Manipulação de Arquivos em C
Resumo de funções

fopen

Protótipo: FILE *fopen (char *nome_do_arquivo,char *modo);

- O primeiro parâmetro "nome_do_arquivo": é o local onde o arquivo se encontra ou se for criado, onde o arquivo deve ser armazenado.
- O segundo parâmetro "*modo": especifica como o arquivo deve ser aberto. Abaixo uma tabela mostrando os modos possíveis:

r	Abre um arquivo texto para leitura. O arquivo deve existir antes de ser aberto.
w	Abrir um arquivo texto para gravação. Se o arquivo não existir, ele será criado. Se já existir, o conteúdo anterior será destruído.
a	Abrir um arquivo texto para gravação. Os dados serão adicionados no fim do arquivo ("append"), se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
rb	Abre um arquivo binário para leitura. Igual ao modo "r" anterior, só que o arquivo é binário.
wb	Cria um arquivo binário para escrita, como no modo "w" anterior, só que o arquivo é binário.
ab	Acrescenta dados binários no fim do arquivo, como no modo "a" anterior, só que o arquivo é binário.
r+	Abre um arquivo texto para leitura e gravação. O arquivo deve existir e pode ser modificado.
w+	Cria um arquivo texto para leitura e gravação. Se o arquivo existir, o conteúdo anterior será destruído. Se não existir, será criado.
a+	Abre um arquivo texto para gravação e leitura. Os dados serão adicionados no fim do arquivo se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
r+b	Abre um arquivo binário para leitura e escrita. O mesmo que "r+" acima, só que o arquivo é binário.
w+b	Cria um arquivo binário para leitura e escrita. O mesmo que "w+" acima, só que o arquivo é binário.
a+b	Acrescenta dados ou cria uma arquivo binário para leitura e escrita. O mesmo que "a+" acima, só que o arquivo é binário.

fclose

Protótipo: int fclose (FILE *fp);

- O ponteiro fp passado à função fclose() determina o arquivo a ser fechado. A função retorna zero no caso de sucesso.

Testando o sucesso de abertura de um arquivos

fp == NULL?

```
if (!fp)   printf ("Erro na abertura do arquivo.");  else printf("Arquivo aberto com sucesso.");
```

putc

Escreve um caractere no arquivo.

Protótipo: int putc (int ch,FILE *fp);

- Parâmetro 1: "int ch" - caracter a ser escrito no arquivo (pode ser uma variável).
- Parâmetro 2: "FILE *fp" - o arquivo a ser escrito.

getc

Retorna um caractere lido do arquivo.

Protótipo: int getc (FILE *fp);

- Parâmetro: "FILE *fp" - o arquivo a ser lido.

feof

EOF ("End of file") - indica o fim de um arquivo. Às vezes é necessário verificar se um arquivo chegou ao fim, para isto podemos usar a função feof(). Ela retorna não-zero se o arquivo chegou ao EOF, caso contrário retorna zero.

Protótipo: int feof (FILE *fp);

Parâmetro: "FILE *fp" - o arquivo a ser verificado.

fprintf

Escreve uma constante de caracteres em um arquivo, essa função funciona como printf, sendo que a diferença é a saída padrão, que em fprintf é um arquivo e em printf é o monitor.

Protótipo: int fprintf (FILE *fp, const char *format,...);
- Parâmetro 1: "FILE *fp" - o arquivo a ser escrito.
- Parâmetro 2: "const char *format" - o que será escrito.

fscanf

Lê uma constante de caracteres de um arquivo, essa função funciona como scanf, sendo que a diferença é a entrada padrão que em fscanf é um arquivo e em scanf é o teclado.

Protótipo: int fscanf (FILE *fp, const char *format,...);
- Parâmetro 1: "FILE *fp" - o arquivo a ser lido.
- Parâmetro 2: "const char *format" - qual a variável que receberá os dados do arquivo.

fgets

Lê uma string de um arquivo. A função lê a string até que um caracter de nova linha seja lido /n ou tamanho-1 caracteres tenham sido lidos. Se o caracter de nova linha for lido, ele fará parte da string, o que não acontecia com gets. A string resultante sempre terminará com /0 (por isto somente tamanho-1 caracteres, no máximo, serão lidos).

Protótipo: char *fgets (char *str, int tamanho, FILE *fp);
- Parâmetro 1 "char *str": a variável que receberá a string.
- Parâmetro 2 "int tamanho": o limite máximo de caracteres a serem lidos.
- Parâmetro 3 "FILE *fp": o arquivo que será lido.

fputs

Escreve uma string num arquivo.

Protótipo: char *fputs (char *str, FILE *fp);
- Parâmetro 1: "char *str" - a variável que contém os dados a serem escritos no arquivo.
- Parâmetro 2: "FILE *fp" - o arquivo a ser escrito.

fread

Leitura de bloco de dados.

Protótipo: unsigned fread (void *buffer, int numero_de_bytes, int count, FILE *fp);
- Parâmetro 1: "void *buffer" - variável na qual serão armazenados os dados lidos.
- Parâmetro 2: "int numero_de_bytes" - o número de bytes a ser lido.
- Parâmetro 3: "int count" - indica quantas unidades devem ser lidas.
- Parâmetro 4: "FILE *fp" - arquivo a ser lido.

fwrite

Leitura de bloco de dados.

Protótipo: unsigned fwrite (void *buffer, int numero_de_bytes, int count, FILE *fp);
- Parâmetro 1: "void *buffer" - variável na qual serão transmitidos os dados ao arquivo.
- Parâmetro 2: "int numero_de_bytes" - o número de bytes a ser escrito.
- Parâmetro 3: "int count" - indica quantas unidades devem ser escritas.
- Parâmetro 4: "FILE *fp" - arquivo a ser escrito.

fseek

Procuras e acessos randômicos em arquivos. Esta move a posição corrente de leitura ou escrita no arquivo de um valor especificado, a partir de um ponto especificado.

Protótipo: int fseek (FILE *fp, long numbytes, int origem);
- Parâmetro 1: "FILE *fp" - arquivo a ser manipulado.
- Parâmetro 2: "long numbytes" - indica quantos bytes o cursor de posição do arquivo será movimentado apartir da sua posição atual.
- Parâmetro 3: "int origem" - indica apartir de onde os numbytes serão contados. Abaixo uma tabela com os possíveis valores:

Nome	Valor	Significado
SEEK_SET	0	Início do arquivo
SEEK_CUR	1	Ponto corrente no arquivo
SEEK_END	2	Fim do arquivo

rewind

Retorna a posição corrente do arquivo para o início.

Protótipo: void rewind (FILE *fp);
- Parâmetro: "FILE *fp" - arquivo a ser manipulado.