Vol.2025-DPS-204 No.8
Vol.2025-EIP-109 No.8
2025/9/18

ユーザのファイル操作を契機とした自動タイムスタンピングシステムによる証跡作成支援手法の提案

大槻 篤史[1]　乃村 能成[2]　嶋吉 隆夫[3]

概要：研究データの正当性を証明するために研究データを管理する必要がある．研究者は，研究データを管理するためにタイムスタンプ付与やデータベースへのアップロードをデータ更新の度に行う必要がある．そこで本稿では，これらの手間を軽減する証跡作成支援システムを設計実装する．提案システムでは，ファイルシステムを監視し，ファイルへの操作を取得する．操作されたファイルのハッシュを作成し，ハッシュと操作時刻，ファイルパスの組を証跡とする．また，ある論文に関して取得した証跡から研究データと論文中の図表の関連を LLM に説明させる．

キーワード：研究データ管理，Model Context Protocol

1. はじめに

研究の主張の組み立てには研究データが利用される．研究データとは，実験データや解析結果などの研究の過程または成果として収集，生成される情報である．研究の確実な積み上げと再現性のために，研究データを研究過程の段階ごとに適切に保存・管理し，研究データの正当性を証明するための説明を示す必要性が生じている [1]．研究データを適切に管理，共有，公開し，正当性を証明するには，安定した研究データ管理基盤が必要である．しかし，このような環境を研究者自身が構築することは非効率である．そこで，研究データ管理システムが開発されている． 研究データ管理システムを用いることで，研究者は統一された手法でデータを管理できる．しかし，既存の研究データ管理システムには，証跡として研究データそのものを保存しているため，ストレージや通信のリソースを消費してしまうという課題がある．また，ユーザにとって操作が手間になるという課題がある．ユーザにとって手間になる操作とは，研究データのアップロードや，研究データの正当性を示すための説明作成といった操作である．ここで，研究データのアップロードは既存システムでは手動であり，操作を忘れるとデータの正当性が証明できない問題がある．また，正当性を示すための説明作成機能は既存システムでは提供されておらず，研究者自身が証拠を収集し，説明を作成しなくてなはらない．

上記の問題を解決するために自動タイムスタンピングシステムを提案する．提案システムでは，利用者のファイルシステムを監視することでファイルへの操作を取得し，操作が行われたファイルの証跡を自動で保存，公開する．ここで，証跡としてファイルそのものではなくファイルのハッシュ値を利用する．また，保存した証跡の情報を利用し，Large Language Model (LLM) および Model Context Protocol (MCP)[2] によって論文中の図表と研究データの関連説明を自動作成する．上記の機能を提供する自動タイムスタンピングシステムによって，既存の証跡保存システムの課題を解決できると考える．

そこで本稿では，提案システムを実現するためにユーザの操作を取得する方法，アップロードの契機とする操作，およびアップロードするデータの形式を検討する．また，研究データとそのデータに基づく論文の図表の関連と，そのデータの証跡について LLM に説明を作成させる機能について，LLM に対してどのようなデータを与えればよいのか検討する．これらの検討した内容を元に自動タイムスタンピングシステムを実装し，評価する．

2. 証跡保存システムの課題とその対処

2.1 証跡保存システムとは

証跡保存システムとは，研究データにタイムスタンプを付与して保存することで証跡を保存し，研究データの存在を証明するシステムである．ここで，研究データとは，研究の過程または成果として収集，生成される情報のことを

[1] 岡山大学大学院環境生命自然科学研究科
[2] 岡山大学学術研究院環境生命自然科学学域
[3] 岡山大学 AI・数理データサイエンスセンター

指す [3]．また証跡とは，ある時点でデータが存在していた証拠のことである．証跡保存システムの例として，国立情報学研究所が提供している GakuNin RDM [*1] がある．このシステムを導入する利点を以下に示す．

(1) 研究データの改ざん防止
証跡として保存した研究データにタイムスタンプを付与することで，保存した時点で研究データが存在したことが証明できる．また，タイムスタンプの時刻以降，研究データが変更されていないことを証明できる．そのため，手元の研究データを変更した場合，証跡として保存したデータと比較することで研究データの改ざんを特定できるため，改ざんを防止できる．

