



Articles / Desktop Programming / Win32

★ C++

★ Win32

★ SQLite

Displaying recent browsing history



Michael Haephrati

30 Sep 2024 CPOL 3 min read

👁 3.1K 📄 51 📌 6

This article explains how and where browsing history is stored and how to fetch it.

This article is about browsing history and focuses on the commonly used browsers: Edge and Chrome. In this article I will explain how to get the recent browsing history, in this case, the history of the last 10 minutes, along with the dates and times in UTC.

Download source - 1.5 MB

Github repo: <https://github.com/securedglobe/browsinghistory>

Introduction

```
Administrator: Command Prompt
C:\Users\micha\source\repos\browsinghistory\Debug\LatestBrowsingHistory
Checking Chrome browsing history:
URL: https://edition.cnn.com/, Visit Time (UTC): 2024-09-29 19:29:11
URL: https://www.cnn.com/, Visit Time (UTC): 2024-09-29 19:29:11
URL: http://www.cnn.com/, Visit Time (UTC): 2024-09-29 19:29:11
URL: https://www.skyscanner.net/?previousCultureSource=GE0_LOCATION&redirectedFrom=www.skyscanner.com, Visit Time (UTC): 2024-09-29 19:28:27
URL: https://www.skyscanner.net/?previousCultureSource=GE0_LOCATION&redirectedFrom=www.skyscanner.com, Visit Time (UTC): 2024-09-29 19:28:27
URL: https://www.skyscanner.net/?previousCultureSource=GE0_LOCATION&redirectedFrom=www.skyscanner.com, Visit Time (UTC): 2024-09-29 19:28:26
URL: https://www.skyscanner.net/?fbgroup=61404588&previousCultureSource=GE0_LOCATION&redirectedFrom=www.skyscanner.com, Visit Time (UTC): 2024-09-29 19:28:26
URL: https://www.skyscanner.com/, Visit Time (UTC): 2024-09-29 19:28:26
URL: https://www.amazon.com/, Visit Time (UTC): 2024-09-29 19:27:50
URL: https://www.google.com/, Visit Time (UTC): 2024-09-29 19:27:39
URL: https://www.google.com/, Visit Time (UTC): 2024-09-29 19:27:39
URL: https://google.com/, Visit Time (UTC): 2024-09-29 19:27:39
URL: https://gator3309.hostgator.com:2096/cpsess6053250007/3rdparty/roundcube/?_task=mail&_mbox=INBOX, Visit Time (UTC): 2024-09-29 19:26:53
URL: https://gator3309.hostgator.com:2096/cpsess6053250007/3rdparty/roundcube/?_task=mail&refresh=1&uid=11025&_mbox=INBOX&_page=, Visit Time (UTC): 2024-09-29 19:26:53
URL: https://gator3309.hostgator.com:2096/cpsess6053250007/3rdparty/roundcube/?_task=mail&caps=pdf%3D1%2CFlash%3D0%2Ctiff%3D0%2Cwebp%3D1&uid=11025&_mbox=INBOX&_action=show, Visit Time (UTC): 2024-09-29 19:26:31
Checking Edge browsing history:
URL: https://www.securedglobe.net/, Visit Time (UTC): 2024-09-29 19:32:12
URL: https://www.securedglobe.net/category/all-products, Visit Time (UTC): 2024-09-29 19:32:05
URL: https://www.visualstudio.com/, Visit Time (UTC): 2024-09-29 19:31:46
URL: https://www.visualstudio.com/, Visit Time (UTC): 2024-09-29 19:31:46
URL: https://visualstudio.com/, Visit Time (UTC): 2024-09-29 19:31:46
URL: https://www.codeproject.com/, Visit Time (UTC): 2024-09-29 19:29:31
URL: https://www.ibm.com/uk-en, Visit Time (UTC): 2024-09-29 19:29:19
URL: https://www.ibm.com/gb-en, Visit Time (UTC): 2024-09-29 19:29:19
URL: https://www.ibm.com/, Visit Time (UTC): 2024-09-29 19:29:19
URL: https://github.com/, Visit Time (UTC): 2024-09-29 19:28:03
URL: https://github.com/, Visit Time (UTC): 2024-09-29 19:28:02
URL: https://github.com/, Visit Time (UTC): 2024-09-29 19:28:02
URL: https://github.com/, Visit Time (UTC): 2024-09-29 19:28:01
URL: https://amecdn.msftauth.net/ms/calligraphicode=1.C557_B12.2.U.b6ec58d8-387a-e3ad-ee72-97e797bcfe39&state=d4fadee9-8479-44c8-baf0-092cfe940958, Visit Time (UTC): 2024-09-29 19:27:30
URL: https://login.live.com/oauth20_authorize.srf?client_id=7eadcef8-456d-4611-9480-4fff72b8b9e2&scope=user_read&redirect_uri=https%3a%2f%2faccount.msftauth.net%2fme%2fcalligraphicode%26response_type=code&state=d4fadee9-8479-44c8-baf0-092cfe940958&response_mode=fragment&prompt=none&login_hint=securedglobe&code_challenge=OARQmQy_BNSZfUg_FmMmaDKX18m6Pnx1YqCrsVpY8code_challenge_method=S256&uid=e03474b57314f4bd269b1e713179&msproxy=1&issuer=msoidtenant=common&ui_locales=en-US&epct=PAQABDgEAAADW6j131mB3T7ugWTT8pFeyFRZTvx2VckypHG8mZRoNB0DrNdXRgt22PUCPXVma5mbvK1BK5160QwD13K2pEILXfUJ-1QXw5-MYh-19HFCQR8O6U3abDTokD-BE7V1C1v0y3bmeF6K10P9R5mOKTgrrpY6ahesOR5ydv43k43-MG122p4gKnmHx5ul2YgS4mYvRXI41ti4y40ruRkZ2TDWiqG1J28e58CAA8jshs=-1&jshe=8jshp=, Visit Time (UTC): 2024-09-29 19:27:30
URL: https://www.microsoft.com/en-gb/, Visit Time (UTC): 2024-09-29 19:27:27
URL: https://www.microsoft.com/, Visit Time (UTC): 2024-09-29 19:27:27
URL: http://www.microsoft.com/, Visit Time (UTC): 2024-09-29 19:27:27
```

