

Hardlink Manager

Instruction Manual

Version 0.2.0

*A cross-platform application for managing hardlink-based
file indexing and synchronization across multiple directories.*

Table of Contents

Motivation	1. Preface
Potential Uses	
About This Manual	
	2. Unders
What Are Hardlinks?	
Why Hardlinks?	
Constraints and Limitations	
	3. Installa
System Requirements	
Installing from Source	
Building a Standalone Executable	
	4. Getting
Launching the Application	
The Main Window	
Navigating the Interface	
	5. File Bro
Directory Tree	
File List and Tabs	
File Operations	
	6. Hardlin
Creating a Hardlink	
Viewing Hardlinks	
Deleting a Hardlink	
	7. Mirror C
Concept Overview	
Creating a Mirror Group	
Managing Mirror Groups	
Automatic Synchronization	
Scanning for Existing Mirrors	
	8. Intersec
Running a Search	

Interpreting Results

Scholarly Archive with Multilingual Names

Thematic Cross-Referencing

Periodical with Alternate Titles

9. Keyboa

10. Config

11. Examp

12. Troubl

13. Appen

1. Preface

1.1 Motivation

Managing large collections of documents—whether scholarly archives, research libraries, media collections, or institutional records—presents a persistent organizational challenge. A single document often belongs logically in multiple locations: a paper on Byzantine Christology might be filed under theology, under Byzantine history, and under the name of its author. Traditional file management forces a choice: store the file in one location, or duplicate it across several.

Duplication wastes disk space and creates a synchronization nightmare. Edit one copy and the others become stale. Delete one and you may forget the others exist. Symbolic links and shortcuts offer partial relief, but they break when the original is moved, and many applications do not handle them gracefully.

Hardlinks solve these problems at the filesystem level. A hardlink is not a pointer to a file—it is the file, sharing the same underlying data on disk. Multiple hardlinks to the same file occupy disk space only once, remain valid even if other links are renamed or moved within the same volume, and are transparent to every application. The file simply appears to exist, fully and independently, in every location where a hardlink has been placed.

Hardlink Manager was created to make these powerful filesystem capabilities accessible through a graphical interface. Rather than requiring users to invoke command-line utilities (such as `mklink` on Windows or `ln` on Unix), the application provides an intuitive way to create, manage, and discover hardlinks across directories. Its Mirror Group system extends this further, enabling automatic synchronization of entire folder sets—so that a file added to any one mirror automatically appears in all the others.

The project originated from the needs of a scholarly archive organized across three complementary indexing schemes: a primary catalogue (organized by format, language, and period), an onomasticon (a name index with entries in multiple scripts and transliterations), and a categoricum (a thematic index for cross-referential research). Maintaining consistency across these interrelated structures by hand proved unsustainable. Hardlink Manager was designed to automate and simplify this work.

1.2 Potential Uses

Hardlink Manager is suited to any scenario in which the same files must be organized under multiple classification schemes simultaneously without duplication:

- Scholarly and Research Archives — Maintain documents under multiple organizational hierarchies (by author, by subject, by date, by language) with a single copy of each file.
- Multilingual Collections — Provide access to the same materials under names in different scripts or transliterations (Greek, Latin, Cyrillic, Arabic, etc.).
- Media Libraries — Organize music, photographs, or video files by multiple criteria (artist, album, genre, year) without multiplying storage usage.

- Institutional Records Management — File documents under departmental, chronological, and project-based hierarchies simultaneously.
- Software Development — Share configuration files, assets, or test fixtures across multiple project directories on the same volume.
- Personal Knowledge Management — Organize notes, papers, and references under overlapping topic trees without worrying about which copy is the canonical one.

The intersection search feature is particularly valuable for cross-referential research, enabling the discovery of documents that span multiple thematic categories.

1.3 About This Manual

This manual covers installation, core concepts, and practical usage of Hardlink Manager version 0.2.0. It is organized to be read sequentially by new users or consulted by section as a reference. Chapter 2 introduces the concept of hardlinks for readers unfamiliar with them. Chapters 3–4 cover installation and first launch. Chapters 5–8 describe the application's features in detail. Chapters 9–10 cover shortcuts and configuration. Chapter 11 provides worked examples, and Chapters 12–13 address troubleshooting and technical details.