(2) 証跡の保存・保護
研究データにタイムスタンプを付与して保存することで，研究の証明となる研究データを管理できる．また，研究データの履歴が取得できる．これにより，自身の成果がいつから存在していたか証明できる．

(3) 研究データを公開可能
研究者が研究データを論文やプロジェクトの成果として公開できる．そのデータは全世界に公開され第三者が閲覧できるため，論文やプロジェクトの正当性を証明できる．

2.2 証跡保存システムの課題

既存の証跡保存システムに関して，以下の課題があると考えた．

課題 1 研究データを保存するためにストレージや通信のリソースを消費する
既存の証跡保存システムでは，研究データそのものをストレージに保存している．そのため，多くの研究者が研究データを保存すると研究データを保存するためにストレージや通信のリソースを消費してしまう．

課題 2 ユーザにとって操作が手間である
研究データを管理する際に手間になる操作として，以下の 2 種類があげられる．
(1) 証跡のアップロード
(2) 研究データの正当性を証明する説明の作成
まず (1) の証跡のアップロード操作は，既存の研究データ管理システムでは手動で行う必要がある．そのため，ユーザがアップロード操作を忘れる可能性がある．ユーザがアップロード操作を忘れた場合，研究データの証跡が残らないため，研究データの正当性が証明できなくなってしまうという問題がある．次に (2) の研究データの正当性を証明する説明の作成操作は，既存の研究データ管理システムでは提供されていない機能である．研究者自身が説明を作成するため，説明作成の際に各研究データについていつから存在していたのか，どのように作成したのかといった情報を収集しなくてはならない．また，実験や論文の執筆から説明作成までの期間が空いていた場合，どのようなデータを利用したのか，どのように作成したのかといった情報を忘れてしまう可能性がある．

2.3 課題への対処

2.2 節で述べた課題 1，2 への対処手法を提案する．

対処 1 研究データのハッシュを証跡として公開する
研究データのハッシュを証跡として公開し，データそのものは，ユーザが保持，あるいは独自に公開する．ハッシュを公開することで，システムそのもののストレージや通信リソースの消費量を減らせると考える．ハッシュには以下のような特性が有る．
(1) 同じデータからハッシュを作成すると，同じ結果が得られる
(2) ダイジェストを作成した結果が一意である
これらの特性から，証跡として保存したデータが存在していることを証明できる．また，ハッシュを用いることで元のデータのサイズとは関係なく，固定サイズの証跡に変換できる．ここで，ハッシュ作成による手法を採用した場合，ハッシュ作成にかかる時間がどの程度かを検討する必要がある．

対処 2-1 ファイル操作を監視して証跡を自動でアップロードする
証跡を自動でアップロードすることで，ユーザは証跡の保存の手間を軽減でき，忘れることがなくなる．証跡を自動でアップロードする手法として，以下のような手法が考えられる．
(1) 定期的に計算機内の証跡をアップロードする
(2) ファイル操作を監視して操作されたファイルを証跡としてアップロードする
定期的に計算機内の証跡をアップロードする手法では，アップロードした直後に更にファイルを更新する可能性がある．この場合，次の周期まで証跡を保存できないという問題がある．しかし，周期を短くすると，その度に全てのファイルをスキャンするため，計算機に負荷がかかる．そのため，ファイル操作を監視して操作されたファイルを証跡としてアップロードする手法を用いる．ファイル操作を監視することで，操作されたファイルを検知し，証跡をアップロードできる．この手法では，ファイルの更新に対応して証跡をアップロードするため，更新されたファイルを証跡として保存できると考えられる．ここで，どのように意味のある変更を取得するかが課題となる．

対処 2-2 LLM を利用して研究データの説明を作成する
(対処 2-1) の手法により，証跡は自動でアップロード

[*1] https://rcos.nii.ac.jp/service/rdm/

されるが，ユーザが各研究データに対応する証跡を探して説明を作成するのは手間である．そこで，LLM を利用して研究データとそのデータに基づく論文の図表の関連と，そのデータの証跡に関する説明を作成させる．これにより，ユーザは LLM に対して説明作成を依頼するだけで説明を作成でき，ユーザは説明が正しいか確認すればよい．また，ユーザが研究データに関する情報を忘れていても説明を作成できる．ここで，以下の 2 つの点を考慮する必要がある．

