ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES

Disciplina: 113476

Profa. Carla Denise Castanho

Universidade de Brasília – UnB
Instituto de Ciências Exatas – IE
Departamento de Ciência da Computação – CIC

9. ARQUIVOS TEXTO

Arquivos

- Até o momento, todos os nossos algoritmos trabalharam com dados na **memória**.
- Como você já sabe, a memória do computador é **volátil**, quando o programa se encerra, os dados são perdidos.
- Como fazemos para guardar dados de maneira **permanente**, ou seja, entre várias execuções do mesmo programa?
- Para isso temos a noção de **arquivo**.

Arquivos

- Um arquivo é uma abstração do sistema operacional, é a maneira como ele organiza dados em memória **não volátil**.
- A forma de armazenamento do arquivo depende do tipo de dispositivo que é usado – pen drive, disco rígido, CD, etc – no entanto, para o programador um arquivo é simplesmente uma **sequência de bytes** (*stream*) onde são lidos ou gravados dados.
- Uma vez que um arquivo é uma sequência de bytes, um valor especial chamado de **EOF** (*End of File*) é utilizado para marcar o final de um arquivo.
- Cada arquivo está associado a um **nome**, pelo qual é conhecido externamente, isto é, o nome que consta no sistema operacional.

Arquivos

- A maioria dos SOs organiza os arquivos em dois tipos:
  - **TEXTO**: os bytes do arquivo representam **caracteres**, isto é, formam um texto. Um programa C, por exemplo, é um arquivo desse tipo, pode ser facilmente visualizado com qualquer editor de texto.
  - **BINÁRIO**: são gravados os dados **como estariam na memória**, byte a byte. Por exemplo, uma variável inteira é gravada com 4 bytes com o conteúdo exato que está na memória. Não conseguimos visualizar com um editor de texto, é necessário um programa que reconheça aquela sequência de bytes e dê significado a ela.

Arquivos

- A maioria dos SOs organiza os arquivos em dois tipos:
  - **TEXTO**: os bytes do arquivo representam **caracteres**, isto é, formam um texto. Um arquivo desse tipo pode ser lido por qualquer editor de texto.
  - **BINÁRIO**: são gravados byte a byte da **memória**, byte a byte. Uma variável inteira é gravada com 4 bytes com o conteúdo exato que está na memória. Não conseguimos visualizar com um editor de texto, é necessário um programa que reconheça aquela sequência de bytes e dê significado a ela.

Neste capítulo, veremos como trabalhar com arquivos do tipo texto.
mais adiante, no final do semestre, trabalharemos com manipulação de arquivos do tipo binário.

Trabalhando com Arquivos

- Para trabalhar com arquivos, a Linguagem C cria uma camada de abstração, um **ponteiro de arquivo**, que é usado para realizar todas as operações.
- Esse ponteiro guarda as informações sobre o uso daquele arquivo, incluindo a **posição corrente**, isto é, qual o próximo byte a ser lido (ou gravado). Toda vez que uma operação de leitura ou escrita é realizada, essa posição é automaticamente atualizada.
- Um ponteiro de arquivo é declarado da seguinte maneira:

```
FILE *fp;
```

- Como acontece com toda variável, pode-se usar qualquer nome para o ponteiro, no entanto, `fp` é o nome mais convencional.

Trabalhando com Arquivos

- Antes de ler ou gravar em algum arquivo precisamos ter certeza de algumas coisas:
  - Onde se encontra o arquivo?
  - O arquivo já existe, e vamos apenas abrí-lo?
  - O arquivo não existe e vamos criá-lo?
  - O arquivo já existe mas vamos recriá-lo zerado?
- Para isso, temos a função **fopen** (*file open*), que possui dois parâmetros, nessa ordem: o **nome** do arquivo e o **tipo de operação** que faremos com ele...

Trabalhando com Arquivos

- Tabela de Operações:
  - r (*read*) - abre um arquivo TEXTO para leitura, ele já deve existir;
  - w (*write*) - abre (ou cria) um arquivo TEXTO para gravação, se já existir, elimina seu conteúdo e recomeça a gravação a partir do seu início;
  - a (*append*) - abre (ou cria) um arquivo TEXTO para gravação, e sempre grava a partir de seu final;
  - r+ - abre um arquivo TEXTO para leitura e gravação; o arquivo deve existir e poder ser modificado;
  - w+ - abre (ou cria) um arquivo TEXTO para leitura e gravação, se o arquivo já existir, o conteúdo anterior será destruído;
  - a+ - abre (ou cria) um arquivo TEXTO para leitura e gravação, as escritas serão realizadas no fim do arquivo.

