# ARQUIVOS

**//Modos de abertura de arquivos**

| Modo | Significado |
|---|---|
| a | Abre um arquivo para operações de anexação – se um arquivo não existir, o sistema operacional o criará |
| r | Abre um arquivo existente para operações de leitura |
| w | Abre um novo arquivo para saída – se um arquivo com o mesmo nome existir, o sistema operacional irá sobrescrever o arquivo |
| r+ | Abre um arquivo existente para leitura e gravação |
| w+ | Abre um novo arquivo para leitura e gravação – se um arquivo com o mesmo nome existir, o sistema operacional irá sobrescrever o arquivo |
| a+ | Abre um arquivo para operações de anexação e leitura – se um arquivo não existir, o sistema operacional criará o arquivo |

**//Abertura de arquivo (abertura.c)**

```
#include <stdio.h>

int main(void)
{
  FILE *pa;   // ponteiro para uma estrutura do tipo FILE

  if((pa = fopen("NOMEARQ.EXT","r"))!= NULL)
    printf("Arquivo NOMEARQ.EXT aberto com sucesso\n");
  else
    printf("Erro ao abrir NOMEARQ.EXT\n");
}
```

**//Fechamento de arquivo (fechar.c)**

```
#include <stdio.h>

int main(void)
{
  FILE *pa = fopen("teste.c","r");

  if (fclose(pa)==EOF)
    printf("Erro ao fechar o arquivo/n");
}
```

```
//Lendo e gravando em arquivos (copia.c)
#include <stdio.h>

int main(void)
{
  FILE *entrada, *saida;
  int letra;

  if ((entrada = fopen("teste.c", "r"))==NULL)
    printf("Erro ao abrir teste.c\n");
  else
    if ((saida = fopen("teste.bak", "w"))==NULL)
      printf("Erro ao abrir teste.bak\n");
    else
    {       // Le e grava cada caractere no arquivo
      while ((letra = fgetc(entrada)) != EOF)
        fputc(letra, saida);
      fclose(entrada);     // Fecha o arquivo entrada
      fclose(saida);         // Fecha o arquivo saida
    }
}
//Executando saída formatada em arquivo (formatado.c)
#include <stdio.h>

int main(void)
{
  FILE *pa;
  int paginas = 800;
  float preco = 79.95;

  if (pa = fopen("FPRINTF.DAT", "w"))
  {
    fprintf(pa, "Programando C/C++\n");
    fprintf(pa, "Páginas: %d\n", paginas);
    fprintf(pa, "Preço: $%5.2f\n", preco);
    fclose(pa);
  }
  else
    printf("Erro ao abrir FPRINTF.DAT\n");
}
//Renomeando
        int rename(char *nome_antigo, char *nome_novo);
if (rename("ARQ.BAK","ARQ.TXT")) printf("Erro ao renomear o arquivo/n");
//Excluindo
        int remove(char *nome_arquivo);
if (remove("ARQ.TXT")) printf("Erro ao remover o arquivo/n");
```

```
//Fechando todos os arquivos de uma só vez (inclusive stdin, stdout e stderr)
        int fcloseall();

//fwrite, fread e rewind (writeRead.c)
#include <stdio.h>
#include <string.h>

int main(void)
{
   int dia = 7, a;
   float salario = 257.54, b;
   char *nome = "Programa com fprintf", *c;
   FILE *saida;

  if ((saida = fopen("SAIDA.SAI", "w+")) == NULL)
     fprintf(stderr, "Erro ao abrir o arquivo SAIDA.SAI\n");
  else
  {
     fwrite(&dia, sizeof(int), 1, saida);
     fwrite(&salario, sizeof(float), 1, saida);
     fwrite(&nome, strlen(nome), 1, saida);
  }
  rewind(saida);    /*reposiciona no inicio do arquivo */

  fread(&a, sizeof(int), 1, saida);
  fread(&b, sizeof(float), 1, saida);
  fread(&c, strlen(nome), 1, saida);
  printf("Dia: %d\nSalario: %f\nNome: %s\n", a, b, c);
  fclose(saida);
}
//fgets e fputs (para arquivos de texto)
        char *fgets(char string, int limite, FILE *arquivo);

        char *fputs(char *string, FILE *arquivo);
```

**Utilizando fgets e fputs para fazer uma cópia de arquivos de texto (getsPuts.c)**

```
#include <stdio.h>

void main(int argc, char **argv){ /* argc = numero de argumentos */
  FILE *entrada, *saida;          /* argv[] = argumentos */
  char string[256];

  if ((entrada = fopen(argv[1], "r")) == NULL)
    printf("Erro ao abrir %s\n", argv[1]);
  else
    if ((saida=fopen(argv[2], "w")) == NULL){
      printf("Erro ao abrir %s\n", argv[2]);
      fclose(entrada);}
    else{
      while (fgets(string, sizeof(string), entrada))
        fputs(string, saida);
      fclose(entrada);
      fclose(saida);}
}
```