When trying to get the browsing history, there are several questions:

- Where is it stored?
- How frequent is it updated?
- In what format is it stored?
- What is the best method to access that information?

Microsoft Edge

Microsoft Edge stores its history in "**<User profile>\AppData\Local\Microsoft\Edge\User Data\Default\History**". It uses a table named **urls** to store each visit, excluding **Incognito mode**.

From the programming view, if you have found the user's profile path and stored it in *userProfilePath*, the database path will be:

```
userProfilePath + "\\AppData\\Local\\Microsoft\\Edge\\User Data\\Default\\History"
```

Chrome

Google Chrome stores its history in "**<User profile>\AppData\Local\Google\Chrome\User Data\Default\History**". It uses a table named **urls** to store each visit, excluding **Incognito mode**.

From the programming view, if you have found the user's profile path and stored it in *userProfilePath*, the database path will be:

```
userProfilePath + "\\AppData\\Local\\Google\\Chrome\\User Data\\Default\\History"
```

The building blocks

For the purpose of this article, I created a Console application using [Visual Studio 2022](#).

Database

Since the browsing history is stored in an [sqlite3](#) database, we need to include the **sqlite3** library or source code files.

To be able to read the browsing history database, we need to be able to handle **sqlite3** databases. You can find further information about **sqlite3** [here](#). We then need to add **sqlite3.c** and **sqlite3.h** to the project.

User's Profile Folder

All locations are relative to the current [user's profile](#) path in the **c:\users** folder. We first need to find that path. Here is a function that does that:

```
// Get the current user's profile path (e.g., C:\Users\\)
std::wstring GetUserProfilePath()
{
    WCHAR path[MAX_PATH];
    if (SUCCEEDED(SHGetFolderPathW(NULL, CSIDL_PROFILE, NULL, 0, path)))
    {
        return std::wstring(path);
    }
    return L"";
}
```

Using the database while being locked

Since the databases may be locked, and they will be locked if **Edge** or **Chrome** are running, we first need to copy them to another location and access them from there.

```
// Function to copy the locked database to a temporary file for querying
bool CopyDatabaseToTemp(const std::wstring& dbPath, std::wstring& tempDbPath)
```

```

{
    wchar_t tempPath[MAX_PATH];
    if (GetTempPathW(MAX_PATH, tempPath) == 0)
    {
        return false;
    }

    wchar_t tempFileName[MAX_PATH];
    if (GetTempFileNameW(tempPath, L"dbcopy", 0, tempFileName) == 0)
    {
        return false;
    }

    tempDbPath = std::wstring(tempFileName);

    try
    {
        std::filesystem::copy_file(dbPath, tempDbPath,
std::filesystem::copy_options::overwrite_existing);
        return true;
    }
    catch (const std::filesystem::filesystem_error& e)
    {
        wprintf(L"Failed to copy database: %s\n", ConvertUtf8ToWide(e.what()).c_str());
        return false;
    }
}

```

Handling Unix dates and times

There are several ways to store dates and times.

WebKit Epoch

The WebKit timestamp starts from January 1, 1601 (UTC).

Unix Epoch

The Unix timestamp starts from January 1, 1970 (UTC).

Converting between WebKit Epoch to Unix Epoch

We use the following function for the conversion. **ConvertWebKitToUnixTime()**

Here is how it works:

Parameters:

webkitTime: This is an *int64_t* value representing a timestamp in microseconds since the WebKit epoch (January 1, 1601).

Return Type:

The function returns a *time_t* value, which represents the Unix timestamp in seconds since the Unix epoch (January 1, 1970).

