Escrevendo em arquivos - As funções fputc(), fprintf() e fputs()

Agora que já aprendemos as operações básicas com arquivos, como abrir e fechar, podemos finalmente trabalhar com eles em nossos projetos.

Neste tutorial de nossa **apostila de C**, iremos ensinar como escrever em arquivos através das funções*fputc*, *fprintf* e *fputs*.

- Deseja entrar no mercado de trabalho? Obtenha sua certificação aqui

Arquivos padrões - *Standard stdin, stdout* e *stderr*

Antes de iniciarmos a escrita nos arquivos, se faz necessário explicar alguns detalhes sobre como os arquivos são vistos em programação C.

A troca de informações em um sistema pode se dar de várias maneiras.
Até o momento em nosso curso de C, fizemos isso através da interação teclado-programa-tela, onde o usuário fornece os dados via teclado, o programa processa essa informação, e exibe algo na tela.

Outra maneira de trocar informações é através da impressora ou de um scanner.
Ou seja, podemos receber informações através de um dispositivo externo (como um leitor de códigos em barra), bem como podemos mandar dados para uma impressora.

E nesta seção, focaremos em um tipo específico de troca de informações, que é através de arquivos localizados no disco magnético de seu computador. Podemos tanto pegar informações dele, como escrever algo em seu HD através da linguagem C.

Resumindo: existem várias maneiras de se trocar informações, e sempre está surgindo novas. Por exemplo, as tecnologias WiFi e *Bluetooth.* Visando facilitar, as coisas foram padronizadas como sendo arquivos.

Ou seja, quando fornecemos dados por meio de um teclado, essas informações são passadas em série, por meio de *stream*, como se fosse um arquivo, embora este não exista.
Nosso programa em C vai ler esses dados do teclado COMO SE FOSSE um arquivo.
Isso ocorre porque ao criar um programa em C, estamos automaticamente usando alguns tipos de arquivos.

Mesmo que apenas tenhamos a interação teclado-programa-tela, existem alguns arquivos abertos, e os mais importantes são:

- **stdin** - é o arquivo de entrada padrão. Até o momento, para nós, foi o teclado.
- **stdout** - é o arquivo de saída padrão. Até o momento, nossa saída padrão é a tela do computador, através do terminal de comando.
- **stderr** - arquivo padrão de erro, onde podemos direcionar mensagens de erro para outro local sem ser o da saída padrão, como para um *log* (espécie de registro de ações)

Além desses, outros arquivos como o *stdaux* (dispositivo auxiliar, geralmente a porta COM1) e o*stdprn*(impressora padrão), são abertos automaticamente.
Por isso, podemos usar a vontade funções como *printf*, *scanf*, *putc* e outras.

Mas agora não vamos mais usar essas entradas e saídas padrão, e sim arquivos, por isso iremos fornecer algumas informações especiais aos nossos programas, informações sobre as entradas e saídas, que serão em formas de arquivos salvos na sua máquina.

fputc() - Como escrever um caractere em um arquivo

Sem mais delongas, vamos as vias de fato e finalmente aprender a criar e escrever em um arquivo externo de nosso computador.

Para fazer isso vamos usar a função fputc(), que recebe dois dados: o caractere e o *FILE**, que terá as informações do local onde iremos escrever o dito caractere:

int fputc(int char, FILE *arq);

Essa função retorna *EOF* caso não tenha conseguido escrever no arquivo, ou retorna o inteiro que representa o caractere, caso tenha ocorrido.

Exemplo de código - Escrevendo um caractere em um arquivo

Escreva um programa que peça um caractere para o usuário e salve esta entrada em um arquivo chamado "char.txt", localizado na mesma pasta do programa executável.

Vamos inicialmente definir nosso endereço através de uma string (*char url[]*), que é simplesmente "char.txt", bem como o caractere *ch* para armazenar o caractere que o usuário digitar.

Como queremos salvar o *char* em um arquivo, criamos um ponteiro *arq* do tipo *FILE*.
Em seguida, pedimos para o usuário digitar algum caractere, que capturamos através da função*getchar()* e salvamos em *ch*.

Agora vamos abrir um arquivo para escrever, e isso é feito através da função *fopen()*, que vai receber a string com a localização do arquivo (isto está armazenado na variável *url*) e o modo de abertura.
Como queremos escrever, o modo é o *"w"*.

Para checarmos se abertura do arquivo foi realmente efetuada, testamos se a *fopen* não retornou NULL para *arq*. Caso não, vamos escrever o caractere no arquivo.
Isso é feito pela função *fputc*, que diferente da *putchar* que necessita só do caractere, ela também necessita saber onda vai ser a saída (pois não é mais a saída padrão, a tela).

