

CollectFiles

Inhaltsverzeichnis

CollectFiles 1

Voraussetzungen und Start 1

Ablage nach Jahreszahl 2

Vorhandene Dateien und Protokolle 3

Fortschrittsanzeige 4

Abbrechen und erneut starten 5

Tests 5

Der Prompt 5

Sucht unter Ubuntu 24.04 rekursiv in einem oder mehreren Verzeichnissen nach Fotos, RAW-Bildern und Videos und verschiebt sie anhand ihres Aufnahmedatums in einen Sammelordner.

Unterstützte Dateiendungen (Groß-/Kleinschreibung spielt keine Rolle):

- **Fotos/Bilder:** .jpg, .jpeg, .jpe, .jfif, .png, .gif, .bmp, .tif, .tiff, .heic, .heif, .hif, .webp, .avif, .jxl.
- **RAW:** .dng, .cr2, .cr3, .crw, .nef, .nrw, .arw, .srf, .sr2, .orf, .rw2, .raw, .raf, .pef, .ptx, .srw, .rwl, .3fr, .fff, .iiq, .mos, .mrw, .kdc, .dcr, .errf, .x3f.
- **Videos:** .mp4, .mov, .mkv, .avi, .webm, .m4v, .3gp, .3g2, .mts, .m2ts, .m2t, .ts, .mpg, .mpeg, .mpe, .m2v, .wmv, .asf, .flv, .f4v, .vob, .ogv, .mxf.

Die Dateien behalten ihr Originalformat. Ohne ermittelbares Jahr aus Ordnername, Dateiname oder Metadaten werden sie übersprungen. Begleitdateien wie .xmp und .json werden nicht gesammelt.

Voraussetzungen und Start

Installation für den eigenen Benutzer (Skript ohne sudo starten):

```
cd ~/Dokumente/ChatGPT/collectFiles
./install.sh
export PATH="$HOME/.local/bin:$PATH"
collectFiles ~/Dropbox ~/tmp ~/verz1 --output ~/Fotosammlung --dry-run
```

Das Skript installiert einen eigenständigen Befehl nach

~/ .local/bin/collectFiles. Python 3.11 oder neuer und ExifTool werden geprüft.

Fehlende Pakete werden unter Ubuntu über apt-get installiert; dafür fragt gegebenenfalls sudo nach dem Passwort. Zusätzliche Python-Pakete werden nicht benötigt. Ein erneuter Aufruf aktualisiert die Installation. Danach wird das Projektverzeichnis zum Ausführen des installierten Befehls nicht mehr benötigt.

Die export-Zeile kann dauerhaft in ~/ .bashrc eingetragen werden, falls ~/ .local/bin noch nicht im Suchpfad enthalten ist. Ein anderes Installationsverzeichnis ist über --prefix

möglich:

```
./install.sh --prefix "$HOME/Programme/collectFiles"  
# Installiert nach ~/Programme/collectFiles/bin/collectFiles
```

Alternativ ist der direkte Start im Projekt mit `./collectFiles` möglich; dabei müssen `collectFiles` und `collect_files.py` zusammenbleiben. Ohne `--dry-run` werden Dateien tatsächlich verschoben. Der Probelauf erzeugt nur Textprotokolle. In der Shell `~/Dropbox` ohne Backslash vor `~` verwenden. Pfade mit Leerzeichen etwa als `"$HOME/Meine Bilder"` angeben.

Ablage nach Jahreszahl

Standardziel ist `./SammeIordner`, bezogen auf das aktuelle Arbeitsverzeichnis. Dateien landen unter `--output/JJJJ/Unterordner/Originaldateiname`, beispielsweise `~/Fotosammlung/2026/2026-09-16 09.55.05.jpg`. Die Unterordnerstruktur bleibt erhalten: Bei einem erkannten Jahresordner wird die Struktur darunter übernommen, sonst die Struktur relativ zum jeweiligen Suchpfad. Beispielsweise wird `Photos from 2002/Urlaub/Strand/video.mp4` zu `--output/2002/Urlaub/Strand/video.mp4`. Ohne Jahresordner wird Suchpfad `Urlaub/20170731_114105.jpg` zu `--output/2017/Urlaub/20170731_114105.jpg`. Auch wenn der Suchpfad innerhalb eines Jahresordners liegt, bleibt die Struktur unterhalb dieses Jahresordners erhalten. Gleichnamige Dateien in verschiedenen Unterordnern erhalten dadurch unterschiedliche Zielpfade. Alle unterstützten Mediendateien mit ermittelbarem Jahr werden berücksichtigt.

```
collectFiles ~/Dropbox ~/tmp ~/verz1 \  
--output ~/Fotosammlung \  
--dry-run
```

