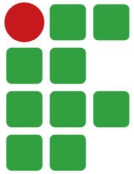

INSTITUTO FEDERAL
Bahia
Campus Santo Antônio de Jesus

http://www.portal.ifba.edu.br/santoantonio

Linguagem de Programação

Prof. George Pacheco Pinto

Agenda

- Linguagem C
  - Manipulação de Arquivos

Arquivos

Texto

- Facilidade de leitura: os dados podem ser lidos por qualquer programa, caractere por caractere.
- Maior gasto de memória: gasta 1 byte por caractere
- Maior gasto de tempo em buscas: para acessar o n-ésimo elemento, exige uma busca seqüencial acessando todos os elementos anteriores no arquivo

Binário

- Menor gasto de memória
- Menor gasto de tempo em buscas: para saber a posição do n-ésimo número fracionário de uma lista de números fracionários, bastaria localizar a posição n*sizeof(float) movendo o cursor do arquivo.
- Dificuldade de leitura: apenas o criador do arquivo sabe como manipulá-lo.

DECLARANDO VARIÁVEL PARA ARQUIVOS

```
FILE *arquivo;
```

- As funções de manipulação de arquivos estão localizadas na biblioteca **stdio.h**

NOMEANDO ARQUIVOS

Exemplo: o arquivo info.txt

Windows: char *arquivo = "c:\\dados\\george\\info.txt";

UNIX: char *arquivo = "c:\/dados\/\/george\/info.txt";

ABERTURA DE ARQUIVOS

Abertura (e fechamento) de arquivos:

```
arquivo = fopen(“nome”, “modo”);
if(arquivo!=0) fclose(arquivo);
```

Onde:

nome = nome do arquivo

modo = tipo do arquivo (txt ou binário), e objetivo de uso (leitura, escrita, anexação)

Obs: o comando fopen retorna um número inteiro: um número maior que zero significa que a abertura foi feita corretamente, 0(zero) indica erro de abertura do arquivo;

Nunca tente fechar um arquivo que não foi aberto!!!

MODOS UTILIZAÇÃO DE ARQUIVOS

Modo	Significado
"r"	Abre um arquivo texto para leitura. O arquivo deve existir antes de ser aberto.
"w"	Abrir um arquivo texto para gravação. Se o arquivo não existir, ele será criado. Se já existir, o conteúdo anterior será destruído.
"a"	Abrir um arquivo texto para gravação. Os dados serão adicionados no fim do arquivo ("append"), se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
"rb"	Abre um arquivo binário para leitura. Igual ao modo "r" anterior, só que o arquivo é binário.
"wb"	Cria um arquivo binário para escrita, como no modo "w" anterior, só que o arquivo é binário.
"ab"	Acrescenta dados binários no fim do arquivo, como no modo "a" anterior, só que o arquivo é binário.
"r+"	Abre um arquivo texto para leitura e gravação. O arquivo deve existir e pode ser modificado.
"w+"	Cria um arquivo texto para leitura e gravação. Se o arquivo existir, o conteúdo anterior será destruído. Se não existir, será criado.
"a+"	Abre um arquivo texto para gravação e leitura. Os dados serão adicionados no fim do arquivo se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
"r+b"	Abre um arquivo binário para leitura e escrita. O mesmo que "r+" acima, só que o arquivo é binário.
"w+b"	Cria um arquivo binário para leitura e escrita. O mesmo que "w+" acima, só que o arquivo é binário.
"a+b"	Acrescenta dados ou cria uma arquivo binário para leitura e escrita. O mesmo que "a+" acima, só que o arquivo é binário

Exemplo abertura

```
int main ()
{
    FILE *arquivo;
    arquivo = fopen ("aula.txt", "r");
    if (arquivo != NULL)
    {
        printf ("Arquivo aberto com sucesso");
        fclose (arquivo);
    }
    else
        printf ("Não foi possível abrir o arquivo");
}
```

ESCRITA EM ARQUIVOS

int fwrite(char *vet, int tam, int num, FILE *arquivo);

***vet**: variável;
tam: o tamanho de cada elemento
num: o número de elementos a ser escrito
***arquivo**: a variável associada ao arquivo

Obs: o comando fwrite retorna um número inteiro: um número maior que zero significa que a escrita foi feita corretamente, 0(zero) indica erro de escrita no arquivo;

EXEMPLO ESCRITA

```
FILE *arq;
int vet[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
arq = fopen ("dados.txt", "wb");
fwrite(vet, sizeof (int), 10, arq);
fclose (arq);
```

LEITURA DE ARQUIVOS

int fread(char *vet, int tam, int num, FILE *arquivo);

Onde:

***vet**: variável a ser lida do arquivo (tipos básicos ou compostos, porém apenas variáveis, e não vetores);

tam: tamanho a ser lido do arquivo;

num: é a quantidade de dados a ser lida(1 para uma só variável, mais para leitura de vetores);

***arquivo**: variável de arquivo

Obs: o comando fread retorna um número inteiro: um número maior que zero significa que a leitura foi feita corretamente, 0(zero) indica erro de leitura do arquivo;

EXEMPLO LEITURA

```
int i, vet[10];
FILE *arq;
arq = fopen ("dados.txt", "rb");
fread(vet, sizeof (int), 10, arq);
for (i=0; i<10; i++)
{
   printf("%d ", vet[i]);
}
fclose(arq);
```

REPOSICIONAMENTO EM ARQUIVOS

rewind() – para arquivos texto e binários

Retorna o indicador de posição do ponteiro FILE em relação ao arquivo **para o seu início.**

void rewind(FILE *arquivo)

REPOSICIONAMENTO EM ARQUIVOS

fseek() - Reposiciona o ponteiro

Move a posição corrente de leitura ou escrita no arquivo de um valor especificado, a partir de um ponto especificado.

```
int fseek (FILE *arquivo, int deslocamento, int origem);
```

```
fseek(arquivo, sizeof(ALUNO)*(posicao), SEEK_SET);
```

Origem	Referência
SEEK_SET	Começo do arquivo
SEEK_CUR	Posição atual
SEEK_END	Fim do arquivo

VERIFICAR FINAL DE ARQUIVO

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

Obs: o comando feof retorna um número inteiro: um número maior que zero significa que o arquivo está no final, e não há mais dados no arquivo, 0(zero) indica que ainda existem dados;

EXEMPLO

Codeblocks

Referências

Consultar ementário.