodný obsah súboru nebol obnovený. Aj keď pôvodná verzia súboru nebola úspešne obnovená, prítomnosť prepísaného obsahu naznačuje, že nástroj má schopnosť získať údaje aj v prípade, že boli čiastočne prepísané alebo modifikované. Nástroj sa tiež úspešne vysporiadal s veľkým logickým súborom, čo naznačuje jeho schopnosť pracovať s rozsiahlymi dátovými súbormi a efektívne obnovovať ich obsah.

Kapitola 7

Záver

Cieľom práce bolo analyzovať proces extrakcie multimediálnych dát z obrazov pevných diskov a zvoliť vhodný nástroj na rozšírenie extrakčného nástroja Maxtor. Tento nástroj automatizuje spracovanie vstupných obrazov disku a ukladá extrahované súbory do výstupného adresára. Okrem toho poskytuje výstupné údaje vo formáte JSON, ktorý obsahuje dôležité informácie o nájdených artefaktoch, ktoré je možné ďalej spracovať.

Na začiatku bolo potrebné naštudovať rozhranie a funkcionalitu natívnej knižnice *The Sleuth Kit* a možnosti jej integrácie do prostredia .NET a jazyku C#. V prípade tejto knižnice bola implementovaná podpora nových funkcií z jej rozhrania. Následne bol vytvorený návrh implementácie zvoleného rozšírenia nástroja, ktorý popisoval jednotlivé kroky integrácie, od základných funkcionalít až po detailné technické aspekty.

Po fáze návrhu bola postupne realizovaná integrácia tohoto nástroja. Hlavným bodom implementácie bola práca na module nástroja PhotoRec, ktorý mal za úlohu efektívne analyzovať a obnovovať stratené dáta z rôznych typov súborových systémov a poškodených médií pomocou metódy vyrezávania dát.

Po dokončení implementácie nasledoval dôkladný testovací proces. Celý nástroj Maxtor bol testovaný na vytvorenej dátovej sade pomocou jednotkových a integračných testov. Testy zahŕňali rôzne scenáre, aby sa overila spoľahlivosť a účinnosť nástroja v rôznych podmienkach a prostrediach.

Integračné testovanie knižnice TSK ukázalo schopnosť úspešne extrahovať dáta z obrazov pevných diskov, avšak tiež odhalilo isté nedostatky. Konkrétne, nástroj mal problémy s veľkými logickými súbormi a v niektorých prípadoch nedokázal obnoviť niektoré súbory. Na druhej strane, testovanie modulu PhotoRec preukázalo úspešnú obnovu dát bez ohľadu na použitý súborový systém, pokiaľ nedošlo k prepísaniu pôvodných dát. Tieto výsledky poskytujú dôležitý základ pre ďalšie vylepšenia a optimalizácie nástroja s cieľom zlepšiť jeho výkonnosť a spoľahlivosť v rôznych situáciách a prostrediach. Do budúcnosti sa v prípade TSK otvára možnosť riešiť problém s veľkými logickými súbormi, čo by predišlo potenciálnym ťažkostiam pri extrakcii. Ďalej je potrebné zvážiť pridanie podpory pre niektoré súborové systémy, s ktorými nástroj momentálne nedokáže efektívne pracovať. Taktiež je možné rozšíriť spektrum podporovaných typov súborov, ktoré nástroj dokáže extrahovať pomocou tejto knižnice. Tieto kroky by umožnili lepšiu adaptáciu a výkonnosť nástroja v širšom spektre situácií a prostredí, čo by prispelo k jeho celkovej účinnosti a využiteľnosti pre používateľov.

Literatúra

[1] Apple. *HFS Plus Volume Format* [online]. Apple Inc., marec 2004 [cit. 2024-1-07]. Dostupné z: https://developer.apple.com/library/archive/technotes/tn/tn1150.html.

[2] Bhat, W. A., AlZahrani, A. a Wani, M. A. Can computer forensic tools be trusted in digital investigations? *Science & Justice*. 1. vyd. 2021, zv. 61, č. 2, s. 198–203. ISSN 1355-0306. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1355030620303002.

[3] Borg, M. *C++/C# interoperability* [online]. Rev. 14. September 2017 [cit. 2024-4-22]. Dostupné z: https://mark-borg.github.io/blog/2017/interop/.

[4] Carrier, B. *File System Forensic Analysis*. 1. vyd. Addison Wesley Professional, 2005. ISBN 0-32-126817-2.

[5] Carrier, B. *SleuthKitWiki* [online]. 2014 [cit. 2024-1-07]. Dostupné z: https://wiki.sleuthkit.org/index.php?title=The_Sleuth_Kit.

[6] Carrier, B. *Autopsy User Documentation* [online]. 2016 [cit. 2023-12-26]. Dostupné z: https://sleuthkit.org/autopsy/docs/user-docs/4.1/index.html.

