Quais as funções necessárias para criar, ler, gravar, apagar, renomear um arquivo?, como elas funcionam?(parâmetros, retornos).

Arquivos na programação servem para manter, salvar, os dados e resultados que antes eram perdidos ao fechar o programa, a manipulação dos arquivos permite salvá-los em forma de bytes em sequencia se tornando um conjunto de dados (stream) permitindo acessar essas informações a qualquer momento.

Dentre as manipulações de arquivo de texto e binário através de um programa em linguagem C, podemos dar algumas ações a esses arquivos dentre elas: criar, ler, gravar, renomear ou apagar um arquivo, para dar início de um arquivo texto(.txt) ou binário(.dat) deve ser feito através do tipo de dado FILE e o nome de um ponteiro a sua escolha, a sintaxe é igual a de criação de um ponteiro: FILE \*ptr; o tipo de dado deve ser em caixa alta pois é uma palavra reservada da programação C onde não é necessário uma biblioteca específica para sua leitura,após a criação para abrir o arquivo e iniciar uma operação deve se seguir a seguinte sintaxe: file=fopen("D:\\Usuários\\Desktop\\teste.txt", "r")  
file sendo o nome do ponteiro, fopen (retorna nulo caso não possa ser aberto) que faz com meu ponteiro aponte um arquivo no meu programa, o primeiro argumento deve ser o diretório do arquivo, ou caso queria criá-lo onde no direito de onde salvou o programa, escreva somente o nome do arquivo que queira criar, caso queria em um diretório diferente, deve-se utilizar duas contra barras pois apenas uma contra barra é resevado para comandos como \n \t, já o segundo argumento informa o modo de abertura, read (r) permite leitura e devolve nulo caso o arquivo não exista, write(w) permite escrita e cria o arquivo caso não exista, append(a) permite acrescentar a um já existente ou cria um novo caso não exista, essas funções são as básicas para arquivos .txt, porém temos outros modos como: (r+) (w+) (a+) as três formar permite leitura, escrita e caso o arquivo não exista, cria, mas cada uma pra uma função, r+ altera dados, w+ recria, a+ acrescenta dados, para criação e manipulação de arquivos binários, não é muito diferente dos arquivos de texto, apenas adicionar um b(binário) após as letras r, w ou a, ficando rb, wb, ab, r+b, w+b, a+b.

Após criar o arquivo e escolher o que quer manipular, deve se fazer a verificação caso necessário se o arquivo existe utilizando o argumento NULL (nulo), como exemplo:  
if(arquivo == NULL)  
 printf("Arquivo inválido")  
Após a verificação e todos os procedimentos que se deseja fazer, deve-se “salvar” o arquivo indicando para o programa aonde terminou, para assim indicar seu EOF (Enf of File), para fechar arquivos deve se usar fclose(); ou fcloseall();. Para apenas um arquivo ultize fclose(nome do ponteiro):  
fclose(file)  
Para vários arquivos utilize:  
fcloseall()

Dentro da parte de processamento do programa podemos utilizar alguma funções como:  
fgetc: que funciona por parâmetro devolvendo a função principal, que foi aberta por fopen, caracteres contidos entre 0 e 255 que é a variação da tabela ASCII, caso não seja encontrado é declarado EOF(End-of-File). Sintaxe: int fgetc (FILE \*file)  
fputc: essa função escreve caracteres devolvendo caso caracter ch em caso de sucesso e EOF caso contrário. Sintaxe: int fputc (int ch, FILE \*arq)  
fscanf: lê dados de forma formatada e funciona da mesma forma que scanf mais com um parâmetro inicial que é o arquivo onde será realizado o processamento. Sintaxe: int fscanf (FILE \*arq, const char \*file, ...)  
fprintf: escreve dados formatados e funciona como printf mas também tem o diferencial do parâmetro inicial que é o arquivo onde será realizado. Sintaxe: int (FILE \*arq, const char \*format, ...)  
fscanf e fprintf são apenas usados para arquivos txt.

**Modos de abertura: read (r), write (w) e append(a)**

**read(r) - Leitura de arquivo**

***r***

Sempre que quisermos ler um arquivo, usamos o modo *read*, que será representado simplesmente pela letra "r":

FILE \*arquivo = fopen("arquivo.txt", "r");

Isso faz o arquivo ser aberto unicamente para ser lido (não podemos alterar nada).