Die Reihenfolge zur Ermittlung des Zieljahres ist:

1. **Ordnerjahr:** Ein übergeordneter Ordner wie `1993`, `2002` oder `Photos from 2002` bestimmt das Zieljahr für alle unterstützten Mediendateien darin, auch in Unterordnern. Dateinamen und EXIF-/MP4-Metadaten werden dabei nicht geprüft. Die Dateien landen in `--output/2002/` mit ihren bisherigen Unterordnern.
2. **Dateiname:** Ohne Ordnerjahr wird ein gültiger Zeitstempel verwendet, beispielsweise `2026-09-16 09.55.05.jpg`, `20170731_114105.jpg` oder `IMG_20190225_094418.JPG`. Präfixe wie `VID_` werden ebenfalls unterstützt. Auch ein Datum ohne Uhrzeit im Format `JJJJMMTT` wird erkannt: `VID-20201123-WA0002.mp4` wird beispielsweise nach `2020` einsortiert. Im Protokoll erscheint die fehlende Uhrzeit als `00:00:00`. Unix-Zeitstempel werden als 10-stellige Sekundenwerte (z. B. `1609459200.jpg`) oder 13-stellige Millisekundenwerte (z. B. `IMG_1609459200123.jpg`) erkannt. Sie werden in UTC umgerechnet; beide Beispiele landen im Jahresordner `2021`. Die Zahl darf nicht Teil einer längeren Ziffernfolge sein. Enthält der Name auch ein gültiges Kalenderdatum, hat dieses Vorrang.
3. **EXIF-/MP4-Metadaten:** Wenn der Dateiname kein gültiges Datum enthält, wird das erste

gültige Feld aus `DateTimeOriginal`, `CreateDate`, `CreationDate`, `MediaCreateDate`, `TrackCreateDate` verwendet. Bei MP4 sind dies die entsprechenden QuickTime-Metadaten.

Die Ordnerregel gilt auch für Unterordner und andere Jahre. Erkannt werden außerdem `photos_from_2022`, `Photos-from-2022`, `Photos 2022`, `Fotos von 2022`, `Fotos aus 2022` und Kopien wie `Photos from 2022 (1)`. Reine Jahresordner bestehen aus genau vier Ziffern mit einer Jahreszahl von 1000 bis 9999. Beispielsweise landet `1993/Urlaub/video.mp4` unter `1993/Urlaub/video.mp4`, unabhängig von Dateiname und Metadaten. Groß-/Kleinschreibung spielt keine Rolle. Bei verschachtelten Jahresordnern gilt der nächste passende übergeordnete Ordner. Wird das Ordnerjahr verwendet, steht im Protokoll nur das Jahr in der Datumsspalte und `Ordnername` als Datumsquelle.

Beispiel: `Photos from 2022/IMG_20190225_094418.JPG` wird nach `--output/2022` verschoben, unabhängig vom Datum im Dateinamen oder in EXIF. Erst wenn alle drei Quellen kein gültiges Datum bzw. Jahr liefern, wird die Datei übersprungen. Das Änderungsdatum des Dateisystems wird nicht verwendet. Maßgeblich ist das Kalenderdatum, wie es ExifTool ausgibt, ohne zusätzliche Zeitzonenumrechnung. MP4-Zeitstempel können je nach Kamera UTC oder lokale Zeit enthalten. Das Programm kann falsche Kameraangaben nicht erkennen.

