Arquivos

Introdução

- As estruturas vistas anteriormente armazenam as informações na memória principal do computador.
  - Nem sempre é conveniente.
- Problemas:
  - A informação é perdida;
  - As estruturas de dados são limitadas;
    - Existe uma **quantidade** de informação que pode ser armazenada para resolver o problema.

Introdução

- Algumas informações geradas pelos programas precisam ser mantidas para posterior uso
  - Exs: cadastro de alunos, cadastro de vendas de uma empresa, estatísticas ...

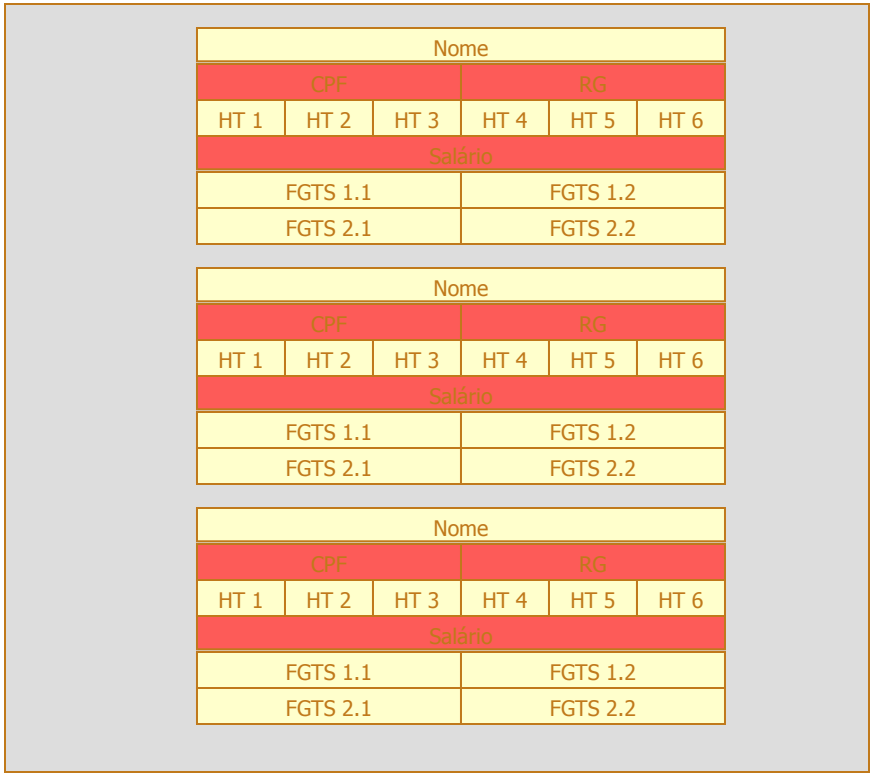
Introdução

- Um arquivo é armazenado em um dispositivo de memória secundária. Pode ser lido e escrito por um programa.

- Em programação, existem vários tipos de informações que podem ser armazenadas em arquivo

Introdução

Arquivo de Registros



Representação Simbólica

Introdução

Vetores / Matrizes

- Armazenam pequena quantidade de informação;
- Ao término do programa, todas as informações armazenadas são perdidas;
- Restrito as definições da linguagem de programação;
- Tamanho da variável deve ser definida antes do uso;

Arquivo

- Armazenam grandes quantidades de informações;
- Mantém informações armazenadas em disco;
- Variável do tipo *file* é um tipo de dado que existe independentemente de qualquer programa e pode ser acessado por outros programas;
- Depende do espaço disponível no disco;

- Um **arquivo** é uma sequência de *bytes* que reside em um dispositivo de armazenamento durável.

