**FACULDADE METROPOLITANA DE MANAUS – FAMETRO**

**Pesquisar funcionalidade**

**MANAUS – AM**

**2020**

**DISCENTES:** Marcelo dos Anjos Caldas - 1900194**.**

**Pesquisar definição e sintaxe:** fopen, fclose, fwrite, fread.

Trabalho acadêmico solicitado pelo docente Bruno, professor da disciplina de Estrutura de Dados II, do 4º Período do curso de Bacharelado em Sistemas de Informação, da instituição FAMETRO, em prol de obtenção de nota parcial para o semestre letivo.

**MANAUS – AM**

**2020**

1. **DEFINIÇÕES:**

**fopen**. Abre um arquivo texto para leitura. O arquivo deve existir antes de ser aberto.

**fclose.** Quando acabamos de usar um arquivo que abrimos, devemos fecha-lo.

**fwrite.** tenta escrever para o fluxo numero\_itens elementos, com tamanho bytes cada. Em caso de sucesso, ou seja, todos os elementos tenham sido escritos com sucesso, fwrite escreveu (tamanho \* numero\_itens) bytes do parâmetro dados para o fluxo. fwrite funciona como se [fputc](http://www.cmaismais.com.br/referencia/cstdio/fputc) fosse chamada tamanho vezes para cada objeto. Note que fwrite apenas funciona como chamadas sucessivas a fputc. Na realidade, fwrite não faz uso da função fputc.

O ponteiro interno de posição do fluxo é avançado pelo número de bytes escritos com sucesso. A marca temporal de última modificação do arquivo é atualizada.

Parâmetros:

dados - Ponteiro para onde os dados a serem armazenados. Deve possuir um tamanho, em bytes, de ao menos (tamanho \* numero\_itens)

tamanho - Tamanho de um único elemento

numero\_itens - Número de elementos, cada um com um tamanho de tamanho bytes

fluxo - Fluxo a ser escrito

**fread**. tenta ler do fluxo numero\_itens elementos, com tamanho bytes cada. Em caso de sucesso, ou seja, todos os elementos tenham sido lidos com sucesso, fread lê (tamanho \* numero\_itens) bytes do fluxo para o parâmetro dados. fread funciona como se [fgetc](http://www.cmaismais.com.br/referencia/cstdio/fgetc) fosse chamada tamanho vezes para cada objeto. Note que fread apenas funciona como chamadas sucessivas a fgetc. Na realidade, fread não faz uso da função fgetc.

O ponteiro interno de posição do fluxo é avançado pelo número de bytes lidos.

Parâmetros:

dados - Ponteiro para onde os dados serão armazenados. Deve possuir um tamanho, em bytes, de ao menos (tamanho \* numero\_itens)

tamanho - Tamanho de um único elemento

numero\_itens - Número de elementos

fluxo - Fluxo a ser lido

**feof** verifica o indicador de fim de arquivo (EOF) do fluxo e retorna um valor diferente de zero se este indicador estiver setado.

O indicador de fim de arquivo (EOF) de um fluxo pode ser limpo utilizando a função [clearerr](http://www.cmaismais.com.br/referencia/cstdio/clearerr).

int feof (FILE \* fluxo);

fluxo - Fluxo a ter o seu indicador de EOF testado.

Retorna um valor diferente de zero se e apenas se o indicador de fim de arquivo (EOF) do fluxo estiver setado. Se o indicador de EOF estiver limpo, retorna zero.

1. **SINTAXE:**

FILE \*fopen (char \*nome\_do\_arquivo,char \*modo);

fclose (FILE \*fp);

size\_t fwrite(void \* dados, size\_t tamanho, size\_t numero\_itens, FILE \* fluxo);

size\_t fread(void \* dados, size\_t tamanho, size\_t numero\_itens, FILE \* fluxo);

int feof (FILE \* fluxo);

**Ex01:**

1. #include <stdio.h
2. main (void)
3. {
4. FILE \*fp;
5. ...
6. fp=fopen ("exemplo.bin","wb");
7. if (!fp)
8. {
9. printf ("Erro na abertura do arquivo. Fim de programa.");
10. exit (1);
11. }
12. ...
13. return 0;
14. }

