

UFRN - Universidade Federal do Rio Grande do Norte
IMD - Instituto Metr pole Digital

INTRODU  O A T CNICAS DE PROGRAMA  O

Atividade extra de Arquivos e Modulariza  o

Stefane de Assis Orichuela - 20210056974

Quest o 01:

Listagem abaixo dos modos de leitura de arquivos para linguagem C:

1. **fscanf:** Essa fun  o retorna a leitura da quantidade de elementos encontrados em determinado arquivo.
2. **fgetc:** A fun  o recebe um  nico par metro, sendo arquivo, e l  um caractere do mesmo.
3. **fgets:** Essa fun  o recebe tr s par metros. Onde, seu objetivo principal   armazenar em uma vari vel a leitura das strings colhidas de determinado arquivo. Nessa fun  o tamb m   poss vel limitar a quantidade de caracteres que a fun  o ter  que ler e armazen -la.

As diferen as entre as fun  es s o claras, o scanf sendo o mais distante, ir  contar os elementos do arquivo, j  o fgetc ler  um caractere do arquivo, e por fim, o fgets sendo um primo do fgetc, ler  at  determinada quantidade de strings de um arquivo.

Questão 02:

<https://github.com/StefWolf/ITP/tree/main/Lista%20arquivos/Quest2>

Para a questão 02, pensei em utilizar como “teste” uma imagem de um cachorrinho que estará disponível junto com o código no link do repositório que criei logo acima.

Primeiramente devo apresentar meu código-fonte:

```
1  #include <stdio.h>
2
3  int main(void) {
4      char caracter[2];
5      int cont = 0;
6      FILE *dog;
7      dog = fopen("cachorrinho.jpg", "r");
8      if(dog == NULL){
9          printf("Error ao abrir arquivo");
10     } else{
11
12         while(fgets(caracter, 2, dog) != NULL){
13             cont++;
14             printf("%s", caracter);
15             if(cont == 16){ //Sabendo que 2 bytes equivale a 16 bits
16                 break;
17             }
18         }
19     }
20 }
21 fclose(dog);
22
23 /* gcc -o main main.c
24    ./main
25 */
26
27 return 0;
28 }
```

Variável dog terá endereço do arquivo.

Verificando se houve erros ao abrir

Repita enquanto a leitura do arquivo não chegue até o limite permitido na função

A cada loop ele irá disponibilizar na tela o caractere em bits que foi encontrado até que o contador chegue a 16 bits

Logo em seguida na própria imagem deixei alguns comentários tanto no código quanto na imagem para poder explicar exatamente o que significa os processos mais “complexos” dele.

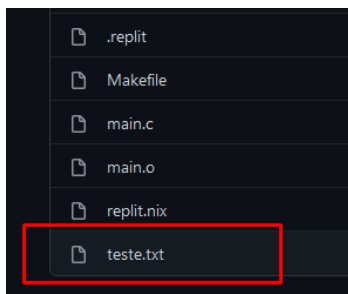
Compilando no Replit, cheguei a um possível resultado:

```
➤ make -s
➤ ./main String dos dois
  JFIF primeiros bytes
```

Questão 03:

<https://github.com/StefWolf/ITP/tree/main/Lista%20arquivos/Quest3>

Na questão dois utilizei no meu repositório além do arquivo principal main.c, um arquivo txt para ser copiado, chamado teste.txt:



```
1  #include <stdio.h>
2
3  void copiarArquivo(FILE *arq1, FILE *arq2){
4      char ler[1000]; /* Aqui coloquei um limite de até 1000 caracteres */
5      while(fgets(ler, 1000, arq1) != NULL){ // enquanto estiver processando os caracteres...
6          fputs(ler, arq2); //passe o que tiver no vetor ler para o arquivo 2 de caractere por caractere
7      }
8      printf("Arquivo copiado!");
9  }
10
11 int main(void) {
12     char a1[100];
13     char a2[100];
14
15     scanf("%s", a1);
16     scanf("%s", a2);
17
18     //a2 = a2 + "cópia";
19     FILE *arquivo1 = fopen(a1, "r");
20     if(arquivo1 == NULL){
21         printf("Não abriu");
22     }
23     FILE *arquivo2 = fopen(a2, "w");
24     if(arquivo2 == NULL){
25         printf("Não foi possível encontrar o caminho do arquivo");
26     }
27     copiarArquivo(arquivo1, arquivo2);
28     fclose(arquivo1);
29     fclose(arquivo2);
30
31     /* gcc -o main main.c
32     ./main
33     */
34     return 0;
35 }
```

Função para fazer a copia do arquivo

Strings de tamanho 100, como exemplo definido no meu código.

