Apache Logs Analysis User Manual

Introduction

This is a user manual for the Apache Logs Analysis tool. This tool is used to analyze Apache logs and generate reports. The report can be configured to contain the desired information with optional filters.

Background information

The web is based on an exchange of data between a client (the browser) and a server. The client makes a request via the HTTP (HyperText Transfer Protocol).

This request is transmitted via the network to the server. The server makes in return a response consisting in particular of a return code and a document. When the response is an document, the client has the possibility to click on a link in order to make another request, the very principle of navigation. All these operations are recorded at the web server level in the form of a log file.

In this project, we propose some data wrangling operations to exploit the information contained in the log files of a web server.

Getting started

Prerequisites

- Unix-like operating system
- g++ compiler required
- log file for analysis
- proper access permissions for the read and write files

Compiling the program

To compile, simply `cd` to the `src` directory and run the following command:

```
$ make
```

To remove the object files and the executable file, run the following command:

```
$ make clean
```

When properly compiled, the executable file will be named `analog` and will be located in the `bin` directory.

Running the program

When running the executable analog file, you should enter the name of the log file you want to analyze.

Normally, the command looks like this:

```
$ ./analog [options] <log_file>
```

Where [options] are the options you want to use and `<log_file>` is the name of the log file you want to analyze. When options are not specified, the program will generate a report with default settings.

For example, if you want to analyze the log file called "access.log" with default settings, you should enter the following command:

```
$ ./analog access.log
```

Attention: If the log file is not in the same directory as the executable file, you should enter the full path of the log file.

Note: There are some predefined log files in the `tmp` directory for your convenience to test the program. (`anonyme.log` is advised against because it contains huge amounts of data)

Options

The following options are available:

```
[-g <dot_file>]
```

Generate a dot file for the graph. The dot file can be used to generate a graph using Graphviz.

Attention: If you want to generate a dot file in a different directory, you should enter the full path of the dot file.

For example, if you want to generate a dot file called "graph.dot", you should enter the following command:

```
$ ./analog -g graph.dot <log_file>
```

```
[-e]
```

Exclude all files with extensions of type image, css, or javascript.

```
[-t <hour>]
```

This options only takes hits in the time interval [hour, hour+1[, for example, if you want to analyze the hits between 10 and 11 (24-hour format), you should enter the following command:

```
$ ./analog -t 10 <log_file>
```

Error and warning messages

The program will display error or warning messages when the command line input is invalid or when the file handling breaks.

- Options
    - Command not recognized
- File handling
    - Log file can't be opened

    - Output more than 1 file
    - Input more than 1 file

- Time constraints

    - Time not recognized
    - Time not within interval
    - No `[-t]` option but time specified

- Graph

    - No output file specified
    - No `[-g]` option but output file specified

Underlying rules for data wrangling

We only consider valid requests, i.e. requests with return code neither 4xx nor 5xx.

Some long urls are shortened to make the graph more readable. For example, `/SiteWebIF/Intranet-etudiant.php?ticket=ST-19118-EabntXV7HXc1iIbI5sou-dsi-vm04` is shortened to `/SiteWebIF/Intranet-etudiant.php?`, string before the '?' character.

Test framework

To test the program, we have created a test framework located in the Tests directory. Starting from Test8 (Previous tests are templates for reference only), we have provided 19 tests for the program. All tests have been pre-run and have passed in MacOS 12.6.

Important: The test framework doesn't work with Ubuntu 20.04, as there is minor differences in the arrangement of nodes with the same number of responses, which doesn't at all affect the correctness of the program.

To run individual tests, `cd` to the `Tests` directory and run the following command:

```
$ ./test.sh <test_name>
```

To run all tests, type

```
$ ./mktest.sh
```

This will run all tests and generate a report in `results.csv` in the `Tests` directory.

Some urls are empty or only contain a "-". Nevertheless, we take them into account in the analysis.

Limitations

There are limitations to the project, as it only supports log file as read file, and only supports the output file as dot file.

Furthermore, limited tests are performed on the program. The program may not work properly due to unexpected inputs.