



ARQUIVOS

ECM404

- ❑ Por que usar arquivos?
 - ❑ Permitem armazenar grande quantidade de informação;
 - ❑ Persistência dos dados (disco);
 - ❑ Acesso aos dados poder ser não sequencial;
 - ❑ Acesso concorrente aos dados (mais de um programa pode usar os dados ao mesmo tempo).

MANIPULANDO OS ARQUIVOS

- ❑ A linguagem C possui uma série de funções para manipulação de arquivos, cujos protótipos estão reunidos na biblioteca padrão de entrada e saída, ***stdio.h***.
- ❑ Suas funções se limitam a abrir/fechar e ler caracteres/bytes
- ❑ É tarefa do programador criar a função que lerá um arquivo de uma maneira específica.

MANIPULANDO OS ARQUIVOS

- ❑ Todas as funções de manipulação de arquivos trabalham com o conceito de "ponteiro de arquivo". Podemos declarar um ponteiro de arquivo da seguinte maneira:

```
int main (int argc, char *argv[]){  
    FILE *p;  
  
    return 0;  
}
```

ABRINDO OS ARQUIVOS

- ❑ Para a abertura de um arquivo, usa-se a função ***fopen***:

```
FILE *fopen(char *nome_arquivo, char *modo);
```

- ❑ O parâmetro **nome_arquivo** determina qual arquivo deverá ser aberto. Este parâmetro pode trabalhar com caminhos absolutos ou relativos:

- ❑ **Caminho absoluto:** descrição de um caminho desde o diretório raiz

- ❑ C:\\Projetos\\dados.txt

- ❑ **Caminho relativo:** descrição de um caminho desde o diretório corrente (onde o programa está salvo)

- ❑ arq.txt

- ❑ ../dados.txt

ABRINDO OS ARQUIVOS

- ❑ Para a abertura de um arquivo, usa-se a função ***fopen***:

```
FILE *fopen(char *nome_arquivo, char *modo);
```

- ❑ O parâmetro **modo** determina que tipo de uso será feito do arquivo

Modo	Função
"r"	Leitura. <ul style="list-style-type: none">• Arquivo deve existir
"w"	Escrita. <ul style="list-style-type: none">• Cria arquivo se não houver.• Apaga o anterior se ele existir
"a"	Escrita. <ul style="list-style-type: none">• Os dados serão adicionados no fim do arquivo ("append")

ABRINDO OS ARQUIVOS

- ❑ Um arquivo de nome ***teste.csv*** pode ser aberto para escrita da seguinte maneira:

```
int main (int argc, char *argv[]){  
    FILE *p;  
    p = fopen("teste.csv", "w");  
  
    if(p == NULL)  
        printf("Erro ao abrir o arquivo. \n");  
  
    fclose(p);  
    return 0;  
}
```

ABRINDO OS ARQUIVOS

- ❑ Um arquivo de nome **teste.csv** pode ser aberto para escrita da seguinte maneira:

```
int main (int argc, char *argv[]){  
    FILE *p;  
    p = fopen("teste.csv", "w");  
  
    if(p == NULL)  
        printf("Erro ao abrir o arquivo. \n");  
  
    fclose(p);  
    return 0;  
}
```

- ❑ A condição **p == NULL** testa se o arquivo foi aberto com sucesso. No caso de erro, a função **fopen** retorna o valor **NULL**.

ABRINDO OS ARQUIVOS

- ❑ Um arquivo de nome ***teste.csv*** pode ser aberto para escrita da seguinte maneira:

```
int main (int argc, char *argv[]){  
    FILE *p;  
    p = fopen("teste.csv", "w");  
  
    if(p == NULL)  
        printf("Erro ao abrir o arquivo. \n");  
  
    fclose(p);  
    return 0;  
}
```

- ❑ Sempre que terminamos de usar um arquivo que abrimos, devemos fechá-lo. Para isso, usa-se a função ***fclose***.
- ❑ A função ***fclose*** retorna **zero** no caso de sucesso ao fechar o arquivo.