(1) LLM にどのような情報を与えればよいのか
(2) (1) の情報を LLM にどのようにして与えるのか

3. 証跡の管理手法

3.1 概要

本章では，2.3 節で述べた対処手法を実現するために以下の項目について説明する．

(1) データベースに保存する証跡のエントリの定義
(2) ファイルアクセス履歴の取得手法
(3) 証跡の外部公開手法
(4) 説明作成に必要な情報およびその提供手法

3.2 データベースに保存する証跡のエントリの定義

研究データがいつから存在していたか証明するためにデータベースに証跡を保存する．証跡としては以下の 3 種類のデータの組を 1 エントリとする．

研究データのハッシュ
　操作が行われたファイルから計算したハッシュ値
更新時刻
　ファイルに対する操作が行われた時刻
ファイルパス
　操作が行われたファイルのパス

データベースのそれぞれのエントリを更新時刻 t ，ハッシュ h ，ファイルパス f の組と表現する．1 つのエントリを e として以下のように表すことができる．

$$e_i = (t_i, h_i, f_i) \tag{1}$$

これらの e の集合 E をデータベースとする．

3.3 ファイルアクセス履歴の取得手法

2.3 節で述べたように，自動で証跡をアップロードする方法として，ユーザのファイル操作を監視して証跡をアップロードする．ファイル操作を監視するためには，ファイル操作の際に発生する情報を取得できれば良い．操作の情報として，操作時刻，ファイルパス，イベントなどが挙げられる．イベントとは，open や write など，ファイル操作に関するシステムコールである．これらの情報をファイルアクセス履歴とする．ファイルアクセス履歴を取得できればユーザの操作内容を把握できると考えられる．そこで，FUSE[4] を利用して独自のファイルシステムを作成する．作成したファイルシステムでは，ファイルに関するシステムコールが発行された際に，ファイルアクセス履歴をログとして出力する．ファイルアクセス履歴には以下の要素が含まれる．

(1) 操作された時刻
(2) 行われた操作
(3) 操作されたファイルのパス

ここで，行われた操作を取得しているため，ファイルが変更される `write` のような特定の操作のみに対応してハッシュを作成する．これにより，意味のある変更のみ取り出せる．

3.4 証跡の外部公開手法

3.4.1 証跡の外部公開手法の比較

証跡のエントリ e を都度外部に自動で公開することは，通信や計算機に対する負荷が無視できない場合がある．提案システムでは，e を生成する契機については，以下の 3 つを想定する．

(1) 論文など，1 ファイルを何度も上書保存する
(2) 写真など，変更はないが，数が増大する
(3) ログなど，1 ファイルへの追記でサイズが増大する

ケース 1 を想定した場合は，ファイルの保存に応じて e を都度公開することでも問題がない．ケース 2 では，ファイル作成の頻度が問題になる．ケース 3 では，毎秒，あるいはそれ以上の頻度での追記が想定されるため，通信や計算機に対する負荷が無視できない．

そこで，提案システムでは，E のうち，ある時間ウィンドウ，例えばある 1 日 d に取得されたエントリの集合 E_d について，E_d のダイジェストを適時に公開することとする．これによって，例えばハッシュ値 1 つを 1 日 1 度外部に公開するだけで済む．

3.4.2 外部への公開手法の実現

証跡を公開後に変更できると，公開後に作成した研究データを公開時点で存在していたように書き換えることができるという問題がある．そこで，証跡を公開するデータベースは，証跡を変更されないようにするために以下の条件を満たす必要がある．

