

Introdução a Programação



Arquivos

Tópicos da Aula

- ◆ Hoje aprenderemos a persistir dados
 - Conceito de arquivos
 - Importância de persistência
 - Modos de acesso de arquivos em C
 - Operações em arquivos
 - Funções de leitura/escrita em modo binário

Arquivos

- ◆ Um arquivo representa um elemento de informação armazenado em memória secundária (disco)
- ◆ Características:
 - Informações são persistidas
 - Atribui-se nomes aos elementos de informação (arquivos e diretórios), em vez de endereços de memória
 - Acesso às informações são mais lentos

Persistência... pra quê?

- ◆ Não perder os dados no fim da execução de um programa
 - Memória temporária(volatil)
 - principal
 - Mais rápida e cara
 - Memória permanente
 - secundária
 - mais lenta e barata
- ◆ Para melhorar velocidade de acesso, a cada acesso, transfere-se trechos maiores do arquivo para espaços da memória (buffers)

Modos de Acesso a Arquivos em C

- ◆ Dois modos de acesso:
 - Texto e Binário

String

Informação persistida

Arquivo Texto:

0 1 2 3 4

Inteiros



Arquivo Binario:

0 1 2 3 4



Modo Texto

- ◆ É interpretado como uma seqüência de caracteres agrupadas em linhas
- ◆ Linhas são separadas por um caractere de nova linha
- ◆ Vantagens:
 - Pode ser lido facilmente por uma pessoa
 - Editado por editores de texto convencionais
- ◆ Desvantagens
 - Codificação dos caracteres pode variar (ASCII, UTF-8, ISSO-8859, etc)
 - Arquivos tendem a ser maiores (todas os dados são convertidos para caracteres)

Modo Binário

- ◆ Dados são armazenados da mesma forma que são armazenados na memória principal
- ◆ Vantagens:
 - Facilmente interpretados por programas
 - Maior velocidade de manipulação
 - Arquivos são, geralmente, mais compactos
- ◆ Desvantagens:
 - Difícil de serem entendidos por pessoas
 - Dependentes da máquina onde foram gerados

Operações em Arquivos

◆ Abertura

- Sistema Operacional (SO) encontra arquivo pelo nome
- Prepara buffer na memória

◆ Leitura

- SO recupera trecho solicitado do arquivo
- Parte ou todo trecho pode vir do buffer

Operações em Arquivos

◆ Escrita

- SO altera conteúdo do arquivo
- Alteração é feita primeiro no buffer para depois ser transferida para o disco

◆ Fechamento

- Informação do buffer é atualizada no disco
- Área do buffer utilizada na memória é liberada

Abertura de Arquivos em C

- ◆ Operações de manipulação de arquivos em C se encontram, geralmente, na `stdio.h`
- ◆ Função de Abertura

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

- **Nome**

- Nome do arquivo

- **FILE**

- Tipo estruturado que representa uma abstração do arquivo

- **modo**

- **r** - Indica leitura
 - **w** – Indica escrita
 - **a** – Indica escrita ao final do existente
 - **t** – Indica modo texto
 - **b** – Indica modo binário

Abrindo arquivos

```
FILE *fptr; /* ponteiro para arquivo */  
fptr = fopen("arqtext.txt", "w");
```

nome do arquivo

Tipo de abertura

Modo t pode ser omitido

- "r" Abrir arquivo **texto** para leitura. O arquivo deve estar presente no disco
- "w" Abrir arquivo **texto** para gravação. Se o arquivo existir ele será destruído e reinicializado. Se não existir, será criado
- "a" Abrir um arquivo **texto** para gravação. Os dados serão adicionados no fim do arquivo existente, ou cria um novo

Abrindo arquivos

```
FILE *fptr; /* ponteiro para arquivo */  
fptr = fopen("arqtext.txt", "w+");
```

Tipo de abertura

- "r+" Abrir arquivo texto para leitura e gravação. O arquivo deve existir e pode ser **atualizado**.
- "w+" Abrir arquivo texto para leitura e gravação. Se o arquivo existir ele será **destruído e reinicializado**. Se não existir, será criado.
- "a+" Abrir um arquivo texto para atualização e para adicionar dados no fim do arquivo existente, ou cria um novo

Abrindo arquivos

```
FILE *fptr; /* ponteiro para arquivo */  
fptr = fopen("arqtext.txt", "wb");
```

Tipo de abertura

- "rb" Abrir arquivo binário para leitura. O arquivo deve estar presente no disco
- "wb" Abrir arquivo binário para gravação. Se o arquivo existir ele será destruído e reinicializado. Se não existir, será criado
- "ab" Abrir um arquivo binário para gravação. Os dados serão adicionados no fim do arquivo existente, ou cria um novo

Abrindo arquivos

```
FILE *fptr; /* ponteiro para arquivo */  
fptr = fopen("arqtext.txt", "wb+");
```

Tipo de abertura

- "rb+" Abrir arquivo binário para leitura e gravação. O arquivo deve existir e pode ser **atualizado**.
- "wb+" Abrir arquivo binário para leitura e gravação. Se o arquivo existir ele será **destruído e reinicializado**. Se não existir, será criado.
- "ab+" Abrir um arquivo binário para atualização e para adicionar dados no fim do arquivo existente, ou cria um novo

Observações sobre Abertura de Arquivos em C

- ◆ SO mantém um “cursor” que indica a posição de trabalho no arquivo
- ◆ Se não for possível a abertura, a função **fopen** retorna NULL