Introdução

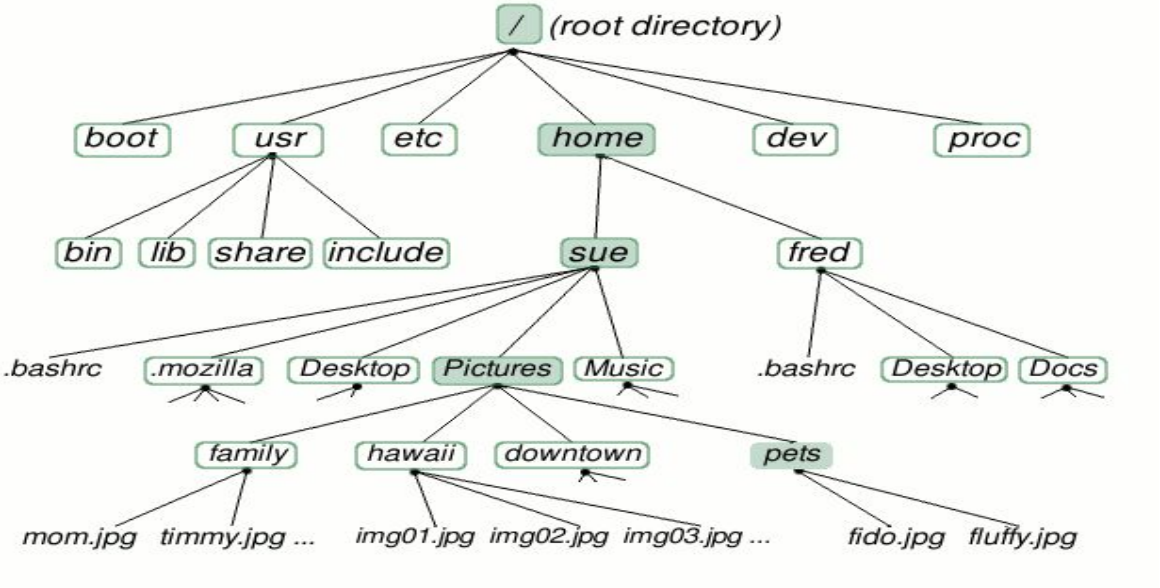
- Arquivos podem armazenar diversos tipos de informação:
  - Texto
  - Imagens
  - Aúdio
  - Vídeo
- A extensão do arquivo determina qual o seu tipo e, portanto, como a sequência de *bytes* deve ser interpretada. Alguns exemplos de extensões são:
  - .txt
  - .png, .jpg, .pbm
  - .wav, .flac, .mp3
  - .avi, .mpeg, .mkv

Introdução

- Arquivos são organizados em diretórios
- Um **diretório** é uma estrutura de catálogo do **sistema de arquivos** que contém referências a outros arquivos de computador, e, possivelmente, outros diretórios.

Introdução

- Um **sistema de arquivos** é uma estrutura que controla como os arquivos são armazenados.



Introdução

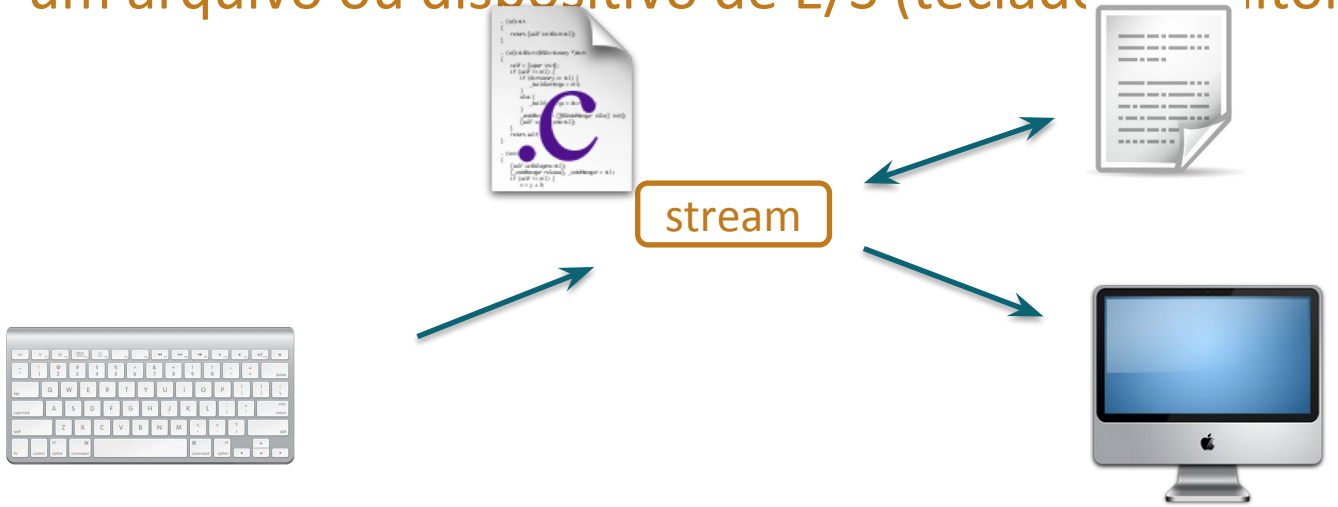
- Arquivos possuem atributos, os quais são metadados utilizados pelo sistema de arquivos:
  - Tamanho (em bytes)
  - Usuário que o criou
  - Datas (criação, ultima modificação, …)
  - Permissões