Exemplo da função *fopen*

```
FILE *fp;
char nomeArquivo[] = "arquivo.txt";
fp = fopen(nomeArquivo, "w");
```

Trabalhando com Arquivos

- Caso *fopen* não consiga abrir o arquivo – porque ele deveria existir, mas não existe, ou não tem permissão suficiente para abri-lo/cria-lo – ela retornará o ponteiro nulo (**NULL**).
- Dessa forma, podemos testar se o arquivo já existe antes de criar um novo, o que recriaria (apagaria os dados) do arquivo existente:

Exemplo - Testando se o arquivo existe, antes de criar

```
FILE *fp;
char nomeArquivo[] = "arquivo.txt";
fp = fopen(nomeArquivo, "r+");

/* caso o arquivo nao exista, cria um novo */
if (fp == NULL) {
    fp = fopen(nomeArquivo, "w");
}
```

Trabalhando com Arquivos

- Ao abrirmos um arquivo com *fopen*, estamos alocando os recursos necessários para trabalhar com ele. É necessário **fechar** o arquivo quando não formos mais utilizá-lo, liberando os recursos:

```
fclose(fp); /* fp eh o ponteiro do arquivo */
```

- Ao fecharmos o arquivo, também garantimos que todos os dados serão efetivamente salvos, e não ficarão simplesmente no buffer -- região temporária de memória onde o SO deixa os dados antes de serem escritos em disco.
- Se quisermos simplesmente escrever os dados em disco, sem fechar o arquivo, fazemos:

```
fflush(fp); /* fp eh o ponteiro do arquivo */
```

Trabalhando com Arquivos

- Se quisermos reposicionar o ponteiro para o início do arquivo, utilizamos:

```
rewind(fp);
```

Arquivos Texto

- Para fazermos a gravação e leitura em um arquivo **texto**, devemos utilizar as funções **fprintf** e **fscanf**, respectivamente.
- Elas funcionam exatamente da mesma maneira que *printf* e *scanf*, no entanto, recebem como primeiro parâmetro o ponteiro de arquivo.

Exemplo - Leitura e escrita em arquivo texto

```
int numero1, numero2, numero3;
FILE *fp;
char nomeArquivo[] = "arquivo.txt";
fp = fopen(nomeArquivo, "w+"); /* abre (ou cria) para leitura e gravacao */

/* recebe dados e escreve no arquivo texto... */
printf("Digite 3 numeros inteiros:");
scanf("%d %d %d", &numero1, &numero2, &numero3);
fprintf(fp, "%d %d %d", numero1, numero2, numero3);

/* le dados do arquivo texto... */
rewind(fp); /* volta o ponteiro para o inicio */
fscanf(fp, "%d %d %d", &numero1, &numero2, &numero3);
printf("Os numero lidos foram: %d %d %d\n", numero1, numero2, numero3);

/* fecha o arquivo (salvando os dados) */
fclose(fp);
```

Arquivos Texto

Um exemplo completo

```c
#include <stdio.h>

int main () {
    FILE *fp;
    char string[50] = "Meu primeiro programa com arquivo  texto";
    char string_lida[50], nomeArquivo[30] = "arquivo.txt";

    fp = fopen(nomeArquivo, "r+");
    if (fp == NULL)
        fp = fopen(nomeArquivo, "w");

    fprintf(fp, "%s", string);
    fclose(fp);

    fp = fopen(nomeArquivo, "r+");
    fscanf(fp, "%s", string_lida);
    printf("string do arquivo e: %s\n", string_lida);
    fclose(fp);

    return 0;
}
```

Arquivos Texto

Um exemplo completo

```
#include <stdio.h>

int main () {
    FILE *fp;
    char string[50] = "Meu primeiro programa com arquivo  texto";
    char string_lida[50], nomeArquivo[30] = "arquivo.txt";

    fp = fopen(nomeArquivo, "r+");
    if (fp == NULL)
        fp = fopen(nomeArquivo, "w");

    fprintf(fp, "%s", string);
    fclose(fp);

    fp = fopen(nomeArquivo, "r+");
    fscanf(fp, "%s", string_lida);
    printf("string do arquivo e: %s\n", string_lida);
    fclose(fp);

    return 0;
```