[7] Carrier, B. *The Sleuth Kit (TSK) Library User's Guide and API Reference* [online]. 2024 [cit. 2024-1-07]. Dostupné z: https://www.sleuthkit.org/sleuthkit/docs/api-docs/4.12.1/index.html.

[8] Fairbanks, K. D. An analysis of Ext4 for digital forensics. *Digital Investigation*. 2012, zv. 9, s. S118–S130. ISSN 1742-2876. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1742287612000357.

[9] GRENIER, C. *Testdisk* [online]. Rev. 22. október 2023 [cit. 2023-12-27]. Dostupné z: https://www.cgsecurity.org/testdisk_doc/.

[10] GRENIER, C. *PhotoRec, Digital Picture and File Recovery* [online]. Rev. 5. máj 2023 [cit. 2023-12-27]. Dostupné z: https://www.cgsecurity.org/wiki/PhotoRec.

[11] Hansen, K. H. a Toolan, F. Decoding the APFS file system. *Digital Investigation*. 1. vyd. 2017, zv. 22, č. 1, s. 107–132. ISSN 1742-2876. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1742287617301408.

[12] Homeland Security, U. D. of. *Computer Forensic Tool-Testing (CFTT) reports* [online]. Rev. 13. november 2023 [cit. 2024-4-16]. Dostupné z: https://www.dhs.gov/science-and-technology/nist-cftt-reports.

[13] Jain, A. a Chhabra, G. S. Anti-forensics techniques: An analytical review. In: *2014 Seventh International Conference on Contemporary Computing (IC3)*. 2014, s. 412–418. DOI: 10.1109/IC3.2014.6897209. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6897209&tag=1.

[14] Kessler, G. C. *File Signatures Table* [online]. Rev. 22. apríl 2024 [cit. 2024-4-22]. Dostupné z: https://web.archive.org/web/20240406002514/https://www.garykessler.net/library/file_sigs.html.

[15] Koritzinsky, J., Schuster, J., Kotas, J., Warren, G. a Dykstra, T. *Platform Invoke (P/Invoke)* [online]. Microsoft, september 2023 [cit. 2024-1-01]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/standard/native-interop/pinvoke.

[16] Martin, R. C. *Clean Code*. 1. vyd. Prentice Hall, 2008. ISBN 0-13-235088-2.

[17] Mathur, A., Cao, M., Bhattacharya, S., Dilger, A., Tomas, A. et al. The new ext4 filesystem: current status and future plans. In: Citeseer. *Proceedings of the Linux symposium*. 2007, sv. 2, s. 21–33.

[18] NIST, O. of Law Enforcement Standards of the. *The Sleuth Kit (TSK)3.2.2/Autopsy 2.24*. 300 7th St SW, Washington, DC 20024, Spojené štáty americké: U.S. Department of Homeland Security, júl 2014. Dostupné z: https://www.dhs.gov/sites/default/files/publications/508_Test%20Report_The%20Sleuth%20Kit%203%202%202%20-%20Autopsy%202%2024%20Test%20Report_November%202015_Final.pdf.

[19] Pontello, M. *TrID - File Identifier* [online]. Rev. 5. apríl 2024 [cit. 2024-4-22]. Dostupné z: https://web.archive.org/web/20240405074811/https://mark0.net/soft-trid-e.html.

[20] Vandermeer, Y., Le Khac, N.-A., Carthy, J. a Kechadi, T. Forensic analysis of the exfat artefacts. *ArXiv preprint arXiv:1804.08653*. 2018.

[21] Whitney, T., Sharkey, K., Jones, M., Hogenson, G. a Cai, S. *Calling Native Functions from Managed Code* [online]. Microsoft, august 2021 [cit. 2024-1-01]. Dostupné z: https://learn.microsoft.com/en-us/cpp/dotnet/calling-native-functions-from-managed-code?view=msvc-170.

Príloha A

Obsah pamäťového média

Táto príloha obsahuje zoznam obsahu priloženého pamäťového média.

Priložené pamäťové médium obsahuje nasledovné:

- `/create_dataset/`
  - **/test_files**
    - zložka do ktorej je možné umiestniť testovacie súbory ktoré sa majú preniesť na virtuálne stroje
  - **/VM_images**
    - obrazy virtuálnych strojov
  - `CreateDataset`
    - skript pre vytvorenie dátovej sady
  - `install.sh`
    - skript pre inštaláciu potrebných nástrojov
  - `README.md`
    - základné informácie o skripte
- `/Maxtor/`
  - zložka obsahujúca projekt
- `/xcsade00/`
  - Zdrojové súbory technickej správy v LaTeX.
- `xcsade00.pdf`
  - Táto technická správa vo formáte PDF.
- `README.md`
  - Sprievodný súbor týmto projektom.