Algoritmos e Programação de Computadores

Instituto de Computação

UNICAMP

Segundo Semestre de 2013

Roteiro

Motivação

- Vimos que há dois tipos de arquivos: textos e binários.
- Variáveis `int` ou `float` ocupam tamanho fixo na memória. Por exemplo, um `int` ocupa 4 bytes e `um double`, 8 bytes.
- Representação em texto requer um número variável de dígitos (por exemplo, `10`, `5.67`, `100.340876`).
  - Lembre-se que cada letra/dígito é um `char` e ocupa 1 byte de memória.
- Armazenar dados em arquivos de forma análoga à utilizada em memória permite:
  - Reduzir o tamanho do arquivo.
  - Guardar estruturas complexas tendo acesso simples.

Arquivos binários em C

- Assim como em arquivos textos, devemos criar um ponteiro especial: um ponteiro para arquivos.

  ```
  FILE *nome_variavel;
  ```

- Podemos então associá-lo com um arquivo real do computador usando o comando `fopen()`.

  ```
  FILE *arq1;
  arq1 = fopen("teste.bin", "rb");
  ```

Modos de abertura de arquivos binários

Um pouco mais sobre a função `fopen()` para arquivos binários:

```
FILE *fopen(const char *caminho, char *modo);
```

Modos de abertura de arquivos binários

modo	operações	indicador de posição
`rb`	leitura	início do arquivo
`wb`	escrita	início do arquivo
`r+b`	leitura e escrita	início do arquivo
`w+b`	escrita e leitura	início do arquivo

Modos de abertura de arquivos binários

- Ao se tentar abrir um arquivo inexistente para leitura (`rb`) ou leitura e escrita (`r+b`), `fopen` retorna `NULL`. Caso o arquivo exista, o arquivo é aberto e seu conteúdo é preservado.
- Ao se tentar abrir um arquivo inexistente para escrita (`wb`) ou escrita e leitura (`w+b`), um novo arquivo é criado e então aberto pelo `fopen`. Caso o arquivo exista, seu conteúdo é primeiramente removido e então aberto para escrita.

Funções para leitura e escrita de arquivos binários

- As funções `fread` e `fwrite` permitem a leitura e escrita de blocos de dados, respectivamente.
- Devemos determinar o número de elementos a serem lidos ou gravados e o tamanho de cada um.

Funções para leitura e escrita de arquivos binários

Para ler dados de um arquivo binário, usamos a função `fread`.

```
int fread(void *pt-mem, int size,
          int num-items, FILE *pt-arq);
```

- `pt-mem`: ponteiro para região da memória (já alocada) para onde os dados serão lidos.
- `size`: número de bytes de um item a ser lido.
- `num-items`: número de itens a serem lidos.
- `pt-arq`: ponteiro para o arquivo.
- A função `fread` retorna o número de itens corretamente lidos.

Funções para leitura e escrita de arquivos binários

Podemos, por exemplo, ler um `double` de um arquivo em formato binário como:

```
FILE *arq;
double aux;

arq = fopen("teste.bin", "w+b");
fread(&aux, sizeof(double), 1, arq);
fclose(arq);
```

Funções para leitura e escrita de arquivos binários

Podemos, por exemplo, ler um vetor de doubles de um arquivo em formato binário como:

```
FILE *arq;
double aux[3];

arq = fopen("teste.bin", "w+b");
fread(aux, sizeof(double), 3, arq);
fclose(arq);
```

Funções para leitura e escrita de arquivos binários

Para escrever em um arquivo binário, usamos a função `fwrite`.

```
int fwrite(void *pt-mem, int size,
           int num-items, FILE *pt-arq);
```

- `pt-mem`: ponteiro para região da memória contendo os itens que devem ser gravados.
- `size`: número de bytes de um item.
- `num-items`: número de itens a serem gravados.
- `pt-arq`: ponteiro para o arquivo.
- A função `fwrite` retorna o número de itens corretamente escritos.