Abre o arquivo que será copiado para leitura.

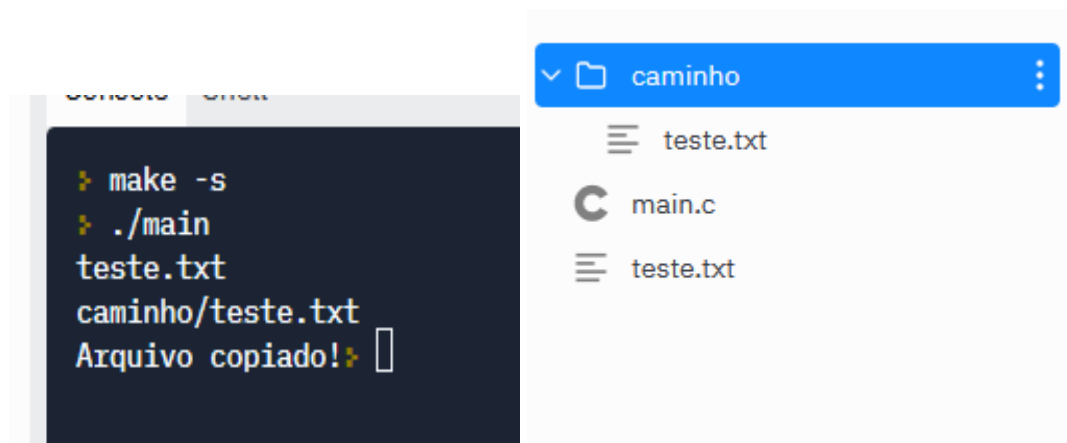
Abre o arquivo que será a cópia para escrita, caso não exista ele cria uma.

arquivo1 será copiado para arquivo2

No código apresentado logo acima, pode-se notar que utilizei uma função para copiar que recebe dois parâmetros: O caminho do arquivo existente e o caminho que quer que o arquivo seja copiado.

As variáveis a1 e a2 irão receber esse endereço através das linhas 15 e 16. Onde o usuário irá atribuir as variáveis o caminho do arquivo e para onde será feito a cópia. Após isso, abri o primeiro arquivo no ponteiro *arquivo1, somente para leitura. Já o próximo foi aberto no *arquivo2 para escrita e também criação caso esse não exista.

Após terminado, segue uma print de um teste de execução feita no replit:



The image shows two side-by-side screenshots from a Replit environment. The left screenshot is a terminal window with a dark background, showing the following commands and output: `make -s`, `./main`, `teste.txt`, `caminho/teste.txt`, and `Arquivo copiado!`. The right screenshot is a file explorer window with a light blue header bar labeled 'caminho'. It displays a directory structure with two files: 'teste.txt' and 'main.c'.

Questão 04:

<https://github.com/StefWolf/ITP/tree/main/Lista%20arquivos/Quest4>

Nessa questão utilizei somente o arquivo main.c e um arquivo chamado log.txt para armazenar constantemente os momentos que o programa é executado e parado.

```
1 #include <stdio.h>
2 #include <time.h>
3 #include <string.h>
4
5
6 int main(void) {
7     char sair[4];
8     struct tm *data_inicio; /*Hora atual não-brasil*/
9     FILE* arquivo;
10
11     time_t segundos;
12     time_t fim;
13
14     time(&segundos);
15
16     data_inicio = localtime(&segundos);
17     arquivo = fopen("log.txt", "a");
18
19     fprintf(arquivo, "Data: %d/%d/%d\n", data_inicio->tm_mday, data_inicio->tm_mon+1, data_inicio->tm_year+1900);
20     fprintf(arquivo, "Tempo inicial: %d:%d:%d\n", data_inicio->tm_hour, data_inicio->tm_min, data_inicio->tm_sec);
21
22     do{
23         scanf("%s", sair);
24     } while (strcmp(sair, "SAIR") != 0);
25
26     struct tm *data_fim;
27     time(&fim);
28     data_fim = localtime(&fim);
29
30     fprintf(arquivo, "Tempo final: %d:%d:%d\n", data_fim->tm_hour, data_fim->tm_min, data_fim->tm_sec);
31
32     fclose(arquivo);
33
34     return 0;
35 }
36
```

Incluindo a biblioteca time.h para manipulação de data e hora

Criação de ponteiro em formato lista e tipo tm que irá recolher informações de data e hora da biblioteca time.h

Variáveis do tipo time que dará o tempo início (segundos) e o tempo fim (fim) no arquivo log.

