Log Analyzer Project Documentation

Team
KMVR Madhava Krishna - CS22B1005
Sai Pranav - CS22B1027
Sugreev S - CS22B1028
Chaandini Viswanathan - CS22B2003

November 18, 2024

Contents

1 Overview

The Log Analyzer is a C-based command-line tool that processes log files and provides various analysis capabilities including filtering, statistics generation, and data export to CSV. It supports parsing log entries with timestamps, log levels, and messages, offering flexible filtering options based on dates, times, and log levels.

2 Building the Project

The project uses Flex for lexical analysis and requires a C compiler.

2.1 Prerequisites

- GCC compiler
- Flex (Fast Lexical Analyzer)
- Make utility

2.2 Compilation Steps

1. Save the lexer code to `log_analyzer.l`
2. Generate the lexer:

```
flex log_analyzer.l
```

3. Compile the program:

```
gcc lex.yy.c -o log_analyzer -lfl
```

3 Log File Format

The tool expects logs in the following format:

```
[YYYY-MM-DD HH:MM:SS] LEVEL: Message
```

Example:

```
[2024-11-17 12:35:00] INFO: System startup complete
[2024-11-17 12:36:15] ERROR: Database connection failed
[2024-11-17 12:37:45] WARNING: High memory usage detected
```

4 Features and Usage

4.1 Basic Log Analysis

```
./log_analyzer logs.txt
```

- Displays overall statistics for all log entries
- Shows count by log level (ERROR, INFO, WARNING, DEBUG)

Implementation:

- Uses `print_log_statistics()` function
- Maintains counters for each log level
- Processes all entries in the `log_entries` array

4.2 Date-Specific Statistics

```
./log_analyzer logs.txt stats '2024-11-17'
```

Implementation:

- Uses `print_daily_statistics()` function
- Filters entries matching the specified date
- Uses string comparison for date matching
- Maintains separate counters for filtered results

4.3 Filtering Capabilities

```
./log_analyzer logs.txt filter "expression"
```

Supported filters:

- Date filters:

```
./log_analyzer logs.txt filter "date='2024-11-17'"
./log_analyzer logs.txt filter "date>='2024-11-17'
  ↪ and date<='2024-11-19'"
```

- Time filters:

```
./log_analyzer logs.txt filter "time>='12:35' and
  ↪ time<='16:45'"
```

- Log level filters:

```
./log_analyzer logs.txt filter "level='ERROR'"
```

- Combined filters:

```
./log_analyzer logs.txt filter "date='2024-11-17'
  ↪ and level='ERROR'"
```

Implementation:

```
typedef struct {
    char type[20];      // "date" or "time" or "level"
    char op[3];         // "=", ">=", "<="
    char value[30];
} FilterCondition;
```

- Parses filter expressions using `parse_filter_condition()`
- Supports multiple conditions with AND logic
- Uses `compare_dates_advanced()` for proper date comparison
- Validates dates using `validate_date()`

5 Technical Implementation Details

5.1 Data Structures

```
typedef struct {
    char timestamp[30];
    char date[11];     // YYYY-MM-DD
    char time[9];      // HH:MM:SS
    char level[10];
    char message[256];
} LogEntry;
```

5.2 Lexical Analysis

Uses Flex for parsing log entries with regular expressions:

```
"["[0-9][0-9][0-9][0-9]"-"[0-9][0-9]"-"[0-9][0-9]" "
[0-9][0-9]":"[0-9][0-9]":"[0-9][0-9]"]"[ ]
("ERROR"|"INFO"|"WARNING"|"DEBUG")":"[ ]
```

6 Error Handling

- File opening errors
- Invalid date formats
- Malformed filter expressions
- CSV file creation errors
- Maximum entry limit (1000 logs)

7 Limitations

- Maximum 1000 log entries
- Maximum 10 filter conditions
- Fixed log format
- No support for custom date formats
- Single file processing only

8 Future Enhancements

1. Support for multiple log formats
2. Configurable maximum limits
3. Support for OR conditions in filters
4. Time zone handling
5. Regular expression support in message filtering
6. Multiple file processing
7. JSON export option
8. Custom date format support

9 Example Use Cases

9.1 Error Analysis

```
./log_analyzer logs.txt filter "date>='2024-11-17' and date<='2024-11-19' \
  and level='ERROR'" into "errors.csv"
```

9.2 Daily Monitoring

```
# Get today's statistics
./log_analyzer logs.txt stats '2024-11-18' into "daily_report.csv"
```

9.3 Time Window Analysis

```
# Analyze peak hours
./log_analyzer logs.txt filter "time>='09:00' and time<='17:00'" \
  into "business_hours.csv"
```