Função open

Sintaxe:

```
int open(const char * path, int oflag, /* [mode_t mode]*/);
```

Acessado em: http://www.br-c.org/doku.php?id=open, 16 jul 2020

Para se abrir um arquivo deve-se chamar a função **open**. A função recebe o nome do arquivo como parâmetro juntamente com o flag indicando o modo de abertura e flags informando opções adicionais de abertura e/ou tratamento de arquivo. Pode-se colocar todo o caminho do arquivo juntamente com o seu nome.

A função irá retorna um número inteiro positivo em caso de sucesso no processo de abertura. Este número é o descritor do arquivo e deve ser armazenado e passado para as demais funções a serem realizadas no arquivo.

O sistema Unix/Linux garante que o número retornado é o menor número de descritor ainda não utilizado pelo sistema. Isto pode ser utilizado dentro de um programa de maneira a abrir um outro arquivo para a Entrada Padrão, Saída Padrão ou Erro padrão. Por exemplo, ao se fechar a Entrada Padrão, descritor 1, esta se disponibilizando o número 1 como sendo o menor descritor. Ao se abrir um novo arquivo, o número será usado para este arquivo. Como as funções do sistema sempre gravam as suas informações na Saída Padrão, estarão gravando no arquivo e não mais na tela.

A função no processo de abertura irá verificar se o arquivo existe, se o usuário tem permissão sobre o diretório onde o arquivo se encontra e também se tem permissão para realizar a operação indicada no parâmetro (leitura/gravação).

Caso ocorra algum erro na abertura a função irá retornar o valor -1 como resultado. Na variável errno estará indicado o erro ocorrido na abertura.

O segundo parâmetro da função **open** irá indicar qual é o modo de abertura sendo realizado no arquivo. Deve-se obrigatoriamente indicar um e somente um dos seguintes flags :

- **O_RDONLY** - O arquivo está sendo aberto para leitura. Caso não se coloque nenhum flag adicional e caso o arquivo não exista a função de abertura retornará um erro.
- **O_WRONLY** - O arquivo está sendo aberto para gravação. Caso o arquivo não exista e não se coloque nenhum flag adicional, a função retornará um erro.
- **O_RDWR** - O arquivo estará sendo aberto para leitura e gravação. Também neste caso, se o arquivo não existir e nenhum flag adicional foi colocado a função retorna erro.

Pode-se colocar alguns flags adicionais com um dos flags acima definidos. Estes flags irão determinar as ações a serem realizadas no processo de abertura do arquivo e devem ser colocadas com or binário no segundo parâmetro.

Os flags possíveis são:

- **O_APPEND** - Cada gravação a ser realizada no arquivo será feita no final do mesmo.
- **O_CREAT** - Caso o arquivo não exista ele será criado no diretório. Esta opção exige que se coloque no terceiro parâmetro o modo de proteção a ser usado na criação do arquivo.

- **O_EXCL** - Este flag deve ser usado em conjunto com o flag **O_CREAT**. Ele indica para a função open retornar um erro caso o arquivo já exista.
- **O_TRUNC** - Se o arquivo existir no diretório, o conteúdo do mesmo será eliminado, deixando o arquivo com o tamanho de zero bytes.
- **O_SYNC** - Indica para o Unix que cada gravação espere que a gravação física dos dados seja efetuada. Isto garante que as informações gravadas pelo processo sejam efetivamente gravadas em disco, evitando perda de informações caso o processo seja cancelado. Em contrapartida o desempenho do processo será seriamente prejudicado.