Vorhandene Dateien und Protokolle

Existiert der berechnete Zielpfad bereits, werden Größe und SHA-256-Inhalt von Quelle und Ziel verglichen. Bei identischem Inhalt wird die **Zielfdatei** ohne Logeintrag unter `./collectFiles-logs/exists/` als Hardlink abgelegt. Erst nach Anlegen des Hardlinks und erneuter Prüfung auf Veränderungen wird die Quelldatei entfernt. Es werden keine Dateiinhalte kopiert. Zielfdatei und Hardlink verweisen auf dieselben Daten; Änderungen über einen der Pfade sind auch über den anderen sichtbar. Bei abweichendem Inhalt bleibt die Quelle erhalten und ein Fehler wird protokolliert. Hardlinks erfordern dasselbe Dateisystem für **Ziel und exists** und dessen Unterstützung. Die Quelle darf auf einem anderen Dateisystem liegen. Schlägt das Verlinken oder Löschen fehl, bleibt die Quelle erhalten und der Fehler wird protokolliert; es gibt keinen Rückfall auf eine Kopie. Mit `--log-dir PFAD` liegt dieser Ordner unter `PFAD/exists/`. Das vorhandene Ziel bleibt erhalten. Bei Namenskollisionen im Ordner `exists` werden freie Namen mit `_1`, `_2` usw. vor der Dateiendung verwendet. Das gilt auch für mehrere neue Dateien mit gleichem Zielpfad: Die erste landet im Jahresordner, weitere werden in `exists` verlinkt. Fehler werden weiterhin in `errors.txt` protokolliert. Im Probelauf bleiben alle Quelldateien erhalten und `exists` wird nicht angelegt. Die Konsolenzusammenfassung zählt diese Dateien separat als „nach exists“.

Jeder Aufruf erzeugt einen eigenen Unterordner in `./collectFiles-logs/`. Mit `--log-dir PFAD` kann ein anderer Protokollordner gewählt werden. Die Textdateien sind UTF-8-kodierte Tabellen mit Tabulator als Trennzeichen:

Dateipfade in allen Protokollen sind relativ zum **Arbeitsverzeichnis beim Aufruf**, auch in Begründungen und Fehlermeldungen. Beispiel beim Aufruf aus `~/Bilder: Eingang/foto.jpg` → `Sammlung/2026/foto.jpg`. Dateien außerhalb dieses

Verzeichnisses werden mit `./` dargestellt. Ein anderer `--log-dir` ändert diesen Bezugspunkt nicht.

Datei	Inhalt
<code>moved.txt</code>	Relativer Quell- und Zielpfad, Datum und Datumsquelle
<code>skipped.txt</code>	Übersprungene Quelldateien mit Begründung, außer bereits vorhandenen Zieldateien
<code>errors.txt</code>	Fehler beim Lesen, Kopieren oder Löschen

Auf demselben Dateisystem wird zunächst ein zusätzlicher Dateiverweis (Hardlink) am Ziel angelegt, ohne die Dateiinhalte zu kopieren. Nach dem Protokolleintrag **LINKED** wird der Quellpfad entfernt und **MOVED** protokolliert. Das Ziel wird dabei ausschließlich neu angelegt; vorhandene Dateien werden niemals überschrieben.

Über Dateisystemgrenzen oder wenn Hardlinks nicht unterstützt werden, wird die Kopie vollständig geschrieben und ihr Inhalt geprüft. Danach folgt **COPIED**, anschließend die Entfernung der Quelle und **MOVED**. Für die Ablage unter **exists** wird ausschließlich ein Hardlink auf die identische Zieldatei angelegt; danach wird die Quelle ohne diese Logeinträge entfernt. **WOULD_MOVE** kennzeichnet einen Probelauf.

Für Metadaten bleibt ExifTool während der Abfrage geöffnet, statt für jede Zehnergruppe neu zu starten. Die Fortschrittsanzeige bleibt in 10er-Schritten. Dateinamen mit Zeilenumbrüchen werden über separate Aufrufe verarbeitet.

Suchpfade dürfen sich nicht überschneiden. Symbolischen Links wird nicht gefolgt. Sammel- und Protokollordner innerhalb eines Suchpfads werden von der Suche ausgeschlossen. Während eines Laufs sollten andere Programme die beteiligten Dateien und Verzeichnisse nicht verändern; insbesondere parallele Läufe vermeiden. Verschieben über Dateisystemgrenzen ist möglich, benötigt aber Platz für die Kopie.

Rückgabewerte: **0** erfolgreich (auch bei regulär übersprungenen Dateien), **1** Verarbeitungsfehler, **2** ungültiger Aufruf, **130** Abbruch mit Strg+C.