(1) ユーザが証跡に対して行った操作がログとして保存される
(2) ユーザ以外は保存した証跡に対して閲覧操作のみ可能

3.5 説明の作成手法

3.5.1 LLM に提供する情報および提供手法

LLM を利用して研究データとそのデータに基づく論文の図表の関連と，そのデータの証跡に関する説明を作成するには，LLM は以下のような情報を取得する必要がある．

(1) 論文に挿入されている図

(2) 図はどのように作成されたのか
(3) 図の元となるデータの証跡
これらの情報は，原稿や図を作成するためのファイル，提案システムによって保存した証跡から得られる．しかし，LLM にはこういった外部の情報を取得する機能はない．そこで，証跡の情報やファイル内容の読み込みには MCP を利用する．ここでユーザは，図の作成を自動化するためにバッチ処理的なプログラムを作成していると考える．ユーザは各図を作成するためのプログラムを原稿内にコメントとして記載することで，LLM が作成方法の情報を取得できるようにする．MCP とは，LLM アプリケーションと外部のデータソースやツールとの統合を可能にするオープンなプロトコルである．MCP は，LLM アプリケーションと外部のデータソースやツールを統合するための共通インターフェースを定義しており，複数のシステム間で一貫性のある連携を実現する．MCP は，LLM アプリケーションが以下のようなことを実現するための標準的な方法を提供している．
(1) 言語モデルとコンテキスト情報を共有する
(2) AI システムにツールや機能を提供する
(3) 統合やワークフローを構成可能な形で構築する
LLM に対してどのような機能が利用できるか知らせることで，LLM は MCP を介して提供している機能を利用するようになる．この MCP を利用することで，説明を作成する際に LLM のみでは実現不可能な動作を実現するための機能を提供する．

3.5.2 説明の例

最終的に作成したい説明の例を**図 1** に示す．この説明は以下のような要素を示している．
(1) 図の元データ
(2) 元データの証跡
  (a) 元データのハッシュ
  (b) いつから存在したのか
  (c) 存在していた日についての証跡一覧および日次ハッシュ
(3) 図を作成するコマンドなどの図作成方法

3.5.3 説明を作成するための動作

3.5.2 項に示した説明を作成するためのユーザの指示および LLM の動作を以下に示す．
(1) ユーザは，LLM に原稿を送信し，図を探すように指示する．このとき，ユーザは図の作成自動化に利用したバッチ処理的なプログラムを原稿にコメントとして記載している必要がある．ここでは，このプログラムとしてシェルスクリプトを例に説明する．
(2) LLM は，原稿を読み込んで図と対応するシェルスクリプトを見つける
  原稿には以下の Listing 1 のように図について作成に利用したシェルスクリプトをコメントとして記載している．

本論文の図の元データに関する証跡の説明

本論文に関する証跡の説明を以下に示す．

以下に示すハッシュは，SHA-256形式で作成している．
また，データ取得日として示す時間はUTCである．

- 日次ハッシュ保存先: https://github.com/...

論文内の図に関する証跡

図1: 平均気温と電気使用量の関係の図

- 元となるデータ: temp_elec.csv
  - 日付・気温・電気使用量(kWh)のカラムを利用
- データ取得日: 2025-07-31T17:09:09Z
- 図のファイル: relationship_temperature_electricity.pdf
- 元データのハッシュ: e70894b...
- 元データの証跡:
  `2025-07-31T17:09:09.387224,e70894b...`
- データ取得日の証跡ログ: daily_hashlog_2025-07-31.log
- データ取得日の日次ハッシュ: 6e54adc...
  (https://github.com/user/daily_hash/2025-07-31.txt)
- 図の作成方法
  - コマンド:
    `./temp_elec.sh`
  - temp_elec.sh: 図作成のためのシェルスクリプト
  - plot_temp_electricity.py:
    CSVから散布図を作成するPythonスクリプト
  - temp_elec.csv: 元データファイル

図 1: 作成する研究データの説明の例

Listing 1: 原稿に図を挿入している例

```
1  \myinsertfigure{temp_elec}
2  % plot_temp_electricity.sh で作成
```

こういった環境やコマンドを探すことで，LLM が論文の中で利用されている図とその作成に用いるシェルスクリプトを見つける．
(3) ユーザは，LLM にシェルスクリプトの中から元データを推測するように指示する
(4) LLM は，各図についてシェルスクリプトや記載されているプログラムから元データを推測する
シェルスクリプトは以下の Listing 2 のような内容が考えられる．

Listing 2: 図を作成するためのシェルスクリプトの例

```
1  #! /bin/bash
2
3  python3 plot_temp_elec.py t.csv e.csv
```