**Ex02:**

#include <stdio.h>

int main(){

/\* Declaramos as variáveis que serão utilizadas \*/

char arquivo[100];

char texto[100];

/\* Declaramos um ponteiro(link para o endereço da memória) para o arquivo de nome: 'pFile'\*/

FILE \* pFile;

/\* Escrevemos na tela a pergunta para o arquivo e/ou caminho do arquivo \*/

printf("Informe o arquivo(COM CAMINHO) que você deseja escrever: \n");

/\* Gravamos o que o usuário digitar (caminho/arquivo)\*/

scanf("%s", &arquivo);

/\* Esta é a função de abertura de arquivos. Com modo 'w'(Abrir um arquivo texto para gravação. Se o arquivo não existir, ele será criado.

Se já existir, o conteúdo anterior será destruído.) \*/

pFile = fopen(arquivo , "w");

/\* Se o arquivo for não-vazio \*/

if(pFile!=*NULL*){

/\* Escrevemos na tela a pergunta para o que deseja ler no arquivo (ESPAÇOS EM BRANCO FINALIZARÁ o texto) \*/

printf("Digite o que você deseja pôr no arquivo(SEM ESPAÇOS): \n");

/\* Gravamos o que o usuário digitar e guardamos na variável 'texto' \*/

scanf("%s", &texto);

/\*Escreve uma string(da variável 'texto') num arquivo. \*/

fputs(texto, pFile);

/\* Quando acabamos de usar um arquivo que abrimos, devemos fechá-lo. Para tanto usa-se a função fclose() \*/

fclose(pFile);

}

return 0;

}

**Ex03:**

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE\* arquivo = fopen("arquivo\_binario.bin", "wb");

if(arquivo == NULL) {

fprintf(stderr, "Erro ao tentar abrir arquivo\_binario.bin.\n");

return 1;

}

/\* Prepara os dados \*/

char dados[] = "http://www.cmaismais.com.br - A melhor referência em português de C.\n";

int tamanho = sizeof(dados);

/\* Escreve os dados \*/

size\_t elementos\_escritos = fwrite(dados, 1, tamanho, arquivo);

if(elementos\_escritos != tamanho) {

fprintf(stderr, "Erro ao tentar escrever no arquivo.\n");

return 2;

}

/\* Fecha o arquivo \*/

fclose(arquivo);

return 0;

}

**Ex04:**

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE\* arquivo = fopen("arquivo\_binario.bin", "rb");

if(arquivo == NULL) {

fprintf(stderr, "Erro ao tentar abrir arquivo\_binario.bin.\n");

return 1;

}

/\* Lê o tamanho total do arquivo \*/

fseek(arquivo, 0, SEEK\_END);

long tamanho = ftell(arquivo);

rewind(arquivo);

/\* Armazena a posição inicial \*/

char\* buffer = (char \*) malloc(tamanho \* sizeof(char));

if(buffer == NULL) {

fprintf(stderr, "Erro ao tentar alocar o buffer de memória.\n");

return 2;

}

/\* Lê o arquivo inteiro em uma só chamada \*/

size\_t elementos\_lidos = fread(buffer, 1, tamanho, arquivo);

if(elementos\_lidos != tamanho) {

fprintf(stderr, "Erro ao tentar ler o arquivo inteiro.\n");

return 3;

}

/\* Desaloca os recursos \*/

free(buffer);

fclose(arquivo);

return 0;

}

**Ex05:**

#include <stdio.h>

int main() {

FILE \*arquivo;

char c;

arquivo = fopen("arquivo.txt", "w");

if(arquivo == NULL) {

fprintf(stdderr, "Erro ao abrir o arquivo.txt\n");

return 1;

}

/\* lê até receber um EOF. receber um EOF não significa que o fgetc

leu até alcançar o fim do arquivo. erros também retornam EOF. \*/

while(fgetc(arquivo != EOF);

if(feof(arquivo)) {

puts("O arquivo inteiro foi lido.");

} else {

puts("O indicador de fim de arquivo (EOF) não está marcado. Um erro ocorreu.");

}

fclose(arquivo);

return 0;

}