Fortschrittsanzeige

Nach der Suche zeigt das Programm die Gesamtzahl der gefundenen unterstützten Mediendateien an. Nach jeweils 10 bearbeiteten Dateien werden die Zähler aktualisiert. Der Endstand wird auch bei weniger als 10 verbleibenden Dateien angezeigt:

```
3139/140  Error/Skip/Exists 0/0/0
```

Die ersten beiden Zahlen bedeuten Gesamtzahl/bereits bearbeitet. Danach folgen Fehler/übersprungen/unter exists verlinkt. „Bearbeitet“ umfasst erfolgreich verschobene Dateien, übersprungene Dateien und Dateien mit Verarbeitungsfehlern. Dateien ohne ermittelbares Jahr und symbolische Links zählen als übersprungen. Dateien mit vorhandenem Ziel werden separat unter „Exists“ gezählt. Im Probelauf beziehen sich die Zahlen auf die geplanten Aktionen. Während der Metadatenabfrage wird der Fortschritt jeweils nach bis zu 10 Dateien angezeigt. Die Anzeige

überschreibt mit einem Wagenrücklauf immer dieselbe Zeile, auch wenn die Ausgabe nicht als Terminal erkannt wird. Erst am Ende folgt ein Zeilenumbruch. Beim Kopieren einer einzelnen großen Datei bleiben die Zähler bis zum Abschluss dieser Datei unverändert.

Abbrechen und erneut starten

Das Programm kann jederzeit mit **Strg+C** abgebrochen werden. Bereits vollständig gelesene Metadaten werden nach jeder Zehnergruppe in `collectFiles-logs/metadata-cache.sqlite3` gespeichert. Beim nächsten Lauf mit demselben Log-Ordner werden sie für unveränderte Dateien wiederverwendet. Mit `--log-dir PFAD` liegt der Zwischenspeicher unter `PFAD/metadata-cache.sqlite3`. Das gilt auch für Probelaufe. Fehlgeschlagene Abfragen sowie Dateien, deren Identität, Größe oder Änderungszeiten sich geändert haben, werden erneut gelesen. Eine beim Abbruch noch nicht gespeicherte Gruppe wird ebenfalls erneut abgefragt. Die Metadaten in den Mediendateien selbst werden nicht verändert.

Bereits verschobene Dateien bleiben im Ziel. Der Quellpfad wird erst entfernt, nachdem der Zielverweis angelegt bzw. die Kopie vollständig geschrieben und ihr Inhalt geprüft wurde.

Bei einem Abbruch können beide Pfade gleichzeitig vorhanden sein. Beim schnellen Verschieben über Hardlinks verweisen sie auf dieselben Dateiinhalte. Bei einem harten Abbruch, etwa durch Stromausfall oder erzwungenes Beenden, kann auch eine unvollständige Zieldatei zurückbleiben.

Beim nächsten Lauf wird ein bereits vorhandenes Ziel nicht auf Vollständigkeit geprüft oder vervollständigt. Bei abweichendem Inhalt (etwa einem unvollständigen Ziel) bleibt die Quelle erhalten und ein Fehler wird protokolliert. Nur bei identischem Inhalt wird die Zieldatei unter `collectFiles-logs/exists/` verlinkt und die Quelle entfernt. Bei Verwendung von `--log-dir PFAD` liegt sie entsprechend unter `PFAD/exists/`.

Tests

```
python3 -m unittest discover -s tests -v
```

Die Tests verwenden temporäre Verzeichnisse und prüfen unter anderem Jahresordner, Namenskollisionen, Probelauf und Kopierfehler.

Der Prompt

Erstelle ein vollständiges Kommandozeilenprogramm namens „collectFiles“ für Ubuntu 24.04 mit Python 3.11 oder neuer. Es soll Fotos, RAW-Bilder und Videos rekursiv aus mehreren Verzeichnissen sammeln und nach ihrem Aufnahmejahr in einen Zielordner verschieben.

Verwende ausschließlich die Python-Standardbibliothek und das externe Programm ExifTool. Oberfläche, Hilfetexte, Protokolle und Fehlermeldungen sollen deutsch sein.

Aufruf:

```
collectFiles SUCHPFAD [SUCHPFAD ...] --output ZIEL --log-dir PROTOKOLLORDNER --dry-run
```

Standardwerte:

- Ziel: ./Sammelordner
- Protokollordner: ./collectFiles-logs
- Ohne --dry-run werden Dateien tatsächlich verschoben.