***r+***

Se adicionarmos um sinal de "+" no "r", será possível abrir o arquivo tanto para leitura como para escrita, e caso ele não exista, será automaticamente criado.

E caso exista, o seu conteúdo anterior será apagado e substituído pelo novo:

FILE \*arquivo = fopen("arquivo.txt", "r+")

***rb***

Abre o arquivo para leitura, mas em modo binário:

FILE \*arquivo = fopen("arquivo.txt", "rb")

**write(w) - Escrita em arquivo**

***w***

Para abrirmos um arquivo para escrita (colocar informações do nosso programa no arquivo), usamos a letra "w". Esse modo automaticamente cria o arquivo, ou substitui seu conteúdo anterior:

FILE \*arquivo = fopen("arquivo.txt", "W")

***w+***

Abertura do arquivo tanto para leitura como para escrita.

Caso já exista o arquivo, seu conteúdo será substituído:

FILE \*arquivo = fopen("arquivo.txt", "w+")

***wb***

Escrita em arquivos no modo binário:

FILE \*arquivo = fopen("arquivo.txt", "wb")

**append(a) - Escrevendo ao final do arquivo (anexando)**

***a***

Ao usar o "w" para escrever (*write*), começamos a inserir informação no início do arquivo.

Mas, e se quisermos inserir ao final? Se não quisermos apagar o que tinha lá antes?

Para isso vamos ANEXAR, informações! Anexar em inglês é *append*, e fazemos isso abrindo o arquivo usando o modo de abertura "a":

FILE \*arquivo = fopen("arquivo.txt", "a")

***a+***

Caso queiramos abrir um arquivo para leitura ou para escrita ao FINAL do arquivo (anexar), usamos o símbolo "+" depois da letra "a":

FILE \*arquivo = fopen("arquivo.txt", "a+")

***ab***

Por afim, assim como na leitura "rb" e escrita binária "wb", podemos anexar informações ao final do arquivo, de maneira binária usando o "ab":

FILE \*arquivo = fopen("arquivo.txt", "ab")

**EOF (*end of file)*, final do arquivo  - Fechando arquivos com *fclose()* e *fcloseall()***

Para saber o final do arquivo (que para o sistema, é uma sequência de bytes), o C vai procurar um sinal, uma constante conhecida por ***EOF***, que sinaliza o final do arquivo.

Para identificar o final de um arquivo, podemos usar a função *fclose*, que recebe um ponteiro para o tipo***FILE*** e retorna um inteiro:

int fclose(FILE \*arq)

Caso o byte lido represente o ***EOF***, a função "fecha" a abertura do arquivo. Ou seja, libera a memória associado ao ponteiro do ***FILE\****.

Assim como ensinamos em ponteiros, ao usar a [função free() para liberar memória alocada](http://www.cprogressivo.net/2013/10/Funcao-free-Como-liberar-memoria-e-evitar-vazamento.html), fechar os arquivos que você não está mais usando é uma excelente prática de programação.

E caso trabalhe com diversos arquivos, você poderá fechá-los todos de uma vez, através da função:

int fcloseall()

**Erros em abertura de arquivos**

Quando abrimos um arquivo, o ponteiro que criamos do tipo ***FILE*** armazenará o endereço de um arquivo.

Porém, nem sempre esta tarefa é possível, gerando um erro.

Quando este erro ocorre o ponteiro irá apontar para NULL, sendo essa prática (checagem se o ponteiro aponta para NULL) muito importante para o tratamento de erros na abertura de [**arquivos em C**](http://www.cprogressivo.net/p/arquivos-em-c.html).

Este erro pode ocorrer por vários motivos. O mais óbvio é abrir um arquivo que não existe (geralmente ocorre quando escrevemos errado o endereço do arquivo).

Em sistemas operacionais mais seguros, como os Linux, o acesso aos arquivos (seja para leitura, escrita ou anexação) é geralmente limitado, não sendo possível acessar e alterar qualquer arquivo (como geralmente ocorre em sistemas mais vulneráveis, como o Windows).

Caso você tente alterar ou acessar um arquivo sem permissão, a função *fopen* também irá retornar NULL para o ponteiro.

if(arquivo == NULL)

   printf("Nao foi possivel abrir o arquivo!");