

**UNIVERSIDADE DO ESTADO DE MINAS GERAIS – CAMPUS DE FRUTAL**

**CURSO DE SISTEMAS DE INFORMAÇÃO**

**SEMINÁRIO: ARQUIVOS EM C, TEXTO E BINÁRIO.**

**IVANILSON**

**LEONARDO**

**LEOPOLDO**

**MAXIMILIANO**

**Frutal (MG)**

**2014**

****

**Ivanilson**

**Leonardo**

**Leopoldo**

**Maximiliano**

**SEMINÁRIO: ARQUIVOS EM C, TEXTO E BINÁRIO.**

Trabalho de Algoritmos e programação apresentado como exigência parcial para obtenção de **Nota para a segunda etapa do segundo período do Curso de Sistemas de Informação** à Universidade do Estado de Minas Gerais – UEMG – Campus de Frutal, sob a orientação do Prof. Ms. Sérgio Portari.

**Frutal (MG)**

**2014**

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.

Dentro da parte de processamento do programa podemos utilizar alguma funções como:  
remove: função disponível na biblioteca stdio.h que apaga um arquivo.  
rename: função também da stdio.h que renomeia um arquivo.  
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.

No processamento de arquivos binário, as operações para processar os dados são chamados de Acesso Direto, e só são utilizados em arquivos em modo binário de leitura, as funções de leitura e escrita são:  
fwrite: faz parte da biblioteca stdio.h se diferenciando pois até agora as funções eram do padrão da linguagem C, a função serve para escrever um bloco de bytes. Sintaxe: int fwrite (const void \*ptr, int size, int n, FILE \*arq) prt é um ponteiro para void const siginifica que o parâmetro não será alterado, size é o tamanho em bytes de cada um dos elementros, n o numero de elementos e arq é onde os dados serão colocados assim a função dá o retorno de quantos itens se conseguiu escrever com sucesso.  
fread: também da biblioteca stdio.h faz a leitura de um bloco de bytes de um bloco binário. Sintaxe: int fread(const void \*ptr, int size, int n, FILE \*arq) igual ao fwrite porém, retorna o numero de itens que se conseguiu ler com sucesso.

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 (End 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()

End of file é dado quando a posição corrente do arquivo estiver além do número de bytes que ele contém, mesmo se abrirmos um arquivo vazio ele precisa verificar uma tentativa de processamento além do final do arquivo, pode não ter nem uma letra, mas precisa tentar achar essa letra pra realmente se dar o EOF, a detecção é feita através da função feof cuja sintaxe é: int feof (FILE \*arq), essa função devolve um valor e só detecta o EOF como dito anterior após uma verificação final que confirma o EOF.