ここから，図を作成するためのプログラムや，元データのファイルが推測できる．また，図作成用のプログラムは以下の Listing 3 のような内容が考えられる．

Listing 3: 図を作成するためのプログラムの例

```
    with open(temperature_file) as f:
        reader = csv.reader(f)
        next(reader)
        for row in reader:
            date, temp = row[0], row[1]
            temperature_data[date] = temp
```

(5) ユーザは，LLM に各図の元データに関する証跡を集めるように指示する
(6) LLM は，各図の元データに関する証跡を集める
証跡を集める際は以下のような順序で動作する．
(a) 元データのファイルのハッシュを計算する
(b) 計算したハッシュを元に証跡保存データベースから元データが存在した日時のうち最も古いものを検索する
(c) 証跡保存データベースから特定の日付の証跡をファイルに出力する
(d) 元データが存在した日について日次ハッシュを作成する
これらの動作によって，元データの存在していた日時を取得する．また，その日時を証明するために必要なハッシュや日次ハッシュを作成する．
(7) ユーザは，LLM に各図の元データに関する説明を作成するように指示する
(8) LLM は，集めた情報を元に各図の元データに関する説明を作成し，ファイルに出力する

4. 自動タイムスタンピングシステムの設計と実装

4.1 システムの構成

3 章で述べた手法を実現するシステムの構成および処理流れを**図 2** に示す．システムの各構成部について以下で説明する．

証跡収集部： ユーザのファイル操作を監視し，証跡を保存する必要がある操作が行われたファイルに関する証跡を作成，保存する．作成した証跡は証跡保存データベースに保存する．

証跡検索部： ユーザが指定したファイルのハッシュ値と同じハッシュ値が証跡保存データベースにあるかどうかを検索する．見つかった場合にはファイルの情報をユーザに表示する．

日次ハッシュ作成部： 1 日 1 回，その日に取得したエントリの一覧を証跡保存データベースから取得する．取得したエントリの一覧から日次ハッシュを作成し，外部データベースに保存する．

証跡保存データベース： ローカルに存在し，証跡として更新時刻とハッシュ，ファイルパスを保存する．

外部データベース： 日次ハッシュを保存する．保存した日次ハッシュとそれに対応する日時はユーザ以外からも閲覧できる．

MCP ホスト： ユーザからの説明作成依頼を LLM に送信する．また，LLM から tool 利用に必要な情報を取得し，MCP クライアントから tool の実行結果を取得する．

MCP クライアント： MCP ホストから tool 利用に必要な情報を取得し，情報に基づいて必要な tool を呼び出す．呼び出した tool の動作結果を取得し，MCP ホストに送信する．

MCP サーバ： MCP クライアントから取得した tool 利用情報をもとに対応した tool を実行する．実行結果は MCP クライアントに送信する．

LLM： MCP ホストから説明作成依頼を取得し，説明を作成する．また，説明作成に必要な情報を取得するために tool に必要な情報を MCP ホストに送信する．

4.2 システムの処理流れ

図 2 に示したシステムの処理流れを説明する．

証跡の作成 (a1–a3) ユーザは，計算機上でファイルアクセスを行う．証跡収集部は，行うファイルシステムに対するアクセスを監視し，ファイルアクセスがあると操作の情報を取得する．取得したファイル操作のうち，`write` のようなファイル更新に関わる操作について，操作されたファイルのハッシュを作成する．作成したハッシュや，操作時刻，ファイルパスといった情報を 1 エントリとして，証跡保存データベースに保存する．

証跡の検索 (b1–b4) ユーザは，検索したいファイルを証跡検索部に送信する．証跡検索部は，ユーザから検索したいファイルを取得すると，そのファイルのハッシュを作成する．作成したファイル値を証跡データベースから検索し，検索結果をユーザに表示する．

日次ハッシュの作成 (c1–c2) 日次ハッシュ作成部は，1 日に 1 回，証跡保存データベースからその日に保存されたエントリを取得する．取得したエントリの一覧から日次ハッシュを作成し，外部データベースに保存する．