Anforderungen:

1. Dateisuche

- Suche rekursiv in allen angegebenen Verzeichnissen.
- Unterstütze gängige Bildformate einschließlich JPEG, PNG, TIFF, HEIC, HEIF, WebP, AVIF und JXL; Kamera-RAW-Formate einschließlich DNG, CR2, CR3, NEF, ARW, ORF, RW2 und RAF; Videoformate einschließlich MP4, MOV, MKV, AVI, WebM, MTS, M2TS und MPEG.
- Beachte Dateierweiterungen unabhängig von Groß-/Kleinschreibung.
- Behalte Originalformat und Dateinamen bei.
- Ignoriere Begleitdateien wie XMP und JSON.
- Folge keinen symbolischen Links.
- Schließe Ziel- und Protokollordner von der Suche aus.
- Lehne überlappende Suchpfade mit verständlicher Fehlermeldung ab.

2. Ermittlung des Zieljahres

Verwende diese feste Priorität:

- a) Nächster übergeordneter Jahresordner.
- b) Gültiges Datum im Dateinamen.
- c) Aufnahmezeit aus ExifTool-Metadaten.

Erkenne Jahresordner wie:

1993

Photos from 2002

photos_from_2022

Photos-from-2022

Photos 2022

Fotos von 2022

Fotos aus 2022

Photos from 2022 (1)

Reine Jahresordner müssen genau vier Ziffern enthalten; gültige Jahre sind 1000 bis 9999. Berücksichtige auch Jahresordner oberhalb des Suchpfads. Bei verschachtelten Jahresordnern gewinnt der nächstgelegene. Ein Ordnerjahr hat Vorrang vor Dateiname und Metadaten.

Erkenne beispielsweise diese Dateinamen:

2026-09-16 09.55.05.jpg

20170731_114105.jpg

IMG_20190225_094418.JPG

VID-20201123-WA0002.mp4

1609459200.jpg

IMG_1609459200123.jpg

Prüfe Kalenderdaten auf Gültigkeit. Ein Datum ohne Uhrzeit erhält 00:00:00. Erkenne Unix-Zeitstempel mit 10 Stellen als Sekunden und 13 Stellen als Millisekunden und rechne sie in UTC um. Sie dürfen nicht Teil längerer Ziffernfolgen sein. Gültige Kalenderdaten haben Vorrang vor Unix-Zeitstempeln.

Prüfe Metadaten in dieser Reihenfolge:

DateTimeOriginal

CreateDate

CreationDate

MediaCreateDate

TrackCreateDate

Übernimm das von ExifTool ausgegebene Kalenderdatum ohne zusätzliche Zeitzonenumrechnung. Verwende niemals das Änderungsdatum des Dateisystems. Überspringe Dateien ohne ermittelbares Jahr mit protokollierter Begründung.

3. Zielstruktur

Speichere unter:

ZIEL/JJJJ/Unterordner/Originaldateiname

Ist ein Jahresordner vorhanden, übernimm die Verzeichnisstruktur darunter. Andernfalls übernimm die Struktur relativ zum jeweiligen Suchpfad.

Beispiele:

Photos from 2002/Urlaub/Strand/video.mp4

→ ZIEL/2002/Urlaub/Strand/video.mp4

SUCHPFAD/Urlaub/20170731_114105.jpg

→ ZIEL/2017/Urlaub/20170731_114105.jpg

Liegt der Suchpfad innerhalb eines Jahresordners, behalte ebenfalls die gesamte Struktur unterhalb dieses Jahresordners bei.

4. Sicheres Verschieben

- Überschreibe niemals vorhandene Ziele, auch keine symbolischen Links.
- Auf demselben Dateisystem: Erzeuge einen Hardlink am Ziel, protokolliere LINKED, entferne erst danach den Quellpfad und protokolliere MOVED.
- Über Dateisystemgrenzen oder bei fehlender Hardlink-Unterstützung: Kopiere exklusiv, prüfe den vollständigen Inhalt mit SHA-256, protokolliere COPIED und entferne erst danach die Quelle. Protokolliere anschließend MOVED.
- Bei Kopier-, Prüf- oder Protokollfehlern muss die Quelle erhalten bleiben.
- Entferne bei abfangbaren Kopierfehlern unvollständige neu angelegte Ziele.
- Prüfe relevante Dateiveränderungen vor dem Entfernen der Quelle.

5. Bereits vorhandene Ziele

Vergleiche Quelle und Ziel anhand von Größe und SHA-256.