ESCRITA E LEITURA DE CARACTERES

❑ É possível ler/escrever um único caractere em um arquivo.

❑ Para escrever, podemos utilizar a função ***fputc***

```
int fputc (int ch, FILE *fp);
```

❑ Para ler, podemos utilizar a função ***fgetc***

```
int fgetc (FILE *fp);
```

❑ **CUIDADO!** Tanto na escrita quanto na leitura, após efetuar a ação, o ponteiro se posiciona automaticamente no próximo caractere do arquivo

ESCRITA E LEITURA DE CARACTERES

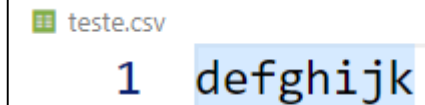
```
int main (int argc, char *argv[]){
    FILE *arq;
    char s[] = "abcdefgh";
    arq = fopen("teste.csv", "w");

    if(arq == NULL)
        printf("Erro ao abrir o arquivo. \n");

    for(int i=0; i<8;i++){
        printf("c[%i]: %c Escrevendo: %c\n",i,s[i],s[i]+3);
        fputc(s[i]+3,arq);
    }

    fclose(arq);
    return 0;
}
```

c[0]:	a	Escrevendo:	d
c[1]:	b	Escrevendo:	e
c[2]:	c	Escrevendo:	f
c[3]:	d	Escrevendo:	g
c[4]:	e	Escrevendo:	h
c[5]:	f	Escrevendo:	i
c[6]:	g	Escrevendo:	j
c[7]:	h	Escrevendo:	k



teste.csv

```
1 defghijk
```

ESCRITA E LEITURA DE CARACTERES

```
int main (int argc, char *argv[]){  
    FILE *arq;  
    char c;  
    arq = fopen("teste.csv", "r");  
  
    if(arq == NULL)  
        printf("Erro ao abrir o arquivo. \n");  
  
    for(int i=0; i<8;i++){  
        c = fgetc(arq);  
        printf("Recebido: %c\n",c);  
    }  
  
    fclose(arq);  
    return 0;  
}
```

Recebido: d
Recebido: e
Recebido: f
Recebido: g
Recebido: h
Recebido: i
Recebido: j
Recebido: k

ESCRITA E LEITURA DE CARACTERES

- ❑ Assim que não houverem mais caracteres a serem lidos, a função ***fgetc*** retornará a constante ***EOF (end of file)***, definida na biblioteca ***stdio.h***.
- ❑ Em muitos computadores esta constante vale -1.

```
if(c == EOF)
    printf("Fim do arquivo!\n");
```

- ❑ No entanto, podemos também utilizar a função ***feof*** para verificar se um arquivo chegou ao fim.

```
int feof(FILE *fp);
```

ESCRITA E LEITURA DE CARACTERES

```
int main (int argc, char *argv[]){
    FILE *arq;
    char c;
    arq = fopen("teste.csv", "r");

    if(arq == NULL)
        printf("Erro ao abrir o arquivo. \n");

    while(1){
        c = fgetc(arq);
        printf("Recebido: %c\n",c);
        if(feof(arq))
            break;
    }

    fclose(arq);
    return 0;
}
```

```
Recebido: d
Recebido: e
Recebido: f
Recebido: g
Recebido: h
Recebido: i
Recebido: j
Recebido: k
```

ESCRITA E LEITURA DE LINHAS

- ❑ Para realizar a leitura de uma linha do arquivo podemos utilizar a função *fgets*. Temos que lembrar de tirar o '\n' do final da linha.

```
fgets(linha, MAX, arq);
```

```
void lerLinha(char linha[], FILE *arq){  
    fgets(linha, MAX, arq);  
    if(linha[strlen(linha)-1] == '\n')  
        linha[strlen(linha)-1] = '\0';  
}
```