研究データに関する説明の作成 (d1–d12) ユーザは研究データに関する説明の作成を MCP ホスト に依頼する．MCP ホストはこの依頼を LLM に渡し，LLM は説明作成に必要な情報を取得するために利用したい tool を特定する．その際，LLM が要求する tool の利用情報は，MCP ホスト，MCP クライアント，MCP サーバ へと伝達される．MCP サーバは, 受け取った利用情報に基づいて tool を実行し，その結果を MCP クライアント，MCP ホスト，LLM の経路で返す．LLM は，取得した tool の実行結果をもとに研究データに関する説明を作成し，MCP ホストに送信する．MCP

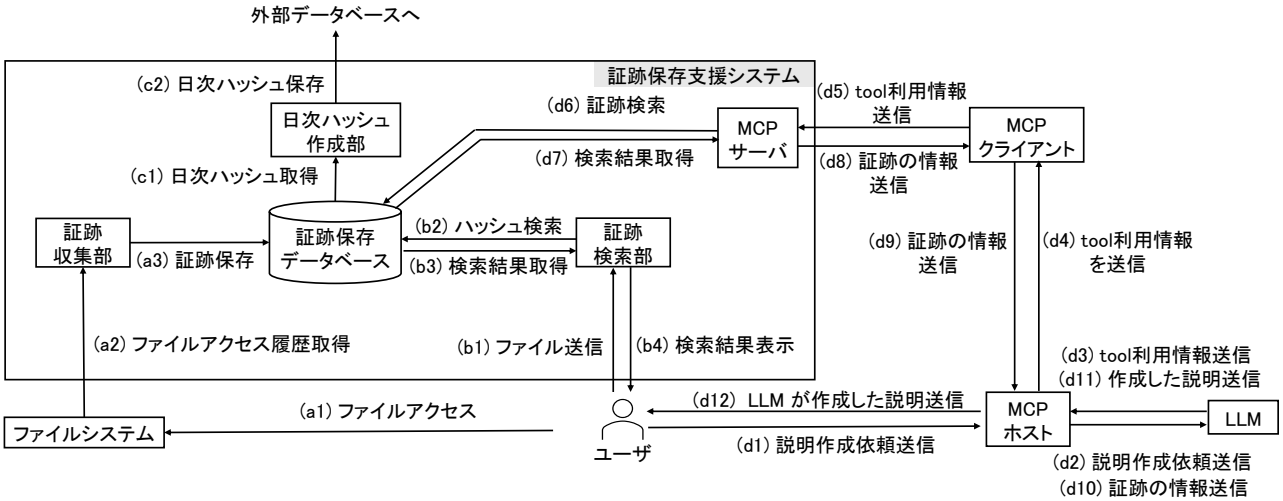

図 2: 自動タイムスタンピングシステムの構成と処理流れ

ホストは，取得した説明をユーザに返却する．

4.3 MCP サーバに必要な機能

3.5.3 項に示した動作の中で LLM のみでは実現できないと考えられる操作は以下の 5 種類である．各操作について，どのような情報を入力として受け取り，どのような情報が出力として必要か示す．

(1) ファイルを読み込む
  (a) 入力: 読み込むファイルのファイルパス (文字列)
  (b) 出力: ファイルの内容 (文字列)
(2) ハッシュを作成する
  (a) 入力: 元データのファイルのファイルパス (文字列)
  (b) 出力: ハッシュを作成した結果 (文字列)
(3) 証跡保存データベースから特定の要素を検索する
  (a) 入力: 元データのファイルのハッシュ値 (文字列)
  (b) 出力: 入力のハッシュ値と一致する全エントリ (文字列)
(4) 証跡保存データベースから特定の日付の証跡をファイルに出力する
  (a) 入力:
    - 元データが存在していた日付 (文字列)
    - 出力先ファイル名 (文字列)
  (b) 出力: 出力先ファイル名を含んだ出力完了メッセージ (文字列)
(5) 元データに関する説明をファイルに出力する
  (a) 入力:
    - 出力先のファイルパス (文字列)
    - 出力する説明の内容 (文字列)
  (b) 出力: 出力先ファイル名を含んだ出力完了メッセージ (文字列)

これらの機能は，ファイル操作やハッシュ作成などの操作が必要になるため LLM のみでは実現できない．この中で，(1),(5) の機能は Filesystem MCP Server[5] というファイル操作に関する機能を提供している MCP サーバを利用することで解決できる．そこで，(2),(3),(4) の機能を提供する MCP サーバ を実装する必要がある．