Bei identischem Inhalt:

- Erzeuge unter PROTOKOLLORDNER/exists einen Hardlink auf die bestehende Zieldatei.
- Verwende bei Namenskollisionen freie Namen mit _1, _2 usw. vor der Endung.
- Entferne die Quelle erst nach erfolgreicher Verlinkung und erneuter Prüfung auf Veränderungen.
- Schreibe hierfür keinen Eintrag in moved.txt oder skipped.txt.
- Zähle den Vorgang separat als Exists.
- Verwende ausschließlich Hardlinks; kein Rückfall auf Kopieren.
- Bei Fehlern bleibt die Quelle erhalten und der Fehler wird protokolliert.

Bei unterschiedlichem Inhalt:

- Behalte Quelle und Ziel unverändert.
- Protokolliere einen Fehler.

Berücksichtige auch mehrere Quelldateien, die innerhalb desselben Laufs denselben Zielpfad ergeben.

6. Probelauf

--dry-run darf keine Mediendateien verschieben, löschen oder verlinken und keinen Ziel- oder exists-Ordner anlegen.

Textprotokolle und Metadaten-Cache dürfen geschrieben werden.

Protokolliere geplante Verschiebungen als WOULD_MOVE. Simuliere auch Zielkollisionen innerhalb des Laufs.

7. Protokolle

Erzeuge pro Aufruf einen eigenen Unterordner im Protokollordner mit diesen UTF-8-Tabellen, getrennt durch Tabulatoren:

- moved.txt: Aktion, Quelle, Ziel, Datum, Datumsquelle
- skipped.txt: Quelle und Begründung
- errors.txt: Verarbeitungsfehler

Alle protokollierten Pfade müssen relativ zum Arbeitsverzeichnis beim Programmstart sein, auch in Fehlermeldungen. Bei Verwendung eines Ordnerjahres steht nur das Jahr in der Datumsspalte und „Ordnername“ als Datumsquelle.

8. Leistung und Wiederaufnahme

- Verwende einen persistenten ExifTool-Prozess mit -stay_open.
- Verarbeite Metadaten in Gruppen von bis zu zehn Dateien.
- Behandle Dateinamen mit Zeilenumbrüchen über separate sichere Aufrufe.
- Speichere erfolgreich gelesene Metadaten nach jeder Gruppe in:
PROTOKOLLORDNER/metadata-cache.sqlite3
- Verwende den Cache beim nächsten Lauf für unveränderte Dateien wieder.
- Erkenne Änderungen anhand von Dateiidentität, Größe und Änderungszeiten.
- Speichere fehlgeschlagene Abfragen nicht als erfolgreiche Cache-Einträge.
- Verändere keine Metadaten in den Mediendateien.

- Beende ExifTool zuverlässig bei Fehlern und Strg+C.

9. Fortschritt und Rückgabewerte

Zeige nach der Suche die Gesamtzahl unterstützter Mediendateien an.

Aktualisiere nach jeweils zehn bearbeiteten Dateien dieselbe Konsolenzeile mit Wagenrücklauf, auch bei umgeleiteter Ausgabe:

```
3139/140 Error/Skip/Exists 0/0/0
```

Bedeutung:

Gesamtzahl/bearbeitet, anschließend Fehler/übersprungen/Exists.

Zeige den Endstand auch bei einer unvollständigen Zehnergruppe und schließe ihn mit einem Zeilenumbruch ab.

Exitcodes:

0 = erfolgreich, einschließlich regulär übersprungener Dateien

1 = Verarbeitungsfehler

2 = ungültiger Aufruf

130 = Abbruch mit Strg+C

10. Lieferung

Erstelle:

- collect_files.py mit der Programmlogik
- collectFiles als ausführbaren Starter
- install.sh
- README.md auf Deutsch
- tests/test_collect_files.py mit unittest-Tests

install.sh soll ohne sudo gestartet werden, Python und ExifTool prüfen und fehlende Pakete bei Bedarf über apt-get mit sudo installieren. Installiere einen eigenständigen Befehl nach ~/.local/bin/collectFiles. Unterstütze alternativ --prefix VERZEICHNIS. Wiederholte Installation soll den Befehl aktualisieren.

Teste insbesondere Datumpriorität, Jahresordner, Unterordnerstruktur, Unix-Zeitstempel, Probelauf, identische und unterschiedliche Zielkollisionen, Hardlinks, Kopierfehler, Quellerhaltung bei Fehlern, symbolische Links und Cache-Wiederaufnahme.

Implementiere das Programm vollständig, führe die Tests aus und dokumentiere Installation, Beispiele, Hardlink-Verhalten und Grenzen bei einem harten Abbruch.