2. Understanding Hardlinks

2.1 What Are Hardlinks?

Every file on a modern filesystem is identified internally by an inode (on Unix/macOS) or a file index number (on Windows NTFS). The inode stores the file's actual data location, size, permissions, and other metadata. What we usually think of as a "filename" is really a directory entry that maps a human-readable name to an inode.

A hardlink is simply an additional directory entry pointing to the same inode. Because multiple directory entries can reference the same inode, the same file can appear in multiple directories under different names—or even under the same name in different locations. The operating system tracks how many directory entries (links) reference each inode; this is the "link count."

When you delete a file, you are actually removing one directory entry. The underlying data is only freed when the link count drops to zero—that is, when no remaining directory entries refer to that inode. This means you can safely delete a hardlink from one location without affecting the file's availability at other locations.

2.2 Why Hardlinks?

- Storage efficiency: Multiple appearances of a file consume disk space only once.
- Transparency: Applications see a normal file at each path; no special handling needed.
- Resilience: Renaming or deleting one link does not invalidate others.
- Automatic content sync: Editing the file through any link edits the same data.
- No dangling references: Unlike symbolic links, hardlinks cannot become "broken."

2.3 Constraints and Limitations

- Same volume only: Hardlinks can only be created between files on the same disk partition or volume. Cross-drive hardlinks are not possible.
- Files only: Directories cannot be hardlinked. Mirror Groups work around this by synchronizing individual files within directories.
- NTFS required (Windows): Hardlinks require an NTFS-formatted volume. FAT32 and exFAT do not support them.
- Permissions: On some systems, creating hardlinks may require elevated privileges.

NOTE

Hardlink Manager validates volume compatibility before creating links and will alert you if a cross-volume operation is attempted.

3. Installation

3.1 System Requirements

- Python 3.7 or later (for running from source)
- Operating system: Windows, macOS, or Linux
- Filesystem: NTFS (Windows), HFS+/APFS (macOS), or ext4/XFS/Btrfs (Linux)
- Dependencies: watchdog (installed automatically via pip)

3.2 Installing from Source

Clone the repository and install dependencies:

```
git clone https://github.com/asturrulumbo/  
    Hardlink-Based-File-Explorer.git  
cd Hardlink-Based-File-Explorer  
pip install -r requirements.txt
```

The application can then be launched directly (see Chapter 4).

3.3 Building a Standalone Executable

To distribute the application without requiring a Python installation, you can build a standalone executable using PyInstaller:

```
python build.py          # Single-file executable  
python build.py --onedir  # Directory-based bundle  
python build.py --clean   # Clean previous builds first
```

The output will be placed in the `dist/` directory. The single-file option produces one portable executable; the directory option produces a folder with faster startup time.

4. Getting Started

4.1 Launching the Application

From the project directory, run:

```
python -m hardlink_manager
```

Alternatively, if you built a standalone executable:

```
./dist/HardlinkManager      # Linux / macOS  
dist\HardlinkManager.exe   # Windows
```

4.2 The Main Window

The main window is divided into two primary areas:

- Left panel: A hierarchical directory tree for navigating your filesystem.
- Right panel: A tabbed notebook with three tabs — File Browser, Mirror Groups, and Intersection Search.

The File Browser tab displays the contents of selected directories in a tabbed interface, allowing multiple folders to be open simultaneously. The Mirror Groups tab provides tools for creating and managing synchronized folder sets. The Intersection Search tab enables multi-folder searches for shared files.

4.3 Navigating the Interface

- Click a folder in the left tree to expand it and view its subdirectories.
- Double-click a folder to open it in a new tab in the File Browser.
- Right-click files or folders to access context menus with hardlink operations.
- Use File > Add Folder to Tree to add root directories to the navigation tree.
- The status bar at the bottom displays metadata for the selected item.