Essa saída é indicada pelo vetor *arq*. E pronto, caractere salvo no arquivo "char.txt".
Mas antes de finalizar o programa, devemos adotar a boa prática de programação que nossa **apostila de C** ensinou: fechar os arquivos, usando a *fclose* e passando o *arq* como argumento.
Nosso código C ficou:

```c
#include <stdio.h>

int main(void)
{
	char url[]="char.txt";
	char ch;
	FILE *arq;

	printf("Caractere: ");
	ch=getchar();

	arq = fopen(url, "w");
	if(arq == NULL)
		printf("Erro, nao foi possivel abrir o arquivo\n");
	else{
		fputc(ch, arq);
```

```
                fclose(arq);
        }

        return 0;
}
```

Pronto, agora vá na pasta onde está localizado seu executável e vai ver que foi criado um arquivo chamado "*char.txt*", e dentro dele está o caractere que você digitou.
Pode fechar o programa, reiniciar a máquina ou fazer o que quiser, que o dito cujo arquivo vai continuar no seu HD.

Bacana, não?
Agora experimente rodar de novo o programa, escrevendo outro caractere.
Veja que o antigo foi substituído. Isso ocorreu devido ao modo de abertura, "w".

Exemplo de código - Adicionando caractere com o modo "a" (*append*)

Crie um programa semelhante ao anterior, mas em vez de substituir o caractere, ADICIONE um caractere ao fim do arquivo.
Escreva no arquivo: "C Progressivo"

Para escrever um texto em sequência, temos que alterar apenas uma coisa: o modo de abertura, de "w" para "a", assim sempre que escrevermos algo, será inserido ao final, anexando, em vez de substituir.

Vamos criar um looping do while para pedir caracteres ao usuário, e este laço só termina se digitarmos enter (caractere '\n').
Como vamos 'mexer' várias vezes no arquivo, abrimos ele antes do looping e fechamos depois.
Nosso código fica:

```
#include <stdio.h>

int main(void)
{
        char url[]="char.txt";
        char ch;
        FILE *arq;

        arq = fopen(url, "a");
        if(arq == NULL)
                printf("Erro, nao foi possivel abrir o arquivo\n");
        else

        do{
                printf("Caractere: ");
                ch=getchar();
                fflush(stdin);

                fputc(ch, arq);
        }while(ch != '\n');

        fclose(arq);

        return 0;
```

```
}
```

Digite caractere por caractere, inclusive espaço, o seguinte: C Progressivo
Em seguida dê enter para terminar.
Veja como ficou seu arquivo *char.txt*

Agora rode o programa de novo, e digite outra coisa qualquer e dê enter ao final.
Veja novamente seu arquivo! Vai ver que o que tinha antes não foi apagado, e o que digitou depois está na linha de baixo.
Interessante, não?

fprintf() - Escrevendo textos (strings) em arquivos

Embora interessante, e importante, a escritura de caracteres em arquivos, ela é um pouco limitada. Afinal, é incômodo escrever caractere por caractere.

Para isso, existe a função *fprintf*, que nos permite escrever strings inteiras em arquivos.
Sua sintaxe simplificada é:
int fprintf(FILE *arq, char string[])

Ou seja, recebe o local onde deve direcionar a saída (para um arquivo, apontado pelo ponteiro *arq* do tipo*FILE*), e a string que devemos adicionar em tal arquivo.
Essa função retorna *EOF* em caso de erro.

Exemplo de código - Armazenando notas, calculando a média e escrevendo no arquivo

Escreva um programa em C que peça 3 notas de um aluno (Matemática, Física e Química), e salve esses dados em um arquivo chamado "notas.txt", que deve ter, ao final, a média das três disciplinas.

Vamos definir a nossa url como "notas.txt" e criar duas variáveis do tipo *float*, a "nota" (que vai armazenar a nota de cada matéria) e a "media" (que vai somar todas as notas, depois se dividir por 3 para ter a média).

A cada vez que pedimos uma nota, adicionamos uma linha de informação ao nosso arquivo, e ao final escrevemos a média.
O interessante é que nossa saída de dados é o arquivo, e não mais a tela. Assim, embora exista saída, nós não vemos na tela, só no arquivo.

Veja como ficou nosso código:

```c
#include <stdio.h>

int main(void)
{
	char url[]="notas.txt";
	float nota,
	      media=0.0;
	FILE *arq;

	arq = fopen(url, "w");
	if(arq == NULL)
```

```
                printf("Erro, nao foi possivel abrir o arquivo\n");
        else{
                printf("Nota de Matematica: ");
                scanf("%f", &nota);
                fprintf(arq, "Matematica: %.2f\n", nota);
                media+=nota;

                printf("Nota de Fisica: ");
                scanf("%f", &nota);
                fprintf(arq, "Fisica: %.2f\n", nota);
                media+=nota;

                printf("Nota de Quimica: ");
                scanf("%f", &nota);
                fprintf(arq, "Quimica: %.2f\n", nota);
                media+=nota;

                media /= 3;
                fprintf(arq, "Media final: %.2f\n", media);
        }
        fclose(arq);

        return 0;
}
```

E se a *fputc* é a equivalente da *putc*, bem como a *fprintf* é a equivalente da *printf*, fica óbvio saber qual a função da *fputs*: é a mesma da *puts*, cuja a saída é uma string com o *new line* "\n" no final.