Tenta abrir o arquivo para leitura e gravação

Testa se o arquivo existe

Caso não exista, cria e abre p/ gravação

Grava no arquivo

Fecha o arquivo

Lê do arquivo

Fecha o arquivo

Arquivos Texto

Um exemplo completo

```c
#include <stdio.h>

int main () {
    FILE *fp;
    char string[50] = "Meu primeiro prograr
    char string_lida[50], nomeArquivo[30] =

    fp = fopen(nomeArquivo, "r+");
    if (fp == NULL)
        fp = fopen(nomeArquivo, "w");

    fprintf(fp, "%s", string);
    fclose(fp);

    fp = fopen(nomeArquivo, "r+");
    fscanf(fp, "%s", string_lida);
    printf("string do arquivo e: %s\n", string_lida);
    fclose(fp);

    return 0;
}
```

Uma vez que fechamos o arquivo e o reabrimos, não há necessidade de usarmos rewind para o ponteiro voltar à posição inicial, pois quando abrimos o arquivo ele já está lá.

Arquivos Texto

- Note que a saída do programa será simplesmente a palavra "*Meu*", pois, da mesma maneira que *scanf*, a função *fscanf* para de ler ao encontrar um espaço.
- Para solucionar isso, temos duas alternativas:
  - fazer um *loop* para ler as *strings* do arquivo enquanto não chegar no final do mesmo;
  - ou utilizar o comando fscanf da seguinte forma:

```
fscanf(fp, "%[^\n]s", string);
```

- Veja os exemplos nos próximos slides.

Arquivos Texto

Um exemplo completo

```c
#include <stdio.h>

int main () {
    FILE *fp;
    char string[50] = "Meu primeiro programa com arquivo  texto";
    char string_lida[50], nomeArquivo[30] = "arquivo.txt";

    fp = fopen(nomeArquivo, "r+");
    if (fp == NULL)
        fp = fopen(nomeArquivo, "w");
    fprintf(fp, "%s", string);
    fclose(fp);
    fp = fopen(nomeArquivo, "r+");
    while (fscanf(fp, "%s", string_lida) > 0) {
        printf("%s ", string_lida);
        getchar();
    }
    fclose(fp);

    return 0;
}
```

Arquivos Texto

Um exemplo completo

```
#include <stdio.h>

int main () {
    FILE *fp;
    char string[50] = "Meu primeiro programa com arquivo  texto";
    char string_lida[50], nomeArquivo

    fp = fopen(nomeArquivo, "r+");
    if (fp == NULL)
        fp = fopen(nomeArquivo, "w");
    fprintf(fp, "%s", string);
    fclose(fp);
    fp = fopen(nomeArquivo, "r+");
    while (fscanf(fp, "%s", string_lida) > 0) {
        printf("%s ", string_lida);
        getchar();
    }
    fclose(fp);

    return 0;
}
```

Note que cada vez que chamamos a funcao *fscanf* ela incrementa automaticamente o ponteiro do arquivo.

Arquivos Texto

Um exemplo completo

```
#include <stdio.h>

int main () {
    FILE *fp;
    char string[50] = "Meu primeiro programa com arquivo  texto";
    char string_lida[50], nomeArquivo[30] = "arquivo.txt";

    fp = fopen(nomeArquivo, "r+");
    if (fp == NULL)
        fp = fopen(nomeArquivo, "w");
    fprintf(fp, "%s", string);
    fclose(fp);
    fp = fopen(nomeArquivo, "r+");
    fscanf(fp, "%[^\n]s", string_lida);
    printf("%s ", string_lida);
    fclose(fp);

    return 0;
}
```

Arquivos Texto

Um exemplo completo

```
#include <stdio.h>

int main () {
    FILE *fp;
    char string[50] = "Meu primeiro programa com arquivo  texto";
    char string_lida[50], nomeArquivo[30] = "arquivo.txt";

    fp = fopen(nomeArquivo, "r+");
    if (fp == NULL)
        fp = fopen(nomeArquivo, "w");
    fprintf(fp, "%s", string);
    fclose(fp);
    fp = fopen(nomeArquivo, "r+");
    fscanf(fp, "%[^\n]s", string_lida);
    printf("%s ", string_lida);
    fclose(fp);

    return 0;
}
```

Lê toda a string de uma vez.