5. File Browser

5.1 Directory Tree

The left-hand directory tree provides a hierarchical view of your filesystem. Directories are loaded lazily—their contents are fetched only when you expand them—which keeps the interface responsive even for deeply nested structures.

To add a new root folder to the tree, use **File > Add Folder to Tree** from the menu bar. Multiple root folders can be active simultaneously.

5.2 File List and Tabs

When you open a directory, its contents are displayed in the right-hand panel as a table with the following columns:

- **Name** — The file or subdirectory name.
- **Size** — The file size in a human-readable format.
- **Hardlink Count** — The number of hardlinks to the same underlying data.
- **Inode** — The filesystem inode (or file index number on Windows).

Multiple directories can be open in separate tabs. Click a tab to switch between open directories. Tabs can be closed individually.

5.3 File Operations

Standard file operations are accessible through the context menu (right-click) or keyboard shortcuts:

- **Open** — Open the selected file with the system's default application.
- **Open in Explorer** — Reveal the file in the operating system's file manager.
- **Copy / Cut / Paste** — Standard clipboard operations for files and folders.
- **Rename** — Rename the selected file or folder (F2).
- **Delete** — Delete the selected item (Delete key).

6. Hardlink Operations

6.1 Creating a Hardlink

To create a hardlink to an existing file:

1. Right-click the source file in the File Browser.
2. Select "Create Hardlink To..." from the context menu.
3. In the dialog, browse to and select the destination folder.
4. Optionally, specify a custom name for the new link. If left blank, the original filename is used.
5. Click OK to create the hardlink.

The new hardlink will appear in the destination folder. It references the same underlying data as the source file; edits to either are reflected in both.

NOTE

Both the source file and destination folder must reside on the same filesystem volume. If they do not, the application will display an error.

6.2 Viewing Hardlinks

To view all hardlinks to a given file:

1. Right-click the file in the File Browser.
2. Select "View Hardlinks" from the context menu.

A dialog will display every path on the system that shares the same inode as the selected file. Each path listed is a hardlink to the same underlying data. The dialog also shows the inode number and total link count. Clicking a path in the list navigates to that location in the File Browser.

6.3 Deleting a Hardlink

To delete a hardlink:

1. Right-click the file and select "Delete" or press the Delete key.
2. A confirmation dialog will appear.

If the file has a link count greater than one (i.e., other hardlinks exist), the dialog will inform you that only this directory entry will be removed—the underlying data remains accessible through the other links. If this is the last remaining link, the dialog warns that the data will be permanently deleted.

7. Mirror Groups

7.1 Concept Overview

A Mirror Group is a set of two or more directories that are treated as equivalent mirrors of each other. All directories in a mirror group maintain identical file contents through hardlinks. When a file is added to any directory in the group, hardlinks to that file are automatically created in every other directory in the group (if automatic synchronization is enabled).

Mirror groups are useful when a single logical entity—a person, a periodical, a theme—has multiple representations in your file hierarchy (e.g., the same author filed under names in different scripts).

Each directory in a mirror group is marked with a hidden `.hardlink_mirror` file, which allows the application to recognize mirror membership across sessions.

7.2 Creating a Mirror Group

There are two ways to create a mirror group:

Method 1: From the Mirror Groups Panel

1. Switch to the Mirror Groups tab in the right panel.
2. Click the "New Group" button.
3. In the dialog, add two or more directories to the group.
4. Assign a name to the group (or accept the auto-generated name).
5. Click OK. The group will be created and an initial synchronization will run to ensure all directories share the same files.

Method 2: From the Context Menu

1. Right-click a folder in the File Browser or directory tree.
2. Select "Create Hardlink Mirror..." from the context menu.
3. The selected folder will be pre-populated; add additional folders.
4. Confirm to create the group.

7.3 Managing Mirror Groups

The Mirror Groups panel lists all registered groups with the following information:

- Group name
- Number of directories in the group

- Synchronization status (enabled or disabled)

From this panel, you can:

- Edit a group — Add or remove directories, rename the group.
- Delete a group — Remove the group registration. The directories and their files are not affected; only the mirror relationship is dissolved.
- Sync Now — Manually trigger a full synchronization of all directories in the group, ensuring they all contain the same files.
- Toggle Sync — Enable or disable automatic filesystem watching for the group.

