```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <time.h>
#include <string.h>
#include <fcntl.h>
#define BUFFSIZE 4096

// https://stackoverflow.com/a/481691/9295513
// https://www.geeksforgeeks.org/strtok-strtok_r-functions-c-examples/
// https://www.tutorialspoint.com/c_standard_library/c_function_fread.htm
// https://www.techonthenet.com/c_language/standard_library_functions/stdio_h/remove.ph
p
// http://www.mathcs.emory.edu/~cheung/Courses/561/Syllabus/3-C/text-files.html
// https://stackoverflow.com/questions/10326586/segmentation-fault-with-strcpy

int main(int argc, char **argv) {
    if (argc < 2) {printf("No.\n");exit(-1);}

    char buf[BUFFSIZE];
    FILE* temp = fopen(argv[1], "r");
    FILE* output_log;
    output_log = fopen("output.log", "a");
    time_t starts;
    time_t ends;
    pid_t pid;
    int status;
    while (fgets(buf, BUFFSIZE, temp) != NULL) {
        char* tempo = (char *) malloc(strlen(buf) + 1);
        char* constantine = (char *) malloc(strlen(buf) + 1);
        printf("%s", buf);
        strcpy(tempo, buf);
        strcpy(constantine, buf);
        char* head = strtok(tempo, " \n");
                                                    int i = 1;
                                                    char* argu_ve[strlen(buf)];
                                                    argu_ve[0] = head;
                                                    char* next;
                                                    while ((next = strtok(NULL, " \
n")) != NULL) {
                                                        argu_ve[i] = next;
                                                        i++;
                                                    }
                                                    argu_ve[i] = NULL;
        time(&starts);
        pid = fork();
        if (pid == 0) {
            execvp(head, argu_ve);
            perror("execvp");
            exit(-1);

        } else if (pid > 0) {
            wait(&status);
            time(&ends);

            fprintf(output_log, "%s\t%s\t%s", buf, ctime(&starts), ctime(&ends));

            if (WIFEXITED(status)) {
                printf("Child process exited with status = %d\n", WEXITSTATUS(status));
            } else {
                printf("Child process exited abnormally. Failure.\n");
```

```
                exit(-1);
            }
        } else {
            perror("fork");
            exit(EXIT_FAILURE);
        }
    }
    fclose(temp);
    remove("temp");
    fclose(output_log);
    return 0;
}
```