Funções para leitura e escrita de arquivos binários

Podemos, por exemplo, gravar um `double` em um arquivo em formato binário como:

```
FILE *arq;
double aux = 2.5;

arq = fopen("teste.bin", "w+b");
fwrite(&aux, sizeof(double), 1, arq);
fclose(arq);
```

Funções para leitura e escrita de arquivos binários

Podemos, por exemplo, gravar um vetor de doubles em um arquivo em formato binário como:

```
FILE *arq;
double aux[] = {2.5, 1.4, 3.6};

arq = fopen("teste.bin", "w+b");
fwrite(aux, sizeof(double), 3, arq);
fclose(arq);
```

Funções para leitura e escrita de arquivos binários

Acessando um valor double num arquivo em formato binário:

```
#include <stdio.h>

int main() {
  FILE *arq;
  double aux1 = 2.5, aux2 = 0;

  arq = fopen("teste.bin", "w+b");
  if (arq != NULL) {
    fwrite(&aux1, sizeof(double), 1, arq);

    rewind(arq); /* volta para o inicio do arquivo */

    fread(&aux2, sizeof(double), 1, arq);
    printf("Conteudo de aux2: %f\n", aux2);
    fclose(arq);
  }

  return 0;
}
```

Funções para leitura e escrita de arquivos binários

Acessando múltiplos valores doubles num arquivo em formato binário:

```
#include <stdio.h>

int main() {
  FILE *arq;
  double aux1[] = {2.5, 1.4, 3.6}, aux2[3];
  int i;

  arq = fopen("teste.bin", "w+b");
  if (arq != NULL) {
    fwrite(aux1, sizeof(double), 3, arq);

    rewind(arq); /* volta para o inicio do arquivo */

    fread(aux2, sizeof(double), 3, arq);
    for (i = 0; i < 3; i++)
      printf("Conteudo de aux2[%d]: %f\n", i, aux2[i]);
    fclose(arq);
  }

  return 0;
}
```

Funções para leitura e escrita de arquivos binários

- Lembre que o indicador de posição de um arquivo, assim que o arquivo é aberto, aponta para o início do arquivo (a menos que ele seja aberto como *append*).
- Quando lemos uma determinada quantidade de itens, o indicador de posição automaticamente avança para o próximo item não lido.
- Quando escrevemos algum item, o indicador de posição automaticamente avança para a posição seguinte ao item escrito.

Funções para leitura e escrita de arquivos binários

- Se na leitura não soubermos exatamente quantos itens estão gravados, podemos verificar o retorno da função `fread`:
  - Esta função retorna o número de itens corretamente lidos.
  - Se alcançarmos o final do arquivo e tentarmos ler algo mais, ela retornará o valor 0.

Sendo assim, podemos ler todos os dados de um arquivo binários (supondo que todos os valores armazenados são do tipo `double`), como segue:

```
for (i = 0; fread(&vetor[i], sizeof(double), 1, arq);  i++);
```

... ou de forma equivalente:

```
i = 0;
while (fread(&vetor[i], sizeof(double), 1, arq))
  i++;
```

Funções para leitura e escrita de arquivos binários

```
#include <stdio.h>

int main() {
  FILE *arq;
  double aux1[] = {2.5, 1.4, 3.6}, aux2[3];
  int n, i;

  arq = fopen("teste.bin", "w+b");
  if (arq != NULL) {
    fwrite(aux1, sizeof(double), 3, arq);

    rewind(arq); /* volta para o inicio do arquivo */

    for (n = 0; fread(&aux2[n], sizeof(double), 1, arq); n++);

    for (i = 0; i < n; i++)
      printf("Conteudo de aux2[%d]: %f\n", i, aux2[i]);
    fclose(arq);
  }

  return 0;
}
```

Conversão de arquivos

- Suponha que desejamos converter um arquivo texto, formado apenas por números inteiros (devidamente espaçados entre si), em um arquivo binário (e vice-versa).
- Programas com estas funcinalidades podem ser úteis para testar programas que usam arquivos binários como entrada ou saída.
- Podemos, por exemplo, gerar uma instância de teste em formato texto (usando um editor de texto qualquer) e depois convertê-la para o formato binário.
- Da mesma forma, podemos converter um arquivo binário em texto, para verificar o resultado produzido por um programa.

