UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

IMD1012 – INTRODUÇÃO ÀS TÉCNICAS DE PROGRAMAÇÃO

ARQUIVOS E MODULARIZAÇÃO – UNIDADE 2 – 2021.2

Daniel Luan Lourenço de Lima

RESOLUÇÕES

1. Os modos de leitura de arquivos em C utilizam algumas funções, que são elas:

- **fgetc**: Apenas recebe o indicador de fluxo do arquivo e retorna um caractere do arquivo, ou *EOF*, ou mesmo um erro.
- **fgets**: Nessa função, diferentemente da *fgetc*, os parâmetros são um ponteiro apontando para uma *string* onde os dados lidos serão guardados, o tamanho máximo dos dados a serem lidos, e o indicador de fluxo do arquivo. A leitura só é interrompida quando o caractere de *nova linha* (\r\n).
- **fscanf**: Tem o funcionamento semelhante ao *scanf*, porém a diferença é que a entrada dos dados não é pelo *stdin*, e sim por um arquivo. Nessa função, os parâmetros são o indicador de fluxo de um arquivo e a *string* onde os dados serão armazenados.

2. Nessa questão, para a compilação usar os comandos via terminal:

```
~gcc -o q2 q2.c
~./q2
```

O arquivo encontrava-se no mesmo diretório do executável, porém poderia estar em qualquer diretório da máquina.

Código-fonte:

```c
#include <stdio.h>
#include <stdlib.h>

int main(void){
    char read[2];
    FILE *img;
    img = fopen("xp.jpg", "r");

    if(img == NULL){
        printf("Error 404. Archive Not Found.\n");
    }
    else{
        fgets(read, 2*sizeof(char), img);
        printf("%s", read);
        fclose(img);
    }

    return 0;
}
```

3. Comandos para compilação:

~gcc -o q3 q3.c

~./q3

O código armazena dois diretórios através de duas *strings*, onde a entrada de ambas é feita através do *stdin*. Assim, elas são usadas para abertura de dois indicadores de fluxos de arquivos para realizar a cópia.

Portanto, ao iniciar o programa, é necessário inserir os dois diretórios, de acordo com o comando da questão.

Código-fonte:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void){
    char read[255];
    char drt_arq1[255];
    char drt_to_cpy[255];
    FILE *arq_to_cpy;
    FILE *arq_cpy;

    fgets(drt_arq1, 255, stdin);
    drt_arq1[strlen(drt_arq1)-1] = '\0';
    fgets(drt_to_cpy, 255, stdin);
    drt_to_cpy[strlen(drt_to_cpy)-1] = '\0';

    arq_to_cpy = fopen(drt_arq1, "r");
    arq_cpy = fopen(drt_to_cpy, "w");

    if(arq_cpy == NULL || arq_to_cpy == NULL){
        printf("Error 404. Archive Not Found.\n");
    } else {

        while(fgets(read, 255, arq_to_cpy) != NULL){
            fputs(read, arq_cpy);
        }
        fclose(arq_to_cpy);
        fclose(arq_cpy);

        printf("\nArquivo copiado!\n");
    }
    return 0;
}
```

4. Comandos para compilação:

~gcc -o q4 q4.c

~./q4

Utilizando a biblioteca *time.h*, salvam-se dois “tempos”: o primeiro, assim que o programa entra em execução, e o segundo, após o usuário escrever corretamente a palavra “SAIR”.

Código-fonte:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

int main(void){
    char end[255];
    struct tm *start;
    time_t start_t;
    time(&start_t);
    start = localtime(&start_t);

    FILE *log;
    log = fopen("src/log.txt", "w");
    if(log == NULL){
        printf("Error 404. Archive Not Found.\n");
    } else {
        fprintf(log, "[%d-%d-%d %d:%d:%d] Inicio do programa\n", start-
>tm_mday, start->tm_mon+1, start->tm_year+1900, start->tm_hour, start-
>tm_min, start->tm_sec);
        do{
           fgets(end, 255, stdin);
            end[strlen(end)-1] = '\0';
        } while(strcmp(end, "SAIR") != 0);
        struct tm *end;
        time_t end_time;
        time(&end_time);
        end = localtime(&end_time);

        fprintf(log, "[%d-%d-%d %d:%d:%d] Final do programa\n", end-
>tm_mday, end->tm_mon+1, end->tm_year+1900, end->tm_hour, end->tm_min,
end->tm_sec);

        fclose(log);
    }
    return 0;
}
```

5. Para a questão 5, foram criados 3 arquivos diferentes:
    a. **main.c**: Onde se encontra o código principal, o qual será compilado.
    b. **findValues.h**: Cabeçalho da nossa biblioteca, indicando quais serão as funções, parâmetros etc.
    c. **findValues.c**: Arquivo contendo as funções de fato, com todo o algoritmo.

Para compilar e executar o programa, foi utilizado o seguinte comando:

```
~gcc findValues.c main.c -o main -g -W
~./main
```

A seguir, os códigos-fontes:

Código-fonte main.c:

```c
#include <stdio.h>
#include "findValues.h"

int main(void){
    int n;

    scanf("%d", &n);

    int vet[n];

    for(int i = 0; i < n; i++)
        scanf("%d", &vet[i]);

    int minorVar = minor(n, vet);
    int majorVar = major(n, vet);

    printf("O menor valor inserido foi: %d\n", minorVar);

    printf("O maior valor inserido foi: %d\n", majorVar);

    return 0;
}
```

Código-fonte findValues.h:

```
#ifndef findValues
#define findValues

#define version 1.0.0;

int major(int n, int vet[]);

int minor(int n, int vet[]);

#endif
```

Código-fonte findValues.c:

```
#include "findValues.h"

int minor(int n, int vet[n]){
    int minor = 0;
    minor = vet[0];
    for(int i = 0; i < n; i++){
        if(vet[i] <= minor){
            minor = vet[i];
        }
    }
    return minor;
}

int major(int n, int vet[n]){
    int major;
    major = vet[0];
    for(int i = 0; i < n; i++){
        if(vet[i] >= major){
            major = vet[i];
        }
    }
    return major;
}
```