VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**JNANA SANGAMA,BELAGAVI – 590018**

**KARNATAKA**



**Assignment Report**

**On**

**DEGITAL FORENSICS**

(File recovery and analysis)

**SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS**

**FOR THE DATA STRUCTURES AND APPLICATIONS (BCS304) COURSE OF III SEMESTER**

Submitted by

|  |  |  |
| --- | --- | --- |
| NAME : VIDHYA K |  |  |
| USN:1CG23CS119 |  |  |

**Guide: HOD:**

**Prof. Asif Ulla Khan .M. Tech. Dr. Shantala C P PhD.,**

Asst. Prof., Dept. of CSE Head, Dept. of CSE

CIT, Gubbi. CIT, Gubbi.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**2024-2025**

**Channabasaveshwara Institute of Technology**

(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)

(**ISO 9001:2015 Certified Institution)**

NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

**Rubric – B.E. Mini-Project [BCS304]**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Course**  **outcome** | **Rubric/Level** | **Excellent**  **(91-100%)** | **Good**  **(81-90%)** | **Average**  **(61-80%)** | **Moderate**  **(40-60%)** | **Score** |
| **CO1** | **Identification of project**  **proposal**  **(05 Marks)** |  |  |  |  |  |
| **CO2** | **Design and**  **Implementation**  **(10 Marks)** |  |  |  |  |  |
| **CO3** | **Presentation skill**  **(05 Marks)** |  |  |  |  |  |
| **CO4** | **Report**  **(05 Marks)** |  |  |  |  |  |
| **Total** | | | | | |  |

**Course outcome:**

**CO 1: Identification of project proposal which is relevant to subject of engineering.**

**CO 2: Design and implement proposed project methodology.**

**CO 3: Effective communication skill to assimilate their project work.**

**CO 4: Understanding overall project progress and performance.**

**Student Signature Faculty signature**

**ABSTRACT**

The Digital Forensics Suite (DFS) is a C-based application designed to facilitate file recovery and forensic analysis by interacting with the file system at a low level. This project aims to address the critical need for efficient tools in digital forensics investigations, focusing on data recovery, file system analysis, and evidence extraction.

The DFS provides functionalities such as recovering deleted files, identifying file signatures, examining metadata, and performing disk image analysis. By leveraging low-level file system operations, the suite ensures precise interaction with storage devices, making it suitable for analyzing raw disk partitions and other forensic artifacts. The use of C programming language ensures performance optimization and direct hardware interaction, enabling investigators to handle large datasets effectively.

The suite supports multiple file systems (e.g., FAT32, NTFS, ext4) and incorporates signature-based file recovery to locate specific file types. It also provides features like raw sector analysis for fragmented data recovery and detailed logging for documentation. The command-line interface (CLI) ensures usability while offering flexibility for customization.

DFS is designed to serve a broad audience, including forensic professionals, IT administrators, and researchers, by addressing the limitations of proprietary tools. Its efficiency, portability, and adaptability make it a valuable contribution to the digital forensics domain.

**CHAPTER 1**

**Introduction**

Digital forensics plays a crucial role in modern investigative processes, enabling the recovery and analysis of digital evidence from electronic devices. With the growing volume and sophistication of cybercrime, there is a need for efficient, reliable, and precise tools that can interact directly with storage devices and file systems to retrieve valuable information. The Digital Forensics Suite (DFS) aims to address this need by providing a robust C-based framework for performing low-level file recovery and analysis.

C programming is uniquely suited for digital forensics due to its low-level capabilities, such as direct memory access, hardware interaction, and file system manipulation. These features allow DFS to perform tasks like recovering deleted files, examining metadata, and analyzing disk images with high precision and efficiency. Additionally, its compact and performance-oriented .

Digital forensics is a critical field that involves the identification, recovery, and analysis of digital evidence from various storage devices. As digital data becomes increasingly central to investigations, efficient tools are necessary to interact with file systems and retrieve valuable information.

The Digital Forensics Suite (DFS) is a software project built using the C programming language, designed to recover and analyze files from storage devices. By leveraging low-level interactions with file systems, DFS enables users to perform essential forensic tasks, including file recovery, metadata extraction, and integrity verification

**CHAPTER 2**

**Problem Statement**

In digital forensics, one of the primary objectives is to recover lost or deleted files from a storage medium, such as hard drives, memory cards, or disk images. This project aims to develop a tool using the C programming language that can perform file recovery and analysis by interacting directly with the file system at a low level. The tool will scan raw disk images or unallocated sectors for known file signatures, recover files, and provide metadata about each recovered file.

**The challenges faced in digital forensics investigations include:**

1. Data Loss: Recovering deleted or corrupted files from various file systems without compromising the integrity of the original data.

2. File System Complexity: Supporting multiple file systems (e.g., FAT, NTFS, ext3/4) with their unique structures and challenges.

3. Efficiency and Accuracy: Analyzing large datasets quickly while maintaining accuracy, particularly in resource-constrained environments.

4. Evidence Integrity: Ensuring the authenticity and integrity of recovered data for use in legal proceedings.

5. Forensic Imaging: Processing raw disk images to extract hidden, fragmented, or partially overwritten data**.**

**CHAPTER 3**

**Implementation**

**flowchart**

Start

Open Disk Image File if fail,show error

Read a 512-byte block from disk image

Match block signature?

Yes no

Close previous output file Continue to write to file

Create new output file record metadata

Write block to file update metadata

More blocks to read?