The Logic:

The *WebKit* timestamp is in microseconds, while the *Unix* timestamp is in seconds. By dividing *webkitTime* by 1,000,000, we convert microseconds to seconds.

Adjusting for the Epoch Difference:

The difference between the two epochs (January 1, 1601, and January 1, 1970) is 369 years.

This difference translates to 11644473600 seconds.

The `- 11644473600LL` in the calculation adjusts the timestamp from the *WebKit* epoch to the Unix epoch.

Final Computation:

```
static_cast<time_t>(webkitTime / 1000000 - 11644473600LL);
```

It takes the *WebKit* timestamp in microseconds, converts it to seconds, and then subtracts the difference in seconds between the two epochs to yield the correct *Unix* timestamp.

Here is the function:

```
// Convert WebKit timestamp (microseconds) to Unix timestamp (seconds)
time_t ConvertWebKitToUnixTime(int64_t webkitTime)
{
    return static_cast<time_t>(webkitTime / 1000000 - 11644473600LL); // Adjusting for
    WebKit epoch
}
```

There is also some code for printing the results in human readable format, in my case, I display them in UTC.

```
// Convert time_t to human-readable UTC time string
std::wstring FormatUnixTimeToUTC(time_t unixTime)
{
    struct tm timeInfo;
    if (gmtime_s(&timeInfo, &unixTime) != 0) // Safe version of gmtime
    {
        return L"Invalid time";
    }
}
```

```
wchar_t buffer[80];
wcsftime(buffer, sizeof(buffer), L"%Y-%m-%d %H:%M:%S", &timeInfo); // Format time
return std::wstring(buffer);
}
```

Printing the recent browsing history

In the source code below, we print both Edge and Chrome's recent browsing history.

C++

```
// Function to read browsing history from a given database path
void PrintUrlsFromDatabase(const std::wstring& dbPath, const time_t currentTime, const
time_t timeRangeInSeconds)
{
    std::wstring tempDbPath;
    if (!CopyDatabaseToTemp(dbPath, tempDbPath))
    {
        wprintf(L"Failed to copy database to temporary file: %s\n", dbPath.c_str());
        return;
    }

    sqlite3* db;
    if (sqlite3_open16(tempDbPath.c_str(), &db) != SQLITE_OK)
    {
        wprintf(L"Failed to open database: %s\n", tempDbPath.c_str());
        return;
    }

    // Query to get URLs and visit times
    const char* query = "SELECT u.url, v.visit_time FROM urls u JOIN visits v ON u.id =
v.url ORDER BY v.visit_time DESC;";

    sqlite3_stmt* stmt;
    if (sqlite3_prepare_v2(db, query, -1, &stmt, nullptr) != SQLITE_OK)
    {
        wprintf(L"Failed to prepare statement: %S\n", sqlite3_errmsg(db));
        sqlite3_close(db);
        return;
    }

    // Execute the query and process the results
    while (sqlite3_step(stmt) == SQLITE_ROW)
    {
        const char* foundUrlUtf8 = reinterpret_cast<const char*>(sqlite3_column_text(stmt,
0));

        int64_t visitTimeWebKit = sqlite3_column_int64(stmt, 1);
        time_t visitTimeUnix = ConvertWebKitToUnixTime(visitTimeWebKit);

        // Check if the URL was visited within the last 10 minutes
        if (difftime(currentTime, visitTimeUnix) <= timeRangeInSeconds)
        {
            // Convert the URL from UTF-8 to wide string using the Windows API function
```

```
std::wstring foundUrl = ConvertUtf8ToWide(foundUrlUtf8);

// Format the visit time to a human-readable UTC string
std::wstring visitTimeStr = FormatUnixTimeToUTC(visitTimeUnix);

wprintf(L"URL: %s, Visit Time (UTC): %s\n", foundUrl.c_str(),
visitTimeStr.c_str());
    }
}

sqlite3_finalize(stmt);
sqlite3_close(db);

// Remove the temporary file after use
std::filesystem::remove(tempDbPath);
}
```

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOLE\)](#)

Written By

Michael Haephrati

CEO Secured Globe, Inc.

 United States

Michael Haephrati is a music composer, an inventor and an expert specializes in software development and information security, who has built a unique perspective which combines technology and the end user experience. He is the author of a [the book Learning C++](#) , which teaches C++ 20, and was published in August 2022.


He is the CEO of [Secured Globe, Inc.](#), and also active at [Stack Overflow](#).

Read our [Corporate blog](#) or read my [Personal blog](#).

Follow [@haephrati](#)



Comments and Discussions

 **1 message** has been posted for this article Visit <https://www.codeproject.com/Articles/5388718/Displaying-recent-browsing-history> to post and view comments on this article, or click [here](#) to get a print view with messages.

[Permalink](#)
[Advertise](#)
[Privacy](#)
[Cookies](#)
[Terms of Use](#)

Posted 30 Sep 2024

Article Copyright 2024 by Michael
Haephrati
Everything else Copyright ©
[CodeProject](#), 1999-2024

Web01 2.8:2024-07-22:1