Fechamento de Arquivos

- ◆ Após leitura/escrita do arquivo, devemos fechá-lo
- ◆ Função de fechamento

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

- Retorna 0 se o arquivo foi fechado com sucesso

Leitura (Modo Binário)

```
int fread(void* p,int tam,int nelem,FILE* fp) ;
```

- **p** é o endereço de memória em que vai ser armazenado o que for lido
- **tam** é o tamanho em bytes de cada elemento lido
- **nelem** é o número de elementos de tamanho **tam** lidos
- Retorna a quantidade de bytes lidos com sucesso (**tam** * **nelem**)

Escrita (Modo Binário)

```
int fwrite(void* p,int tam,int nelem,FILE* fp) ;
```

- `p` é o endereço de memória cujo conteúdo deseja-se salvar em arquivo
- `tam` é o tamanho em bytes de cada elemento escrito
- `nelem` é o número de elementos de tamanho `tam` escritos
- Retorna a quantidade de bytes escritos com sucesso (`tam * nelem`)

Verificando o Final do Arquivo

- ◆ Em operações de leitura do arquivo, é comum verificarmos se o final do arquivo já foi atingido
- ◆ Função de verificação de fim de arquivo

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

- Retorna 1 se o fim do arquivo foi atingido
- Retorna 0 caso contrário

Usando fwrite na Escrita

```
#include <stdio.h>
typedef struct ponto {
    float x,y;
} Ponto;
```

Programa que salva n pontos em
um arquivo binário

```
int main () {
    int i,n;
    Ponto p;
    FILE* fp = fopen("arquivo", "wb");
    if (fp == NULL) {
        printf("Erro na abertura do arquivo.\n");
        exit(1);
    }
    printf("Digite numero de pontos a gravar\n");
    scanf("%d",&n);
    for (i = 0; i < n; i++) {
        scanf("%d %d",&p.x,&p.y);
        fwrite(&p, sizeof(Ponto), 1, fp);
    }
    fclose(fp);
}
```

Gravando cada ponto entrado
pelo usuario usando fwrite

fwrite(&p, sizeof(Ponto), 1, fp);

Usando fread na Leitura

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

```
int main () {  
    int i,n;  
    Ponto p;  
    FILE* fp = fopen("arquivo", "rb");  
    if (fp == NULL) {  
        printf("Erro na abertura do arquivo.\n");  
        exit(1);  
    }  
    while(!feof(fp)) {  
        fread(&p, sizeof(Ponto), 1, fp);  
        printf("Ponto lido: (%d,%d)", p.x, p.y);  
    }  
    fclose(fp);  
}
```

Programa que lê todos os pontos armazenados em um arquivo binário

Testa se o fim do arquivo já foi atingido

Lê cada ponto e guarda na variável p

Leitura/Escrita de Blocos de Dados

- ◆ As funções fread/fwrite permitem ler/escrever grandes blocos de dados em um arquivo
 - Um dos parâmetros indica qual é a quantidade de dados de um determinado tipo a ser lido/escrito
- ◆ Portanto podem ser úteis para ler/escrever estruturas ou vetores em um arquivo numa única chamada de função

Usando fwrite na Escrita

```
#include <stdio.h>
typedef struct ponto {
    float x,y;
} Ponto;
```

Número de pontos do
vetor

```
void salva (char* arquivo, int n, Ponto* vet) {
    FILE* fp = fopen(arquivo, "wb");
    if (fp == NULL) {
        printf("Erro na abertura do arquivo.\n");
        exit(1);
    }
    fwrite(vet, sizeof(Ponto), n, fp);
    fclose(fp);
}
```

Função que salva um vetor de pontos em um
arquivo binário

Usando fread na Leitura

```
void carrega (char* arquivo, int n, Ponto* vet) {  
    FILE* fp = fopen (arquivo, "rb");  
    if (fp == NULL) {  
        printf("Erro na abertura do arquivo.\n");  
        exit(1);  
    }  
    fread (vet, sizeof(Ponto), n, fp);  
    fclose(fp);  
}
```

Função que recupera um vetor de pontos de um arquivo binário

Usando as Funções Definidas Anteriormente

```
int main() {
    Ponto *entrada, *saida; int nPontos, cont,pos ;
    FILE *arquivo;
    char nome_arquivo[121];
    printf("Digite o nome do arquivo:\n");
    scanf("%120s",nome_arquivo);
    printf("\nDigite o número de pontos:\n");
    scanf("%d",&nPontos);
    entrada = (Ponto *) malloc(nPontos*sizeof(Ponto));
    for (cont = 0; cont < nPontos;cont++) {
        printf("Digite coordenadas x,y:\n");
        scanf("%f%f",&entrada[cont].x,&entrada[cont].y);
    }
    /* continua */
```

Programa que salva e recupera um vetor de pontos em um arquivo binário

Usando as Funções Definidas Anteriormente

```
salva(nome_arquivo, nPontos, entrada);  
do {  
    printf("Digite agora a posição do ponto que  
        deseja ver: \n");  
    scanf("%d",&pos);  
} while (pos > nPontos || pos<=0 );  
saida = (Ponto *) malloc (nPontos*sizeof(Ponto));  
carrega(nome_arquivo, nPontos, saida);  
printf("O ponto na posicao %d é  {%f,%f}\n", pos,  
    saida[pos-1].x, saida[pos-1].y);  
}
```

Gravando os pontos no
arquivo

Lendo os pontos do
arquivo