Algoritmos e Programação II

Arquivos - parte 2

Prof. Viviane Bonadia dos Santos

Tipos de Arquivos

- Arquivos podem ter o mais variado conteúdo, mas do ponto de vista dos programas existem apenas dois tipos de arquivo:
  - **Arquivo texto:** Armazena caracteres que podem ser mostrados diretamente na tela ou modificados por um editor de textos simples
    - **Exemplos:** código-fonte C, documento texto simples, páginas HTML

Tipos de Arquivos

- Arquivos podem ter o mais variado conteúdo, mas do ponto de vista dos programas existem apenas dois tipos de arquivo:
  - **Arquivo binário:** Sequência de bits sujeita às convenções dos programas que o gerou, não legíveis diretamente
    - **Exemplos:** arquivos executáveis, arquivos compactados, documentos do Word

Arquivos Binários - Motivação

- Armazenar dados em arquivos de forma análoga a utilizada em memória permite:
  - Reduzir o tamanho do arquivo;
  - Realizar busca não sequencial;
  - Permite ler ou gravar um bloco de bytes do tamanho desejado.

Números armazenados em modo binário podem ser gravados ou lidos exatamente como se apresentam na memória. Por exemplo, o número 25.678 é armazenado, na memória do computador, ocupando dois bytes (short int). Em modo texto ocuparia 5 bytes e seria armazenado na forma ASCII.

Abrir arquivo binário

```
int main () {
   FILE *arq;
   arq = fopen(“nome do arquivo”, “modo”);
   …
   fclose(arq);
   return 0;
}
```

- nome: string com nome / caminho do arquivo
- modo: string que indica se o arquivo será aberto para leitura, escrita, ambos, ou variações

Abrir arquivo binário

- O modo de abrir um arquivo binário deve ser exatamente uma das seguintes opções:

Modo	Operações	Ponto do arquivo	Observações
rb	Leitura	Início	Se o arquivo não existir ou não puder ser encontrado, então fopen falha.
wb	Escrita	Início	Cria o arquivo, se necessário. Se o arquivo já existir, seu **conteúdo anterior será apagado**.
ab	Escrita	Final	Cria o arquivo se necessário. Se o arquivo já existir, **mantém o conteúdo anterior.**

Abrir arquivo binário

- O modo de abrir um arquivo binário deve ser exatamente uma das seguintes opções:

Modo	Operações	Ponto do arquivo	Observações
r+b	Leitura e escrita	Início	Se o arquivo não existir ou não puder ser encontrado, então fopen falha.
w+b	Leitura e escrita	Início	Cria o arquivo, se necessário. Se o arquivo já existir, seu **conteúdo anterior será apagado**.
a+b	Leitura e escrita	Leitura: início. Escrita: Final.	Cria o arquivo se necessário. Se o arquivo já existir, **mantém o conteúdo anterior.**

Leitura e escrita de dados

- As funções **fread** e **fwrite** permitem a leitura e escrita de blocos de dados
- A ideia é semelhante a alocação de memória dinâmica
- Devemos determinar o número de elementos a serem lidos ou gravados e o tamanho de cada um
- As funções necessárias para manipulação destas funções estão na stdio.h

Leitura de dados

```
fread(void *ptr, size_t size, size_t cont, FILE *stream);
```

- **ptr** representa a variável que vai receber o conteúdo lido do arquivo;
- **size** representa o tamanho do bloco, em bytes, que será lido do arquivo;
- **cont** representa o número de blocos (do tamanho especificado) que será lido;
- **stream** é a referência para o arquivo que será lido.

Leitura de dados

- Lê do arquivo f, num_elem elementos de tam_elem bytes cada e armazena na variável dados
- Retorna o número de elementos lidos com sucesso

```
/* Le um inteiro do arquivo 'myfile.bin' */

FILE *f;
int val;

f = fopen ( "myfile.bin" , "rb" );
fread(&val, sizeof(int), 1, f);
fclose (f);
```

Escrita de dados

```
fwrite(void *ptr, size_t size, size_t cont, FILE *stream);
```

- **ptr** representa a variável que armazena o conteúdo a ser gravado no arquivo;
- **size** representa o total em bytes que será escrito do arquivo;
- **cont** representa o número de blocos (do tamanho size) que será escrito no arquivo;
- **stream** é a referência para o arquivo onde as informações serão escritas.

Escrita de dados

- Escreve no arquivo f, num_elem elementos de tam_elem bytes cada, armazenados na variável dados
- Retorna o número de elementos gravados com sucesso

```
/* Escreve conteudo de buffer no arquivo 'myfile.bin' */

FILE * pFile;
char buffer[] = {'x' , 'y' , 'z'};

pFile = fopen ( "myfile.bin" , "wb" );
fwrite (buffer, sizeof(char), sizeof(buffer), pFile);
fclose (pFile);
```

Acesso direto

- Podemos fazer o acesso não sequencial usando a função fseek
- Esta função altera a posição de leitura/escrita no arquivo
- O deslocamento pode ser relativo ao:
  - Início do arquivo
  - Ponto atual
  - Final do arquivo

Acesso direto

```
fseek(arquivo, deslocamento, referencia);
```

Deslocamento: quantos caracteres desejamos avançar (se positivo) ou retroceder (se negativo) em relação à posição referencial.
Posição referêncial: de onde o deslocamento deve ser realizado:

- **SEEK_CUR:** deslocamento em relação à posição de leitura e escrita atual.
- **SEEK_E D:** deslocamento em relação ao final do arquivo.
- **SEEK_SET:** O deslocamento é em relação ao início do arquivo.

fseek retorna 0 em caso de sucesso.

Avançar e retroceder a posição de leitura ou escrita

- Reposicionar o “indicador” no início do arquivo:

```
void rewind(FILE *f)
```

- Consultar a posição atual da leitura ou da escrita (O valor retornado pela função ftell é sempre do tipo long int):

```
posicao = ftell(arquivo);
```

Acesso direto

- Um arquivo pode armazenar registros (como um banco de dados)
- Isso pode ser feito de forma bem fácil se lembrarmos que um registro, como qualquer variável em C, tem um tamanho fixo
- O acesso a cada registro pode ser direto, usando a função fseek
- A leitura ou escrita do registro pode ser feita usando as funções fread e fwrite