МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ
«Санкт-Петербургский национальный исследовательский
университет информационных технологий, механики и оптики»
Факультет информационных технологий и программирования
Кафедра информационных систем

Программирование
Лабораторная работа № 3

Выполнил студент:
Орлов Александр

Группа: M3107

САНКТ-ПЕТЕРБУРГ
2021

Задача
Вам предоставлен access.log одного из серверов NASA (скачать). Это текстовый файл, каждая строка которого имеет следующий формат:
$remote_addr - - [$local_time] “$request” $status $bytes_send
$remote_addr - источник запроса
$local_time - время запроса
$request - запрос
$status - статус ответ
$bytes_send - количество переданных в ответе байт
Например:
198.112.92.15 - - [03/Jul/1995:10:50:02 -0400] "GET /shuttle/countdown/ HTTP/1.0" 200 3985
Требуется
1. Подготовить список запросов, которые закончились 5xx ошибкой, с количеством неудачных запросов
2. Найти временное окно (длительностью параметризуются), когда количество запросов на сервер было максимально

main.c:
```c
/* Orlov Aleksandr, 12-5, M3107,  29.10.2021 */

#include <stdio.h>
#include <string.h>
#include "nasa.h"

int main(void)
{
    long errors = 0, window;
    Timestamp* tail, * head;
    char file[FILENAME_SIZE];

    printf("Input filename: ");
    scanf("%s", file);

    FILE* f = fopen(file, "r");

    printf("Input window size in seconds: ");
    scanf("%d", &window);
    if (f != NULL)
    {
        parse(f, &errors, window);
        printf("Bad requests: %i", errors);
    }
    else
        printf("Problems with the file\n");
    return 0;
}
```

nasa.c:
```c
/* Orlov Aleksandr, 12-5, M3107,  29.10.2021 */
```

```c
#include <stdio.h>
#include <time.h>
#include <math.h>
#include <string.h>
#include "nasa.h"

long parse_time(char* time_string)
{
    char time_format[7] = "//::: ", parsed_time[7][6] = { "" };
    int i = 0, time_type = 0, index = 0, month = 0;
    char months[12][4] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep",
"Oct", "Nov", "Dec" };

    while (time_string[i])
    {
        if (time_string[i] == time_format[time_type])
            ++time_type, index = 0;
        else
            parsed_time[time_type][index++] = time_string[i];
        ++i;
    }

    for (int j = 0; j < 12; j++)
        if (!strcmp(parsed_time[1], months[j]))
            month = j;

    struct tm tdate = {
        .tm_mday = atoi(parsed_time[0]),
        .tm_mon = month,
        .tm_year = atoi(parsed_time[2]) - 1900,
        .tm_hour = atoi(parsed_time[3]),
        .tm_min = atoi(parsed_time[4]),
        .tm_sec = atoi(parsed_time[5]),
        .tm_isdst = 0
    };

    time_t time = mktime(&tdate);

    return time;
}

char* transform_seconds(long timestamp)
{
    char time_string[TIME_STRING_SIZE];
    int i;
    struct tm* u;

    u = localtime(&timestamp);
    i = strftime(time_string, TIME_STRING_SIZE, "%d/%m/%Y:%H:%M:%S", u);
    time_string[i] = '\0';
    return time_string;

}

Timestamp* add(Timestamp* head, long timestamp)
{
    Timestamp* value = (Timestamp*)malloc(sizeof(Timestamp));

    value->timestamp = timestamp;
    value->next = NULL;
    head->next = value;
    head = value;
    return value;
}
```

```
Timestamp* pop(Timestamp* tail)
{
    Timestamp* temp = tail;

    tail = tail->next;
    free(temp);
    return tail;
}

void parse(FILE* f, long* errors, long window)
{
    char request[MAX_REQUEST_SIZE];
    int number = 0;
    Timestamp* tail = NULL, * head = NULL;
    struct max {
        long left;
        long right;
        long window;
        long number;
    } max;

    max.window = 0;
    max.number = 0;

    while (fgets(request, MAX_REQUEST_SIZE, f))
    {
        char time_string[TIME_STRING_SIZE];
        char request_body[MAX_REQUEST_SIZE];
        int j = 0, k = 0, len;

        len = strlen(request);

        while (j < MAX_REQUEST_SIZE && request[j++] != '[');

        while (j < MAX_REQUEST_SIZE && request[j] != ']')
            time_string[k++] = request[j++];

        time_string[k] = '\0';

        if (tail == NULL)
        {
            tail = (Timestamp*)malloc(sizeof(Timestamp));
            tail->timestamp = parse_time(time_string);
            tail->next = NULL;
            head = tail;
            number++;
        }
        else
        {
            long delta = head->timestamp - tail->timestamp;
            if (delta <= window)
            {
                if (number > max.number)
                {
                    max.left = tail->timestamp;
                    max.right = head->timestamp;
                    max.number = number;
                    max.window = delta;
                }
                head = add(head, parse_time(time_string));
                number++;
            }
            else
            {
```

```
                head = add(head, parse_time(time_string));
                tail = pop(tail);
            }
        }
        k = 0;

        while (j < MAX_REQUEST_SIZE && request[j++] != '"');

        while (j < MAX_REQUEST_SIZE && request[j] != '"')
            request_body[k++] = request[j++];

        request_body[k] = '\0';

        if ((j + 2) < MAX_REQUEST_SIZE && request[j + 2] == '5') {
            *errors += 1;
            printf("%s\n", request_body);
        }
    }

    if (head != NULL && tail != NULL)
    {
        long delta = head->timestamp - tail->timestamp;

        if (delta <= window)
            if (number > max.number)
            {
                max.left = tail->timestamp;
                max.right = head->timestamp;
                max.number = number;
                max.window = delta;
            }
    }
    printf("Max number of requests %d from %d to %d\n", max.number, max.left, max.right);
    //printf("Max number of requests %d from %s to %s\n", max.number,
transform_seconds(max.left), transform_seconds(max.right));
    return;
}
```

nasa.h:

```
/* Orlov Aleksandr, 12-5, M3107,  29.10.2021 */

#define MAX_REQUEST_SIZE 5000
#define TIME_STRING_SIZE 40
#define FILENAME_SIZE 261

typedef struct {
    long timestamp;
    struct Timestamp* next;
} Timestamp;

void parse(FILE* f, long* errors, long window);
```

Вывод:

В результате выполнения лабораторной работы удалось реализовать программу для анализа логов NASA с выводом списка ошибочных запросов и нахождением временного окна, когда количество запросов на сервер было максимальным.