7.4 Automatic Synchronization

When automatic synchronization is enabled for a mirror group, the application uses a filesystem watcher (powered by the watchdog library) to monitor all directories in the group for new file additions.

When a new file is detected in any watched directory, the system waits briefly (0.5 seconds by default) to allow the file write to complete, then creates hardlinks to that file in every other directory in the group. The relative path within the directory is preserved, so subdirectory structures are maintained.

NOTE

The filesystem watcher runs in a background thread and is designed to be resource-efficient through debouncing. It can be toggled on or off per group.

7.5 Scanning for Existing Mirrors

If you have directories that are already mirrors of each other (created manually or by another tool), the application can discover them automatically:

Content-Based Scanning

Computes SHA-256 fingerprints of directory contents and groups directories with identical content signatures. This detects mirrors regardless of how they were created.

Hardlink-Based Scanning

Scans for directories that share files with the same inode numbers. Directories sharing any hardlinked files are proposed as a mirror group. Uses a union-find algorithm to transitively connect related directories.

Both scanning methods are accessible from the Mirror Groups panel via the "Scan for Mirrors" button. Scan results are presented for review before any groups are created.

8. Intersection Search

8.1 Running a Search

The Intersection Search feature finds files that appear in all of a specified set of directories. This is particularly useful for discovering documents that span multiple organizational categories.

1. Switch to the Intersection Search tab.
2. Add two or more directories to the search set using the folder selector.
3. Optionally, enter a filename pattern to filter results (case-insensitive substring matching).
4. Click Search.

8.2 Interpreting Results

Results are displayed in a table with the following columns:

- Filename — The name of the file found in all specified directories.
- Size — The file size.
- Inode — The inode number confirming the files are hardlinked (same data).
- Locations — The full paths where the file was found.

The search works by comparing inode numbers across directories, so it correctly identifies hardlinked files even if they have been renamed in different locations.

9. Keyboard Shortcuts & Context Menus

The following keyboard shortcuts are available throughout the application:

Shortcut	Action
Ctrl+C	Copy selected item
Ctrl+X	Cut selected item
Ctrl+V	Paste
Delete	Delete selected item
F2	Rename selected item
Right-click	Open context menu

Context menus provide the following additional actions depending on context:

File Context Menu

- Open / Open in Explorer
- Copy / Cut / Paste
- Create Hardlink To...
- View Hardlinks
- Rename / Delete

Folder Context Menu

- Open Folder / Open in New Tab / Open in Explorer
- Copy / Cut / Paste
- Create Hardlink Mirror...
- Add to Existing Mirror...
- View Hardlink Mirrors
- Rename / Delete

10. Configuration and Data Storage

Hardlink Manager stores its persistent data (mirror group registrations) in a platform-specific application data directory:

Platform	Path
Windows	%APPDATA%/HardlinkManager/
macOS	~/Library/Application Support/HardlinkManager/
Linux	~/.config/hardlink_manager/

The primary data file is `mirror_groups.json`, which stores all mirror group definitions including group IDs, names, folder paths, synchronization settings, and timestamps. This file is read on startup and updated automatically whenever groups are modified.

Each directory belonging to a mirror group also contains a hidden `.hardlink_mirror` marker file. These markers enable the application to discover mirror memberships during scanning operations.

NOTE

Deleting the `mirror_groups.json` file will reset all group registrations. The marker files in individual directories will allow groups to be rediscovered via the scanning feature.

11. Example Workflows

11.1 Scholarly Archive with Multilingual Names

Scenario: You maintain an archive of patristic scholarship. Authors are known by names in multiple scripts, and you want each variant accessible as a folder name.

1. Create folders in your Onomasticum directory for each name variant: John_of_Damascus/, Yuhanna_ad-Dimashqi/, etc.
2. Right-click any one of these folders and select "Create Hardlink Mirror..."
3. Add all variant folders to the new mirror group.
4. Add a PDF to any one folder. With synchronization enabled, hardlinks automatically appear in all other variant folders.
5. Use Intersection Search to find documents present across both the Onomasticum name folders and thematic Categoricalum folders.