**Testando o final de um arquivo ao ler e imprimindo em stdout**

```
  while (!feof(entrada))
    fputc(fgetc(entrada),stdout);
```

**fscanf (fscanf.c)**

```
#include <stdio.h>

void main(void){
  FILE *pa;
  int idade;
  float salario;
  char nome[64];

  if ((pa = fopen("DADOS.DAT", "w")) == NULL)
    printf("Erro ao abrir DADOS.DAT para saída\n");
  else{
    fprintf(pa, "33 3500.0 Programa");
    fclose(pa);
    if ((pa = fopen("DADOS.DAT", "r")) == NULL)
      printf("Erro ao abrir DADOS.DAT para entrada\n");
    else{
      fscanf(pa, "%d %f %s", &idade, &salario, nome);
      printf("Idade %d Salário %f Nome %s\n", idade, salario, nome);
      fclose(pa);}}
}
```

## Posicionamento do ponteiro de arquivo com base em sua posição atual

```
#include <stdio.h>

int fseek(FILE *canal, long deslocamento, int relativo_a);
```

| constante | significado |
|---|---|
| SEEK_CUR | Da posição atual no arquivo |
| SEEK_SET | Do início do arquivo |
| SEEK_END | Do final do arquivo |

```
fseek(pa, 256, SEEK_SET); // 256 bytes após. O deslocamento 0 é o início
```

## Do teclado para o disco (diretoParaDisco.c)

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]){
  FILE *fp;
  char ch;

  if(argc!=2) {
    printf("voce esqueceu de entrar o nome do arquivo\n");
    exit(1);
  }
  if((fp=fopen(argv[1], "w"))==NULL) {
    printf("arquivo nao pode ser aberto\n");
    exit(1);
  }
  do {
    ch = getchar();
    putc(ch, fp);
  } while(ch!='$');
  fclose(fp);
}
```

## Uma pequena agenda (.c)

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>

#define SIZE 100
```

```
struct list_type {
  char name[40];
  char street[40];
  char city[30];
  char state[3];
  char zip[10];
} list[SIZE];

int menu(void);
void init_list(void),
enter(void);
void display(void),
save(void),
load(void);

int main(void){
  char choice;

  init_list();

  for(;;) {
    choice = menu();
    switch(choice) {
      case 'i': enter();
        break;
      case 'v': display();
        break;
      case 's': save();
        break;
      case 'c': load();
        break;
      case 't': exit(0);
    }
  }
}
void init_list(void)     /* Inicializa a lista. */
{
  register int t;
  for(t=0; t<SIZE; t++) *list[t].name = '\0';
  /* um nome de comprimento zero significa vazio */
}
void enter(void){        /* poe os nomes na lista */
  register int i;

  for(i=0; i<SIZE; i++)
    if(!*list[i].name) break;
```

```
  if(i==SIZE) {
    printf("lista cheia\n");
    return;
  }

  printf("nome: ");
  gets(list[i].name);

  printf("rua: ");
  gets(list[i].street);

  printf("cidade: ");
  gets(list[i].city);

  printf("estado: ");
  gets(list[i].state);

  printf("CEP: ");
  gets(list[i].zip);
}
void display(void){      /* mostra a lista */
  register int t;

  for(t=0; t<SIZE; t++) {
    if(*list[t].name) {
      printf("%s\n", list[t].name);
      printf("%s\n", list[t].street);
      printf("%s\n", list[t].city);
      printf("%s\n", list[t].state);
      printf("%s\n\n", list[t].zip);
    }
  }
}
void save(void){         /* Salva a lista. */
  FILE *fp;
  register int i;

  if((fp=fopen("maillist", "wb"))==NULL) {
    printf("arquivo nao pode ser aberto\n");
    return;
  }

  for(i=0; i<SIZE; i++)
    if(*list[i].name)
      if(fwrite(&list[i],
        sizeof(struct list_type), 1, fp)!=1)
          printf("erro de gravao no arquivo\n");
```

```
}
void load(void){        /* Carrega o arquivo */
  FILE *fp;
  register int i;

  if((fp=fopen("maillist", "rb"))==NULL) {
    printf("arquivo nao pode ser aberto\n");
    return;
  }

  init_list();
  for(i=0; i<SIZE; i++)
    if(fread(&list[i],
      sizeof(struct list_type), 1, fp)!=1) {
        if(feof(fp)) return;
        printf("erro de leitura no arquivo\n");
    }

  fclose(fp);
}
menu(void){             /* obtem uma selecao do menu */
  char s[80];

  do {
    printf("(I)nserir\n");
    printf("(V)isualizar\n");
    printf("(C)arregar\n");
    printf("(S)alvar\n");
    printf("(T)erminar\n");
    printf("escolha: ");
    gets(s);
  } while(!strchr("ivcst", tolower(*s)));
  return tolower(*s);
}
```