File Recovery Tool: Final Report

1. Introduction

The **File Recovery Tool** is a C++ program designed to recover deleted or lost files from storage devices (such as hard drives, USB drives, or memory cards) by scanning raw disk data for known file signatures. This tool is particularly useful in forensic analysis, data recovery, and accidental file deletion scenarios.

Development Team:

- **Yassin Bassam** (ID: 320230141)
- **Ahmed Khaled** (ID: 320230133)
- **Omar Nasser** (ID: 320230147)
- **Mohamed Ahmed** (ID: 320230135)
- **Halim Osama** (ID: 320230144)
- **Sylvia Emad** (ID: 320230126)
- **Bavly Nagy** (ID: 320230136)

Key Features:

- Supports multiple file formats (PDF, JPG, PNG, GIF, MP3, MP4, DOCX, ZIP).
- Works on both **Windows** and **Linux**.
- Scans disk sectors directly for maximum recovery efficiency.
- Provides real-time progress tracking (scan speed, recovered files).
- Generates a recovery log for analysis.

2. Implementation

2.1 File Signature Detection

The program identifies files by matching **header** and **footer** signatures (magic numbers) against raw disk data. Each file type has a unique signature:

File Type	Header Signature	Footer Signature
PDF	`%PDF`	`%%EOF`
JPG	`FF D8 FF`	`FF D9`

File Type	Header Signature	Footer Signature
PNG	`\x89PNG`	`IEND\xAE\x42\x60\x82`
GIF	`GIF87a` or `GIF89a`	`;` (semicolon)
MP3 (ID3)	`ID3`	(None)
MP3 (MPEG)	`FF FB`	(None)
MP4	`ftypisom` or `ftypmp42`	(None)
DOCX/ZIP	`PK\x03\x04`	(None)

2.2 Drive Access

- **Windows:** Uses `CreateFile` with `GENERIC_READ` to access physical drives (`\\.\PhysicalDriveX`).
- **Linux:** Uses `open()` with `O_RDONLY` to read from `/dev/sdX` .

2.3 Recovery Process

1. **Drive Verification:** Checks if the drive is accessible and reads the first sector for validation.
2. **Buffer Scanning:** Reads disk data in **4MB chunks** for efficiency.
3. **Signature Matching:** Compares buffer data against known file headers.
4. **File Extraction:** When a match is found, the program extracts data until:
   - The **footer** is detected (if available).
   - The **maximum file size** is reached.
5. **Output:** Saves recovered files with sequential names (`file_1.pdf` , `file_2.jpg` , etc.) in the specified directory.

2.4 Logging & Progress Tracking

- A **recovery_log.txt** file records all recovered files.
- Real-time stats display:
  - **MB scanned**
  - **Scan speed (MB/s)**
  - **Number of files recovered**

3. Results

3.1 Performance

- **Scan Speed:** Typically **50-200 MB/s** (depends on drive speed and system performance).

- **Accuracy:** Successfully recovers files if:
  - The file header/footer is intact.
  - The file was not overwritten.
- **Supported Formats:** Works well for **PDF, JPG, PNG, GIF, MP3, MP4, DOCX, ZIP**.

3.2 Limitations

- **Fragmented Files:** May fail if file data is scattered.
- **Overwritten Data:** Cannot recover files if disk sectors were reused.
- **Encrypted/Compressed Files:** Only recovers raw file data (no decryption).

3.3 Sample Output

```
=== Recovery Complete ===
Scanned 2048 MB in 12.5 seconds
Average speed: 163.8 MB/s
Total files recovered: 42
Output directory: C:\recovered_files
```

4. Conclusion

This **File Recovery Tool** provides a **fast, efficient, and cross-platform** solution for retrieving lost files. It is particularly useful for forensic investigators, IT professionals, and end-users who need to recover accidentally deleted files.

Future Improvements

- Add **deep scan mode** for fragmented files.
- Support more **file formats** (e.g., EXE, AVI, SQLite).
- Implement **file carving** for better recovery of corrupted files.

5. How to Use

1. **Run as Administrator/Root** (required for direct disk access).
2. **Enter the drive path** (e.g., `\\.\PhysicalDrive1` or `/dev/sdb`).
3. **Specify output directory** (default: `recovered_files`).
4. **Wait for completion** and check the recovered files.

This tool is **open-source** and can be extended for more advanced recovery scenarios.