Yes No

Go to read data block

Close all files output recovered metadata

End

**Flowchart Explanation**

1. Start: Begin the file recovery process.
2. Open Disk Image: Open the raw disk image file. If unsuccessful, exit the program with an error.
3. Read Data Blocks: Read the next 512-byte block of data from the disk image.
4. Match Signature: Check if the block starts with a known file signature.
   * Yes: Close the previous output file (if open) and create a new file. Record the metadata for the recovered file.
   * No: Proceed to the next step.
5. Write Block: If an output file is open, write the current block to it and update metadata.
6. Repeat: Continue reading and processing blocks until the end of the disk image is reached.
7. Close Files: Close any remaining open files and the disk image.
8. Output Metadata: Display the metadata for all recovered files.
9. End: The recovery process is complet

**Program**

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

#include <string.h>

#define BLOCK\_SIZE 512

#define SIGNATURE\_COUNT 3

typedef struct {

char \*signature;

char \*extension;

size\_t signature\_length;

} FileType;

typedef struct {

long start\_block;

long end\_block;

char filename[256];

size\_t size;

} RecoveredFile;

FileType supported\_types[SIGNATURE\_COUNT] = {

{"\xFF\xD8\xFF", ".jpg", 3}, // JPEG

{"\x89PNG", ".png", 4}, // PNG

{"\x25\x50\x44\x46", ".pdf", 4} // PDF

};

// Check if a buffer starts with a known file signature

int match\_signature(const uint8\_t \*buffer, FileType \*file\_type) {

for (int i = 0; i < SIGNATURE\_COUNT; i++) {

if (memcmp(buffer, supported\_types[i].signature, supported\_types[i].signature\_length) == 0) {

\*file\_type = supported\_types[i];

return 1;

}

}

return 0;

}

// Recover files from a disk image

void recover\_files(const char \*input\_file) {

FILE \*input = fopen(input\_file, "rb");

if (!input) {

perror("Error opening disk image");

return;

}

uint8\_t buffer[BLOCK\_SIZE];

FILE \*output = NULL;

int file\_count = 0;

long block\_count = 0;

RecoveredFile recovered[100];

int recovered\_index = 0;

FileType matched\_type;

while (fread(buffer, 1, BLOCK\_SIZE, input) == BLOCK\_SIZE) {

if (match\_signature(buffer, &matched\_type)) {

// Close any previously open file

if (output) fclose(output);

// Create a new file

sprintf(recovered[recovered\_index].filename, "recovered\_%03d%s", file\_count++, matched\_type.extension);

output = fopen(recovered[recovered\_index].filename, "wb");

if (!output) {

perror("Error creating output file");

fclose(input);

return;

}

// Record metadata

recovered[recovered\_index].start\_block = block\_count;

recovered[recovered\_index].size = 0;

recovered\_index++;

}

// Write to the current file

if (output) {

fwrite(buffer, 1, BLOCK\_SIZE, output);

recovered[recovered\_index - 1].size += BLOCK\_SIZE;

recovered[recovered\_index - 1].end\_block = block\_count;

}

block\_count++;

}

// Close the last file

if (output) fclose(output);

fclose(input);

// Print metadata

printf("Recovered %d files:\n", file\_count);

for (int i = 0; i < recovered\_index; i++) {

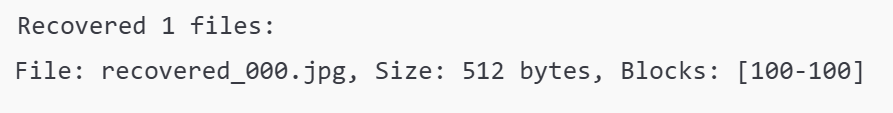
printf("File: %s, Size: %ld bytes, Blocks: [%ld-%ld]\n",

recovered[i].filename, recovered[i].size, recovered[i].start\_block, recovered[i].end\_block);

}

}

**RESULT SCREENSHORT:-**

****

**Chapter 4**

**Conclusion**

The Digital Forensics Suite (DFS) developed using the C programming language offers an efficient and scalable solution for file recovery and analysis. By leveraging low-level file system interactions and implementing core data structures and algorithms, DFS provides the necessary tools to perform critical forensic tasks. Here’s a summary of the key takeaways.

The development of DFS using C is a powerful approach to digital forensics, combining efficiency, flexibility, and low-level control over file systems. While challenges such as fragmented data recovery and large-scale data processing exist, the suite’s reliance on efficient data structures and algorithms offers a robust solution. With continued improvements, DFS can become an invaluable tool in forensic investigations, providing critical insights into digital evidence for cybersecurity, law enforcement, and legal professionals.

As an open-source project, DFS addresses the gaps left by proprietary and expensive forensic tools, making it accessible to a broader audience. Its efficiency, portability, and flexibility cater to diverse user groups, including forensic professionals, IT administrators, students, and researchers.

Through its robust features and user-friendly design, DFS empowers users to uncover critical digital evidence, contributing to the resolution of investigations, recovery of valuable data, and advancement of forensic knowledge. The Digital Forensics Suite is a step forward in creating reliable, accessible, and customizable solutions in the field of digital forensics.

**References**

1. <https://chatgpt.com/c/674f39f9-77ec-8003-8a52-46a200799ed6>
2. <https://www.w3schools.com/c/c_intro.php>
3. <https://www.programiz.com/c-programming>
4. <https://www.geeksforgeeks.org/c-programming-language/>
5. hhps://www.blackbox.ai/chat/SOpWYMu