Convertendo um arquivo texto para binário

```
#include <stdio.h>

int main() {
  FILE *txt, *bin;
  int x;

  txt = fopen("teste.txt", "r");
  bin = fopen("teste.bin", "wb");

  if ((txt != NULL) && (bin != NULL))
    while (fscanf(txt, "%d", &x) != EOF)
      fwrite(&x, sizeof(int), 1, bin);

  if (txt)
    fclose(txt);
  if (bin)
    fclose(bin);

  return 0;
}
```

Convertendo um arquivo binário para texto

```
#include <stdio.h>

int main() {
  FILE *bin, *txt;
  int x;

  bin = fopen("teste.bin", "rb");
  txt = fopen("teste.txt", "w");

  if ((txt != NULL) && (bin != NULL))
    while (fread(&x, sizeof(int), 1, bin))
      fprintf(txt, "%d\n", x);

  if (bin)
    fclose(bin);
  if (txt)
    fclose(txt);

  return 0;
}
```

Acesso não sequencial

- Podemos fazer um acesso não sequencial num arquivo usando a função `fseek`.
- Esta função altera a posição de leitura/escrita no arquivo.
- O deslocamento pode ser relativo ao:
  - início do arquivo (`SEEK_SET`)
  - ponto atual (`SEEK_CUR`)
  - final do arquivo (`SEEK_END`)
- A função `fseek` pode ser usada tanto com arquivos binários quanto com arquivos textos, no entanto, é muito mais facilmente utilizada com arquivos binários.

Acesso não sequencial

```
int fseek(FILE *pt-arq, long num-bytes, int origem);
```

- `pt-arq:` ponteiro para arquivo.
- `num-bytes:` quantidade de bytes para se deslocar.
- `origem:` posição de início do deslocamento (`SEEK_SET`, `SEEK_CUR`, `SEEK_END`).
- A função `fseek` retorna 0 se foi bem sucedida, ou um valor não nulo, caso falhe.

Acesso não sequencial

```
#include <stdio.h>

int main() {
  FILE *arq;
  int vetor[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, x;

  arq = fopen("teste.bin", "w+b");

  if (arq != NULL) {
    fwrite(vetor, sizeof(int), 10, arq);

    fseek(arq, 2 * sizeof(int), SEEK_SET);
    fread(&x, sizeof(int), 1, arq);
    printf("X1 = %d\n", x);

    fseek(arq, 4 * sizeof(int), SEEK_CUR);
    fread(&x, sizeof(int), 1, arq);
    printf("X2 = %d\n", x);

    ...
```

Acesso não sequencial

```
    ...

    fseek(arq, -3 * sizeof(int), SEEK_CUR);
    fread(&x, sizeof(int), 1, arq);
    printf("X3 = %d\n", x);

    fseek(arq, -1 * sizeof(int), SEEK_END);
    fread(&x, sizeof(int), 1, arq);
    printf("X4 = %d\n", x);

    fseek(arq, -4 * sizeof(int), SEEK_END);
    fread(&x, sizeof(int), 1, arq);
    printf("X5 = %d\n", x);

    fclose(arq);
  }

  return 0;
}
```

Acesso não sequencial

- O que o programa anterior imprime ao ser executado?

```
X1 = 3
X2 = 8
X3 = 6
X4 = 10
X5 = 7
```

Registros

- Um arquivo pode armazenar registros (como um cadastro).
- 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`.
- Considere uma aplicação para um cadastro de alunos onde:
  - Cada aluno tem um nome e um registro acadêmico (RA).
  - Deseja-se implementar funções para imprimir o cadastro completo de alunos e para alterar o nome de um aluno, dado o seu RA.