Convertendo a variável do tempo inicial para time

Recolhendo informações do momento presente da data e do tempo

Abrindo arquivo log para escrita contínua.

Escrevendo logo no log a data que abriu e o momento que também foi executado.

Enquanto não digitar SAIR, ele não passará para a próxima etapa.

Apresentando o tempo de agora após sair do loop anterior

Registrar no log

Como mostrado, chamei a biblioteca time.h para fazer a integração do tempo e da data. Após isso, criei uma listagem de ponteiros do tipo tm (da biblioteca) para poder chamar as diferentes propriedades para o meu código: dia, mês, ano, hora, minuto, segundo.

Por fim, ao executar no replit consegui o seguinte resultado:

```
log.txt x
1 Data: 27/1/2022
2 Tempo inicial: 21:4:6
3 Tempo final: 21:4:10
4 Data: 27/1/2022
5 Tempo inicial: 21:7:16
6 Tempo final: 21:7:18
7 Data: 27/1/2022
8 Tempo inicial: 23:38:20
9 Tempo final: 23:38:23
10
11
```

Deixando claro que o tempo nesse exemplo está sendo processado no horário GMT.

Questão 5:

<https://github.com/StefWolf/ITP/tree/main/Lista%20arquivos/Quest5>

Para a questão 5 irei dividi-la em três partes:

1. rastreador.h

Aqui temos o cabeçalho, que será utilizado para criar a “ponte” e gerar uma biblioteca com as funcionalidades do arquivo rastreador.c para o arquivo principal main.c

```
1  #ifndef RASTREADOR_H
2  #define RASTREADOR_H
3
4  #define versao 1.0
5
6  int encontrarMenor(int vet[], int i, int cont);
7  int encontrarMaior(int vet[], int i, int cont);
8
9  #endif
10
```

Variável constante indicando a versão

Funções de encontrar o maior e o menor valor do vetor.

2. rastreador.c

Aqui se encontram as funções que serão incluídas na biblioteca rastreador.h. Como pedido: Função de encontrar o menor e maior valor de um vetor.

```
1  #include "rastreador.h"
2
3  int encontrarMenor(int vet[], int i, int cont){
4      if(i < 0){
5          return cont;
6      } else{
7          if(vet[i] < cont){ //Verifica se o valor atual é menor que o menor indicado até agora
8              cont = vet[i];
9          }
10         return encontrarMenor(vet, i-1, cont);
11     }
12 }
13
14 int encontrarMaior(int vet[], int i, int cont){
15     if(i < 0){
16         return cont;
17     } else{
18         if(vet[i] > cont){ //Verifica se o valor atual é maior que o maior indicado até agora
19             cont = vet[i];
20         }
21         return encontrarMaior(vet, i-1, cont);
22     }
23 }
24 }
```

Chamando o cabeçalho

Oh! note que eu fiz elas de modo recursivo com os valores dos vetores sendo carregados de forma decrescente. Como parâmetros, a função recebe: o vetor, um contador limite do tamanho do vetor que

poderá ser modificado na recursão e uma variável contador que indica o tamanho fixo do vetor. (vetor, i, contador)

3. main.c

Para concluir, criei o algoritmo base junto com uma pequena interface básica para orientar o usuário do que tem que fazer:

```
1  #include <stdio.h>
2  #include "rastreador.h"
3
4  int main(void) {
5
6      int contador;
7      printf("Rastreador de dados versão %2.f\n", versao);
8      printf("Quantos valores está o vetor? ");
9      scanf("%d", &contador);
10     int vetor[contador];
11
12     printf("\n----- Coloque os valores no vetor -----\n");
13     for(int i = 0; i < contador; i++){
14         scanf("%d", &vetor[i]);
15     }
16
17     printf("Menor valor do vetor: %d\n", encontrarMenor(vetor, contador - 1, vetor[contador-1]));
18     printf("Maior valor do vetor: %d\n", encontrarMaior(vetor, contador - 1, vetor[contador-1]));
19     //vet, i-1, cont
20     return 0;
21 }
```

Chamando biblioteca criada

Criando e adicionando valores ao vetor

Saída do resultado

Segue o resultado final do programa:

```
Rastreador de dados versão 1
Quantos valores está o vetor? 10

----- Coloque os valores no vetor -----
7
5
6
23
9
0
5
1
6
8
Menor valor do vetor: 0
Maior valor do vetor: 23
>
```