Pesquisa e Ordenação de Dados

Unidade 3.1:

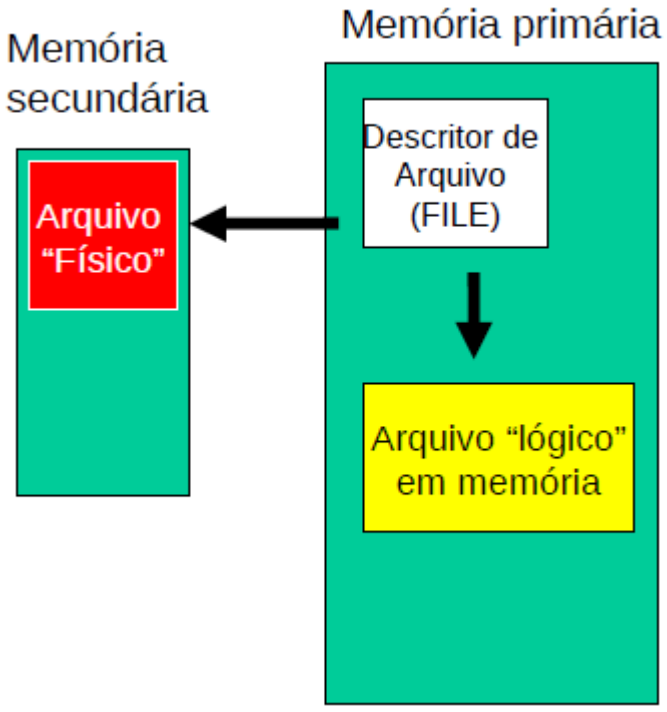
Manipulação de Arquivos em C

Arquivos

- Um arquivo em disco representa um elemento de informação em um dispositivo de memória secundária
  - Modo texto: sequência de caracteres
  - Modo binário: sequência de bytes
    - Mais adequado para a gravação de dados mais complexos, como números, structs, arrays, etc.
- Sistemas operacionais oferecem um conjunto de serviços para ler e escrever informações em disco:
  - *Abertura de arquivo*: o SO encontra o arquivo com o nome dado e prepara o *buffer* de memória;
  - *Leitura de arquivo*: o SO recupera o trecho solicitado do arquivo (parte pode estar no *buffer*);
  - *Escrita no arquivo*: o SO acrescenta ou altera o conteúdo do arquivo. A alteração é feita primeiramente no *buffer* para depois ser transferida para o disco;
  - *Fechamento do arquivo*: toda a informação constante no *buffer* é atualizada no disco e a área de memória do *buffer* é liberada.
- O SO também mantém um cursor que indica a posição atual no arquivo
  - As operações de leitura e escrita se dão a partir da posição corrente do cursor

Acesso a arquivos em C

- Arquivo “físico”: armazenado em memória secundária.
- Arquivo “lógico”: estruturas e dados associados ao arquivo presentes na memória principal.
- Descritor de arquivo:
  - Estrutura de dados denominada **FILE**
  - Representa uma abstração do arquivo
- Criando um descritor:
  - `FILE *ponteiroArquivo;`
- Todas as funções da biblioteca padrão de C que manipulam arquivos encontram-se na biblioteca <stdio.h>



Abrindo um arquivo

- Função **fopen**

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

  - Associa um descritor a um arquivo físico.
  - Para o nome do arquivo podemos usar caminho relativo (sistema procura a partir da pasta atual) ou absoluto (endereço completo desde a raiz; usar duas barras \\ para separar as pastas).
  - Retorna um ponteiro para o tipo FILE. Se não puder ser aberto, retorna NULL.

```
...
FILE *fp = fopen("entrada.txt", "rt");
if (fp == NULL) {
  printf("Erro na abertura do arquivo!");
  exit(1); // aborta o programa
}
...
```

Modos de Abertura

- **r** (*read*): modo de leitura (arquivo já existente)
- **w** (*write*): modo de escrita (se não existe cria; se existe sobrescreve)
- **a** (*append*): modo e escrita no final do arquivo (se não existe cria; se existe não sobrescreve)
- **+**: para ler/ escrever simultaneamente
  - r+ indica leitura/escrita em um arquivo existente
  - w+ indica leitura/escrita em um novo arquivo
- **t** (*text*): modo texto
- **b** (*binary*): modo binário

Fechando um arquivo

- Função **fclose**

```
int fclose(FILE *fp)
```

  - Retorna zero se foi fechado com sucesso, ou a constante EOF (*end of file*), indicando a ocorrência de um erro.
  - Quando um programa não precisa mais utilizar um arquivo por um período de tempo longo, o arquivo pode ser fechado.
  - Fechar um arquivo protege os dados, garante que atualizações feitas serão salvas e libera o arquivo para que outros usuários ou programas possam utilizá-lo.

Fim do arquivo

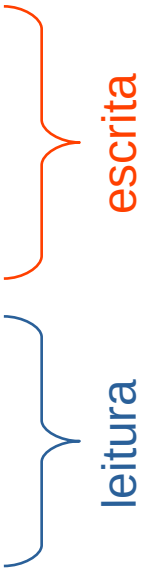
- Função **feof**

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

  - Retorna:
    - 0 (falso), se ainda não chegamos ao final do arquivo;
    - inteiro diferente de 0 (verdadeiro) se a posição corrente para o arquivo indicado é o final do arquivo.