Exemplo: declarações básicas

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

#define TAM 5 /* tamanho do vetor usado como cadastro */

struct Aluno {
  char nome[100];
  int RA;
};
typedef struct Aluno Aluno;

/* Funcao que imprime todo o conteudo do cadastro */
void imprimeCadastro(char nomeArq[]);

/* Funcao que altera o nome de uma pessoa, dado seu RA */
void alteraNome(int ra, char nome[], char nomeArq[]);
```

Exemplo: função principal

```
int main() {
  char nomeArq[] = "alunos.bin"; /* nome do arquivo do cadastro de alunos */
  Aluno cadastro[TAM] = { {"Joao da Silva", 111111},
    {"Jose Souza", 121212}, {"Luis Santos", 131313},
    {"Maria Pereira", 141414}, {"Ana Alves", 151515} };

  /* abre o arquivo para escrita */
  FILE *arq = fopen(nomeArq, "w+b");

  if (arq == NULL) {
    printf("Erro ao abrir o arquivo de cadastro de alunos.\n");
    return 0;
  }

  fwrite(cadastro, sizeof(Aluno), TAM, arq);
  fclose(arq);

  imprimeCadastro(nomeArq);
  alteraNome(131313, "Luisa Saints", nomeArq);
  imprimeCadastro(nomeArq);

  return 0;
}
```

Exemplo: função que imprime arquivo

```
void imprimeCadastro(char nomeArq[]) {
  Aluno aluno;

  /* abre o arquivo para leitura */
  FILE *arq = fopen(nomeArq, "rb");

  if (arq == NULL) {
    printf("Erro ao abrir o arquivo de cadastro de alunos.\n");
    return;
  }

  printf("----- Imprimindo Cadastro -----\n");

  while (fread(&aluno, sizeof(Aluno), 1, arq))
    printf("%06d %s\n", aluno.RA, aluno.nome);

  printf("\n");

  fclose(arq);
}
```

Exemplo: função que altera o nome de um aluno

```
void alteraNome(int ra, char nome[], char nomeArq[]) {
  Aluno aluno;
  int OK = 0;

  /* abre o arquivo para leitura e escrita */
  FILE *arq = fopen(nomeArq, "r+b");

  if (arq == NULL) {
    printf("Erro ao abrir o arquivo de cadastro de alunos.\n");
    return;
  }

  while ((OK == 0) && fread(&aluno, sizeof(Aluno), 1, arq))
    if (aluno.RA == ra) {
      strcpy(aluno.nome, nome); /* altera o nome no registro */
      fseek(arq, -1 * sizeof(Aluno), SEEK_CUR); /* volta uma posicao */
      fwrite(&aluno, sizeof(Aluno), 1, arq); /* sobrescreve o registro */
      OK = 1;
    }

  fclose(arq);
}
```

Exemplo: saída do programa

```
----- Imprimindo Cadastro -----
111111 Joao da Silva
121212 Jose Souza
131313 Luis Santos
141414 Maria Pereira
151515 Ana Alves

----- Imprimindo Cadastro -----
111111 Joao da Silva
121212 Jose Souza
131313 Luisa Saints
141414 Maria Pereira
151515 Ana Alves
```

Exercícios

Intercalação

Escreva um programa que leia dois arquivos binários contendo números inteiros e ordenados, e escreva um único arquivo binário com os números ordenados de ambos os arquivos.

Ordenação

Escreva um programa que leia uma série de números inteiros de um arquivo binário e escreva um arquivo binário contendo estes números ordenados.

Importante: em ambos os casos, seu programa não deve usar um vetor auxiliar para armazenar os números.