- *$ ls –l*

```
rw-r--r--     1 lucas  staff  154432 Jun  3 13:14  aula_arquivos.pptx
drwxr-xr-x  3 lucas  staff  102       Jun  3 13:15  testes
```

Arquivos em Linguagem C

- Um **arquivo** é representado como uma ***stream***
- Uma ***stream*** é uma representação lógica (em memória) de um arquivo ou dispositivo de E/S (teclado, monitor, etc)



Arquivos em Linguagem C

- Existem dois tipos de streams na linguagem C:
  - Texto: sequência de linhas, onde cada linha contém zero ou mais caracteres e termina com um \n
  - Binária: sequência de *bytes* que são lidos e gravados exatamente como estão armazenados em memória
- *Streams* padrão:
  - **stdin**: entrada de dados padrão de um programa
  - **stdout**: saída de dados padrão de um programa
  - **stderr**: saída de erros padrão de um programa

Arquivos em Linguagem C

- Em C, o tipo de dados que implementa uma **stream** é o FILE
- FILE é uma *struct* definida na biblioteca <*stdio.h*>:

```
typedef struct  {
        unsigned char *_p;      /* current position in (some) buffer */
        int       _r;           /* read space left for reading */
        int       _w;           /* write space left for writing */
        short   _flags;         /* flags, below; this FILE is free if 0 */
        short   _file;          /* fileno, if Unix descriptor, else -1 */
} FILE;
```

Arquivos em Linguagem C

- Os nomes de arquivos são armazenados em *strings*
- Cada sistema operacional utiliza uma forma de indicar o caminho do arquivo:

Windows

```
char *filename = "c:\\Users\\Lucas\\arquivo.txt";
```

Linux

```
char *filename = "/home/Lucas/arquivo.txt";
```


- *Obs: No Windows é realmente necessário utilizar duas barras invertidas para identificar um diretório.*

Arquivos – modo texto

- Seqüência de caracteres agrupadas em linhas.
- Números são guardados como cadeias de caracteres.
- Cada linha é separada pelo caracter 10 decimal (LF) ou o "\n".
- Existe uma indicação de fim de arquivo que é enviada pelo DOS e é reconhecida pelo compilador C.

Arquivos – modo texto

- Exemplo de arquivo modo texto:

```
Ainda que eu falasse\n
a língua dos homens\n
Que eu falasse\n
a língua dos anjos\n
Sem amor eu nada seria\n
EOF
```

Arquivos – modo binário

- As informações são armazenadas de acordo com o número de bytes que ocupam na memória.
  - 4 bytes para inteiro, 4 para float, 1 para char...
- Qualquer caractere é lido ou gravado sem alteração, mantendo-se a ordem de gravação realizada.
- Não existe indicação de fim de arquivo.