4.4 実装

図 2 に示した自動タイムスタンピングシステムを実装した．また，4.3 節において，実装する必要があると示した機能を実現するための tool を提供する MCP サーバを実装した．各機能に対応する tool 名とその引数を示す．

(1) ハッシュを作成する tool:
  `create_hash(str filepath)`
(2) 証跡保存データベースから特定の要素を検索する tool:
  `search_target_from_log(str target)`
(3) 証跡保存データベースから特定の日付の証跡をファイルに出力する tool:
  `create_daily_log_file(str day, str path)`

5. 評価

5.1 概要

本節では，以下の 3 つの項目について評価を行う．

ハッシュ作成にかかる時間　提案システムでは，システムそのもののストレージや通信リソースの消費量を減らすために研究データのハッシュを作成して公開している．ここで，ハッシュを作成するのにかかる時間は，研究データが更新される周期よりも短い必要がある．これは，研究データへの操作の周期よりもハッシュ作成にかかる時間が長い場合，ハッシュを作成する時点で，既にファイルが更新されている可能性があるからである．そこで，提案システム起動時にファイルサイズごとにハッシュ作成にどの程度の時間がかかるか計測する．

FUSE による履歴取得の動作確認　提案システムでは，

FUSE を利用して独自のファイルシステムを作成し，ユーザのファイル操作を取得する．ここで，作成したファイルシステムによってユーザのファイル操作を取得できているか確認したい．そこで，一定期間提案システムを起動し，その期間の一部で原稿の執筆を行う．この提案システムの起動時間に対応したファイル操作を取得できているかどうか確認することで，FUSE による履歴取得ができていることを確認する．また，原稿執筆に対応した証跡を証跡保存データベースに保存できているか確認し，証跡を保存するべき操作が行われたファイルの証跡を保存できていることを確認する．

LLM が作成した説明の妥当性　実装した MCP サーバによって，説明を作成するために必要な情報を与えられているかを確認する必要がある．そこで，研究データの説明作成機能により，適当な説明が作成できるか確認する．また，作成された説明を確認することで，LLM に与えた情報が説明作成に適した情報であったかを確認する．

評価環境を**表 1** に示す．

表 1: 評価に使用した計算機の環境

計算機	
OS	Ubuntu 24.04.1 LTS (Noble Numbat)
CPU	AMD Ryzen 5 8640U w/ Radeon 760M Graphics
メモリ	16GB

5.2　評価結果

5.2.1　ハッシュ作成にかかる時間

本計測では，ログファイルに対して，1 秒ごとに 30B のデータを追記する操作を 1 年間続けていた場合について考える．このケースでは，1 年後にファイルは約 950MB になっており，このファイルサイズに対してハッシュ作成にかかる時間が 1 秒以内であればよい．

評価した結果 1GB のファイルのハッシュを計算するのにかかる時間は約 0.5 秒であった．また，ハッシュ作成にかかる時間とファイルサイズの関係は線形的になっており，2GB 程度のファイルであれば 1 秒以内にハッシュを作成できると考える．このことから，提案システムの証跡保存機能は想定した状況のもとで利用可能である．

5.2.2　FUSE による履歴取得の動作確認

以下に提案システムを起動する時間と，その期間中に原稿を執筆した時間を示す．

(1) 起動時間: 24 時間
(2) 原稿執筆: 3 時間

実行した結果，システムの起動中に 2,669,068,835 回のファイルアクセス履歴が記録されていた．ファイルアクセス履歴は，評価のためにシステムを起動した直後から停止する直前までの履歴が記録されていた．このことから，FUSE によって適切にファイル操作を監視し，ファイルアクセス履歴を記録できていると考える．また，原稿執筆を行った 3 時間の間に原稿に関する証跡が 781 回証跡保存データベースに保存されていた．このことから，研究データが更新される操作に対して適切に証跡を保存できているといえる．

5.2.3　説明作成機能の妥当性

以下の条件の研究データに対して動作を確認する．

(1) 元データ: temp_elec.csv
(2) 元データの作成日: 2025-07-31
(3) 図を作成するためのシェルスクリプト: temp_elec.sh

