Algoritmos e Programação de Computadores

Instituto de Computação

UNICAMP

Primeiro Semestre de 2013

Roteiro

Motivação

- Vimos que há dois tipos de arquivos: textos e binários.
- Variáveis `int` ou `float` têm tamanho fixo na memória. Por exemplo, um `int` ocupa 4 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
`rb`	leitura
`wb`	escrita
`r+b`	leitura e escrita
`w+b`	escrita e leitura

Modos de abertura de arquivos binários

- Se um arquivo for aberto para leitura (`rb`) e ele não existir, a função retorna `NULL`.
- Se um arquivo for aberto para escrita (`wb`) e ele não existir, um novo arquivo é criado. Se o arquivo existir, seu conteúdo é primeiramente removido.
- Se um arquivo for aberto para leitura e escrita (`r+b`) e ele existir, então seu conteúdo não é removido. Se o arquivo não existir, a função retorna `NULL`.
- Se um arquivo for aberto para escrita e leitura (`w+b`) e ele existir, então seu conteúdo é primeiramente removido. Se o arquivo não existir, um novo arquivo é criado.

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 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 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 `double` 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

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

Usando o exemplo anterior, podemos ler um `double` em formato binário como segue:

```
#include <stdio.h>

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

  arq = fopen("teste.bin", "w+b");
  if (arq != NULL) {
    fwrite(&aux1, sizeof(double), 1, arq);
    rewind(arq);
    fread(&aux2, sizeof(double), 1, arq);
    printf("Conteudo de aux2: %lf\n", aux2);
    fclose(arq);
  }

  return 0;
}
```

Funções para leitura e escrita de arquivos binários

Usando o exemplo visto anteriormente, podemos ler um vetor de `double` em formato binário como segue:

```
#include <stdio.h>

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

  arq = fopen("teste.bin", "w+b");
  if (arq != NULL) {
    fwrite(aux1, sizeof(double), 3, arq);
    rewind(arq);
    fread(aux2, sizeof(double), 3, arq);
    for (i = 0; i < 3; i++)
      printf("Conteudo de aux2[%d]: %lf\n", i, aux2[i]);
    fclose(arq);
  }

  return 0;
}
```

Funções para leitura e escrita de arquivos binários

- Lembre que indicador de posição de um arquivo, assim que um arquivo é aberto, é apontado para o início do arquivo.
- 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, poderemos usar o que é retornado pela função `fread`:
  - Esta função retorna o número de itens corretamente lidos.
  - Se alcançarmos o final do arquivo e tentarmos ler algo, ela retorna 0.

No exemplo do vetor, poderíamos ter lido os dados como segue:

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

... ou de forma equivalente:

```
i = 0;
while (fread(&aux2[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};
  double aux2[3];
  int n, i;

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

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

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

  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.

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.

Assim, se quisermos alterar o terceiro `double` de um arquivo:

```
double aux1[] = {2.5, 1.4, 3.6}, aux2 = 5.0;
FILE *arq = fopen("teste.bin", "w+b");
fwrite(aux1, sizeof(double), 3, arq);

fseek(arq, 2 * sizeof(double), SEEK_SET);
fwrite(&aux2, sizeof(double), 1, arq);
fclose(arq);
```

```c
#include <stdio.h>

int main() {
  FILE *arq;
  double aux1[] = {2.5, 1.4, 3.6}, aux2[3], aux3 = 5.0;
  int i;

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

    fseek(arq, 2 * sizeof(double), SEEK_SET);
    fwrite(&aux3, sizeof(double), 1, arq);

    fseek(arq, 0, SEEK_SET); /* equivalente a rewind(arq) */
    fread(aux2, sizeof(double), 3, arq);

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

  return 0;
}
```

Registros

- 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`.

Exemplo com registros

Vamos considerar uma aplicação para um cadastro de alunos:

```
#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 em arquivo */
void imprimeArquivo(char nomeArq[]);

/* Funcao 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", 100001},
    {"Jose Souza", 100002}, {"Luis Santos", 100003},
    {"Maria Pereira", 100004}, {"Ana Alves", 100005} };

  /* 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);

  imprimeArquivo(nomeArq);
  alteraNome(100003, "Luisa Saints", nomeArq);
  imprimeArquivo(nomeArq);

  return 0;
}
```

Exemplo: função que imprime arquivo

```
void imprimeArquivo(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("Nome: %s, RA: %d\n", aluno.nome, aluno.RA);

  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);
}
```

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.