11.2 Thematic Cross-Referencing

Scenario: You have a thematic index (Categoricalum) with categories like Theology/Christology and Literature/Theological_Poetry. You want to find documents relevant to both themes.

1. Manually hardlink relevant documents into each thematic folder using "Create Hardlink To..." from the context menu.
2. Open the Intersection Search tab.
3. Add both Theology/Christology and Literature/Theological_Poetry to the search set.
4. Click Search. The results show all documents that appear in both categories—your cross-referential overlap.

11.3 Periodical with Alternate Titles

Scenario: A Greek periodical from the 1920s is catalogued under both its Greek title and a Latin transliteration.

1. Create both title folders under the appropriate catalogue path.
2. Create a mirror group containing both folders.
3. Add issues to either folder; they are automatically mirrored to the other.
4. Researchers searching by either title will find the same complete collection.

12. Troubleshooting

"Cannot create hardlink: files are on different volumes"

Hardlinks require both the source and destination to be on the same filesystem volume. Ensure both paths are on the same drive or partition. Use the operating system's disk management tools to verify volume assignments.

"Permission denied" when creating hardlinks

On some systems, creating hardlinks requires administrative or elevated privileges. On Windows, try running the application as Administrator. On Unix systems, ensure you have write permissions to the destination directory.

Mirror group synchronization not working

Verify that automatic synchronization is enabled for the group (check the toggle in the Mirror Groups panel). Ensure the application is running—the filesystem watcher only operates while Hardlink Manager is active. If issues persist, try a manual "Sync Now" to diagnose any underlying errors.

Files appear with hardlink count of 1

A hardlink count of 1 means only one directory entry references this file—it has no additional hardlinks. This is the default state for any newly created file.

Scanning does not find expected mirrors

Content-based scanning requires directories to have identical contents. If files have diverged, the scan will not match them. Try hardlink-based scanning instead, which detects any shared inodes. Also ensure the directories being scanned are on the same volume.

13. Appendix: Technical Reference

13.1 Architecture Overview

Hardlink Manager is organized into three layers:

- Core (hardlink_manager/core/) — Business logic for hardlink operations, mirror groups, synchronization, filesystem watching, and search.
- UI (hardlink_manager/ui/) — The tkinter-based graphical interface, including the main window, file browser, mirror panel, search panel, and modal dialogs.
- Utilities (hardlink_manager/utils/) — Cross-platform filesystem helpers for inode queries, volume validation, and filename sanitization.

13.2 Core Modules

hardlink_ops.py

Provides `create_hardlink()`, `delete_hardlink()`, and `find_hardlinks()` functions. Validates same-volume constraints before operations.

mirror_groups.py

Implements `MirrorGroupRegistry` for persistent group management. Includes both content-based (SHA-256 fingerprinting) and hardlink-based (union-find) mirror discovery algorithms.

sync.py

Handles file synchronization across mirror group directories. Preserves relative path structure and creates intermediate directories as needed.

watcher.py

Filesystem event monitoring using the `watchdog` library. Implements debounced, thread-safe watching with per-group toggle support.

search.py

Inode-based intersection search across multiple directories with optional filename pattern filtering.

13.3 Data Formats

mirror_groups.json

```
{
  "groups": [
    {
      "id": "uuid-string",
      "name": "Group Name",
      "folders": ["/path/a", "/path/b"],
      "sync_enabled": true,
      "created_at": "ISO-8601 timestamp",
      "modified_at": "ISO-8601 timestamp"
    }
  ]
}
```

13.4 Cross-Platform Considerations

- Windows: Uses `os.stat().st_ino` for file index numbers. Requires NTFS. Filename sanitization strips Windows-forbidden characters.
- macOS: Standard POSIX inode handling via `os.stat()`. HFS+ and APFS supported.
- Linux: Standard POSIX inode + device ID. Works with ext4, XFS, Btrfs, and other common filesystems.

Hardlink Manager version 0.2.0

This manual was generated for the Hardlink-Based File Explorer project.