O terceiro parâmetro deve ser fornecido caso na abertura se tenha especificado o flag **O_CREAT**. Este terceiro parâmetro indica a máscara de proteção a ser usada na criação do arquivo. Pode-se especificar mais de um flag neste campo, bastando fazer-se or binário entre os seguintes valores:

- **S_IRUSR** - Leitura para usuário
- **S_IWUSR** - Gravação para usuário
- **S_IXUSR** - Execução para usuário
- **S_IRRGRP** - Leitura para grupo
- **S_IWGRP** - Gravação para grupo
- **S_IXGRP** - Execução para grupo
- **S_IROTH** - Leitura para outros
- **S_IWOTH** - Gravação para outros
- **S_IXOTH** - Execução para outros

Pode-se também indicar a proteção a ser usada no arquivo usando-se o modo numérico octal de permissão aceito pelo sistema Unix (veja no help do sistema o comando chmod). As permissões de um arquivo sofrem a influência da configuração atual do sistema, veja no help do sistema o funcionamento do comando umask.

Veja o exemplo:

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <fcntl.h>
#include <string.h>

int main (int argc, char  *argv[])
{
   int iFileDescriptor; /* Descritor do arquivo a ser aberto */

   printf("Tentando abrir o arquivo 'arquivo_teste_1' para leitura\n");

   iFileDescriptor = open ("arquivo_teste_1", O_RDONLY);
/* Abrindo um arquivo em modo leitura. Se este arquivo não existir no diretório
local, ocasionara em erro na abertura.. */

   if (iFileDescriptor < 0)
   {
      fprintf (stderr, "Erro : %s\n", strerror(errno));
   }
   else
   {
      printf ("Arquivo 1 aberto\n");
   }
```

```
   close (iFileDescriptor);

   printf("Tentando abrir o arquivo 'arquivo_teste_2' para gravação\n");

   iFileDescriptor = open("arquivo_teste_2", O_WRONLY);
/* Abrindo um arquivo em modo gravação. Se este arquivo não existir no diretório
local, ocorrerá em erro na abertura... */

   if (iFileDescriptor < 0)
   {
      fprintf(stderr, "Erro : %s\n", strerror(errno));
   }
   else
   {
      printf("Arquivo 2 aberto\n");
   }

   close(iFileDescriptor);

   printf("Tentando abrir o arquivo 'arquivo_teste_3' para leitura\n");

   iFileDescriptor = open("arquivo_teste_3", O_RDONLY | O_CREAT, 0744);
/* Abrindo um arquivo para leitura que não existe. Como foi utilizado a flag
O_CREAT, se o arquivo não existir ele será criado com a permissão 0744 (leitura,
gravação e execução para usuário e leitura para grupo e outros). */

   if (iFileDescriptor < 0)
   {
      fprintf(stderr, "Erro : %s\n", strerror(errno));
   }
   else
   {
      printf("Arquivo 3 aberto\n");
   }

   close (iFileDescriptor);

   printf("Tentando abrir o arquivo 'arquivo_teste_4' para gravacao\n");

   iFileDescriptor = open("arquivo_teste_4", O_WRONLY | O_TRUNC);
/* Abrindo um arquivo para gravação. Este arquivo não existe no diretório local,
ocasionando em erro na abertura. Se o arquivo existisse, como foi especificado a
flag O_TRUNC, o arquivo seria truncado (zerado). */

   if (iFileDescriptor < 0)
   {
      fprintf (stderr, "Erro : %s\n", strerror(errno));
   }
   else
   {
      printf ("Arquivo 4 aberto\n");
   }

   close (iFileDescriptor);

   printf("Tentando abrir o arquivo 'arquivo_teste_5' para gravacao\n");

   iFileDescriptor = open ("arquivo_teste_5", O_WRONLY | O_CREAT | O_EXCL,
0755);
/* Abrindo um arquivo para gravação com os flags O_CREAT e O_EXCL,
ou seja, o arquivo somente será criado se já não existir no sistema de
arquivos. */
```

```
   if (iFileDescriptor < 0)
   {
      fprintf (stderr, "Erro : %s\n", strerror(errno));
   }
   else
   {
      printf ("Arquivo 5 aberto\n");
   }

   close (iFileDescriptor);
   return 0;
}
```