想定したデータに対して作成された説明が図 1 である．この説明では，想定したデータについての説明を記載できており，証跡についても正しい値を記載している．このことから，説明作成機能によって作成される説明は妥当である．

6.　関連研究

文献 [6] では，研究データ管理において研究者による手動でのデータ転送が非効率性や，データ品質の低下を招く可能性があることを問題点として挙げている．そして、標準化された研究データ管理システムを提案し，データベースの一貫性と正規化を保証し，手動操作に伴う非効率性を軽減することで，データ品質の向上と効率的なデータ収集を目指している．これは，本稿で課題としている，既存の研究データ管理システムにおいてユーザの手間となる手動操作や，操作の失念によるデータ証跡保存の漏れとも共通する課題である．このような手動依存の運用は，データ収集効率を低下させるだけでなく，データの欠損を引き起こす要因となりうる．したがって，研究データ管理においては，データ管理の自動化を進めることが重要である．

また，文献 [7] では，いくつかの分野の研究者が研究データ管理システムに求めている機能やサポートについて調査している．調査結果として研究者が求めている機能の 1 つにメタデータの自動作成があげられている．これにより，本稿で提案している LLM を利用した研究データの説明作成機能は研究者にとって有益な機能であると考えられる．

7.　おわりに

本研究では，ユーザのファイル操作を契機とした自動タイムスタンピングシステムを提案した．まず，既存の研究データ管理システムの問題点として，ストレージや通信のリソースを消費してしまうことと，証跡のアップロードやメタデータの付与が手動であり，ユーザが操作を忘れる可能性があることを示した．次に，これらの問題点を解決する手法として，ユーザのファイル操作を監視することで，操作されたファイルのハッシュを証跡として保存する手法

を提案した．この手法を実現するために証跡を外部に公開する手法を比較した．比較した結果，提案システムでは，ある日に取得された証跡のエントリの集合から作成したダイジェストと作成した日時を保存する手法を用いた．また，提案手法を用いて保存した証跡を用いて，LLM に研究データの正当性を示すための説明を作成させる機能を提案した．そして，これらの機能を有した提案システムの構成を示し，設計，実装した．

残された課題として，履歴変換部の実装がある．現在は，履歴変換は行わず取得したファイルアクセス履歴のうち，`read` と `getattr` の操作以外全てに対応してハッシュを作成している．そのため，本来はハッシュを作成する必要がない操作に対してもハッシュを作成している．また，提案システムを利用した際の計算機への負荷の評価がある．システムの起動時と非起動時の CPU 使用率やメモリの使用量を比較することで，提案システムが実際に利用できるかどうかを評価する必要がある．

参考文献

[1] 大学 ICT 推進協議会：学術機関における研究データ管理に関する提言，大学 ICT 推進協議会（オンライン），入手先 ⟨`https://rdm.axies.jp/sig/57/`⟩（参照 2025-08-07）．
[2] Anthropic: Introduction - Model Context Protocol, Anthropic (online), available from ⟨https://modelcontextprotocol.io/introduction⟩ (accessed 2025-08-07).
[3] 日本学術会：オープンサイエンスの深化と推進に向けて，日本学術会（オンライン），入手先 ⟨`https://www.scj.go.jp/ja/info/kohyo/pdf/kohyo-24-t291-1.pdf`⟩（参照 2025-08-07）．
[4] The kernel development community: FUSE, The kernel development community (online), available from ⟨https://www.kernel.org/doc/html/latest/filesystems/fuse.html⟩ (accessed 2025-08-07).
[5] ModelContextProtocol: Filesystem MCP Server, ModelContextProtocol (online), available from ⟨https://github.com/modelcontextprotocol/servers/tree/main/src/filesystem⟩ (accessed 2025-08-07).
[6] Müller, J., Heiss, K. and Oberhoffer, R.: Implementation of an open adoption research data management system for clinical studies, *BMC Research Notes*, Vol. 10 (online), DOI: 10.1186/s13104-017-2566-0 (2017).
[7] Donaldson, D. R. and Koepke, J. W.: A focus groups study on data sharing and research data management, *Scientific Data*, Vol. 9 (online), DOI: 10.1038/s41597-022-01428-w (2022).