Funções de leitura/escrita

- Modo texto
  - fputc
  - fputs
  - fprintf
  - fgetc
  - fgets
  - fscanf



- Modo binário
  - fwrite //escrita
  - fread //leitura

Arquivos em Modo Texto - Escrita

- Função **fputc**

```
int fputc(char c, FILE *fp)
```

  - Escreve um único caractere no arquivo.
  - Retorna o próprio caractere escrito em caso de sucesso, ou a constante **EOF** (*end of file*) em caso de erro de escrita.

```
...
char texto[20] = "Exemplo de texto";
for (int i=0; i<strlen(texto); i++){
   fputc(texto[i], fp);
   fputc(' ', fp);
}
...
```

Arquivos em Modo Texto - Leitura

- Função **fgetc**

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

  - Lê o próximo caractere do arquivo.
  - A cada leitura, os dados são transferidos para a memória e o cursor avança para o próximo dado.
  - Retorna o código do caractere lido em caso de sucesso, ou a constante EOF em caso de erro ou fim do arquivo.

```
while(!feof(fp))   {
   c = fgetc(fp);
   printf("%c", c);
}
```

Arquivos em Modo Texto - Escrita

- Função **fputs**

```
int fputs(char *str, FILE *fp)
```

  - Escreve uma string no arquivo.
  - Retorna um valor diferente de zero em caso de sucesso, ou EOF em caso de erro de escrita.

```
char texto[20] = "Exemplo de texto";
fputs(texto, fp);
```

Arquivos em Modo Texto - Leitura

- Função **fgets**

```
char* fgets(char *str, int size, FILE *fp)
```

  - Lê uma sequência de caracteres do arquivo até que um caractere de quebra de linha ‘\n’ seja encontrado ou que o número máximo de caracteres (size - 1) seja atingido
  - O conteúdo lido será armazenado na variável str.
  - Retorna um ponteiro para o primeiro caractere da string lida em caso de sucesso, ou NULL em caso de erro de leitura

```
char texto[20];
fgets(texto, 20, fp);
printf("%s", texto);
```

Arquivos em Modo Texto - Escrita

- Função **fprintf**

```
int fprintf(FILE *fp, "formato", variáveis)
```

  - Similar ao printf, porém, escreve para o arquivo indicado por fp.
  - Retorna o número de bytes escritos.

```
char nome[40] = "Pesquisa e Ordenação de Dados";
char cod[7] = "GEX609";
int cr = 4;
fprintf(fp, "%s - %s (%d créditos)\n", cod, nome, cr);
```

Arquivos em Modo Texto - Leitura

- Função **fscanf**

```
int fscanf(FILE *fp, "formato",
variáveis)
```

  - Dados são capturados de um arquivo previamente aberto para leitura.
  - Retorna o número de dados lidos com sucesso.

```
char nome[40];
char cod[7];
int cr;
fscanf(fp, "%s - %s (%d créditos)\n", cod, nome, &cr);
printf("Leu: %s\n%s\n%d\n", cod, nome, cr);
```

Arquivos em Modo Binário - Escrita

- Função **fwrite**

```
int fwrite(void *buffer, int size, int count, FILE *fp)
```

  - Grava um conjunto de dados em um arquivo binário.
  - Deve-se informar um ponteiro genérico para o dado a ser gravado, o tamanho (em bytes) de cada unidade do dado, o número de unidades a serem gravadas e o ponteiro para o arquivo.
  - Retorna o número de elementos gravados. Em caso de sucesso, este número deve ser igual a count.

```
int total, v[5] = {1, 2, 3, 4, 5};
total = fwrite(v, sizeof(int), 5, fp);
if(total != 5)
   printf("Erro na gravação");
```

Arquivos em Modo Binário - Leitura

- Função **fread**

```
int fread(void *buffer, int size, int count, FILE *fp)
```

  - Lê *size* bytes (uma unidade de dado) do arquivo referenciado por *fp*, *count* vezes (total de unidades a serem lidas) e copia para o endereço de memória apontado pelo primeiro parâmetro
  - Retorna o número de elementos gravados. Em caso de sucesso, este número deve ser igual a count

```
int total, v[5];
total = fread(v, sizeof(int), 5, fp);
printf("%d %d %d %d %d",v[0],v[1],v[2],v[3],v[4]);
```

Reposicionar cursor

- Função **fseek**

```
int fseek(FILE *fp, long offset, int origem)
```

  - Em um arquivo binário, temos controle de quantos bytes ocupa cada informação armazenada no arquivo. Com isso, podemos alterar a posição do cursor no arquivo.
  - offset (deslocamento): indica o número de bytes que iremos avançar a partir de “origem”
  - origem:em relação a que posição estamos avançando. Pode conter as constantes:
    - SEEK_SET (ou 0): início do arquivo
    - SEEK_CUR (ou 1): ponto atual do arquivo
    - SEEK_END (ou 2): final do arquivo
  - Retorno: zero em caso de sucesso

```
fseek(fp, 2 * sizeof(int), SEEK_SET);
```

Posição do Cursor

- Função **rewind**

```
void rewind(FILE *fp)
```

  - Desloca o cursor para o início do arquivo.

- Função **ftell**

```
long ftell(FILE *fp)
```

  - Retorna a posição atual do cursor.

Resumo

Função	O que faz?
fopen()	Abre um arquivo.
fclose()	Fecha o arquivo garantindo a transferência do *buffer*.
fflush()	Descarrega o *buffer*.
fscanf()	Leitura de entrada formatada (semelhante ao scanf()).
fprintf()	Escrita de saída formatada (semelhante ao printf()).
fgets()	Obtém uma string do arquivo.
fgetc()	Obtém um caracter do arquivo.
fputs()	Insere uma string no arquivo.
fputc()	Insere um caracter no arquivo.
fread()	Lê um bloco de dados do arquivo.
fwrite()	Escreve um bloco de dados no arquivo.
fseek()	Reposiciona o ponteiro.
rewind()	Reposiciona o ponteiro para o início do arquivo.
ftell()	Retorna a posição do ponteiro.