ESCRITA E LEITURA DE LINHAS

```
int main (int argc, char *argv[]){  
    FILE *arq;  
    char linha[MAX];  
  
    arq = fopen("teste.csv", "r");  
  
    if(arq == NULL){  
        printf("Erro ao abrir o arquivo. \n");  
        exit(EXIT_FAILURE);  
    }  
}
```

teste.csv

```
1  titulo coluna 1;titulo coluna 2;titulo coluna 3  
2  valor 1; valor 2; valor 3  
3  valor 4; valor 5; valor 6
```


ESCRITA E LEITURA DE LINHAS

```
//Realiza a leitura do cabeçalho do arquivo
lerLinha(linha,arq);
printf("%s\n",linha);

//Continua a leitura das demais linhas até o final do arquivo
while(1){
    lerLinha(linha,arq);
    printf("%s\n",linha);

    if(feof(arq))
        break;
}

fclose(arq);
return 0;
}
```

```
titulo coluna 1;titulo coluna 2;titulo coluna 3
valor 1; valor 2; valor 3
valor 4; valor 5; valor 6
```

ESCRITA E LEITURA DE LINHAS

- ❑ Entretanto, em alguns casos, após realizar a leitura de uma linha, temos que converter o valor armazenado para seus respectivos valores numéricos através das funções ***atoi*** ou ***atof***.

```
void separarInteiros(char linha[], int quantidade, int valores[]){  
    char *token = strtok(linha,";");  
    for(int i=0;i<quantidade;i++){  
        valores[i] = atoi(token);  
        token = strtok(NULL,";");  
    }  
}
```

ESCRITA E LEITURA DE LINHAS

```
//Continua a leitura das demais linhas até o final do arquivo
while(1){
    lerLinha(linha,arq);
    printf("%s\n",linha);
    separarInteiros(linha,3,vet);
    if(feof(arq))
        break;
}

for(int i=0;i<3;i++){
    printf("vet[%i] = %i\n",i,vet[i]);
}
```

```
titulo coluna 1;titulo coluna 2;titulo coluna 3
1;2;3
vet[0] = 1
vet[1] = 2
vet[2] = 3
```

ESCRITA E LEITURA DE LINHAS

- ❑ Quando os dados forem lidos ou gravados no arquivo para posterior acesso em Excel, os valores reais (float) podem utilizar o caractere “ , ”, enquanto para cálculos na programação devemos utilizar o “ . ”.
- ❑ Esta troca deve ser feita após a string ser lida do arquivo e antes de ser gravada no arquivo.

```
void troca(char linha[], char antigo, char novo){  
    for(int i=0; i<strlen(linha); i++){  
        if(linha[i] == antigo)  
            linha[i] = novo;  
    }  
}
```

ESCRITA E LEITURA DE LINHAS

```
//Continua a leitura das demais linhas até o final do arquivo
while(1){
    lerLinha(linha,arq);    //original
    printf("%s\n",linha);

    troca(linha,',','.');    //convertendo para .
    printf("%s\n",linha);

    troca(linha,'.','');    //convertendo para ,
    printf("%s\n",linha);

    if(feof(arq))
        break;
}
```

```
titulo coluna 1;titulo coluna 2;titulo coluna 3
1,1;2,2;3,3
1.1;2.2;3.3
1,1;2,2;3,3
```

ESCRITA E LEITURA DE LINHAS

- ❑ A escrita em arquivos pode acontecer, de forma geral, de duas formas:
 - ❑ Escrita direta no arquivo;
 - ❑ Escrita indireta no arquivo. Neste caso, “escrevemos” em uma string antes de se escrever no arquivo.

```
//escrita direta  
fprintf(arq, "%s;%s;%s\n", "x1", "x2", "x3");
```

```
//escrita indireta  
sprintf(linha, "%i;%i;%i\n", 1, 2, 3);  
fprintf(arq, "%s", linha);
```

ATIVIDADE

- ❑ Abra a pasta Downloads no VS Code
 - ❑ Utilize o comando `git clone https://github.com/ECM404/atividade4.git --recursive`
 - ❑ Entre na pasta utilizando o comando `cd atividade4`
 - ❑ Inicialize o diretório com o comando `make install`