Arquivos – modo binário

- Exemplo de arquivo no modo binário:

```
0000   FF D8 FF E1  1D FE 45 78  69 66 00 00  49 49 2A 00
0010   08 00 00 00  09 00 0F 01  02 00 06 00  00 00 7A 00
0020   00 00 10 01  02 00 14 00  00 00 80 00  00 00 12 01
0030   03 00 01 00  00 00 01 00  00 00 1A 01  05 00 01 00
0040   00 00 A0 00  00 00 1B 01  05 00 01 00  00 00 A8 00
0050   00 00 28 01  03 00 01 00  00 00 02 00  00 00 32 01
0060   02 00 14 00  00 00 B0 00  00 00 13 02  03 00 01 00
0070   00 00 01 00  00 00 69 87  04 00 01 00  00 00 C4 00
0080   00 00 3A 06  00 00 43 61  6E 6F 6E 00  43 61 6E 6F
0090   6E 20 50 6F  77 65 72 53  68 6F 74 20  41 36 30 00
00A0   00 00 00 00  00 00 00 00  00 00 00 00  B4 00 00 00
00B0   01 00 00 00  B4 00 00 00  01 00 00 00  32 30 30 34
00C0   3A 30 36 3A  32 35 20 31  32 3A 33 30  3A 32 35 00
00D0   1F 00 9A 82  05 00 01 00  00 00 86 03  00 00 9D 82
00E0   05 00 01 00  00 00 8E 03  00 00 00 90  07 00 04 00
```

Arquivos

- Diferença principal:
  - Modo texto: possui caractere de nova linha e de fim de arquivo
  - Modo binário: informações armazenadas na forma de bytes, sem marca de final de arquivo ou final de linha.

Arquivos - modo binário

- Arquivos binários podem conter diferentes estruturas manipuladas pelos programas:
  - Vetores
  - Cadeia de caracteres
  - Matrizes
  - Registros

Arquivos - modo binário

- Em geral, arquivos são formados por uma coleção de registros. Cada registro é composto por campos.
  - Um dos campos é considerado campo-chave e é o campo que diferencia um registro dos demais.
- Um sistema de banco de dados é formado por um ou vários arquivos, com programas para: inclusão, exclusão, alteração, consultas...

Funções em C (stdio.h)

- fopen() = abre um arquivo
- fclose() = fecha um arquivo
- ferror() = retorna verdadeiro se ocorreu um erro
- fputc() = escreve um caracter em um arquivo
- fgetc() = lê um caracter de um arquivo
- fputs() = escreve uma string em um arquivo
- fgets() = lê uma string de um arquivo
- fwrite() = escreve uma estrutura (struct) em um arquivo
- fread() = lê uma estrutura (struct) de um arquivo
- fseek() = posiciona o arquivo em um byte específico
- feof() = retorna verdadeiro se atingiu o final do arquivo
- rewind() = coloca o ponteiro do arquivo no seu início
- remove() = apaga um arquivo
- fflush() = descarrega o conteúdo de um arquivo

Funções em C (stdio.h)

- A biblioteca também define a estrutura de arquivo a ser utilizada:
  - FILE
- Macros:
  - NULL = 0 (define um ponteiro nulo)
  - EOF = -1 (retorno da função)
  - FOPEN_MAX = máximo de arqs que podem ser abertos
  - SEEK_SET = 0 (início do arquivo)
  - SEEK_CUR = 1 (posição atual)

Ponteiro para Arquivo

- Definição de variável do tipo arquivo:
  - `FILE *arq;`
  - **arq é uma variável ponteiro capaz de identificar um arquivo no disco.**
    - **aponta para informações do arquivo: nome, status e posição do arquivo.**

Criar ou Abrir Arquivo

- Definição de variável do tipo arquivo:

```
FILE *arq;
```

- Função fopen(nome_arquivo, modo_abertura)
  - abre ou cria um arquivo, retornando o ponteiro apontado para o mesmo.
- Ex:

```
arq = fopen(nome_arquivo, modo_abertura)
```

Criar ou Abrir Arquivo

arq = fopen(nome_arquivo, modo_abertura)

nome_arquivo: string contendo o nome do arquivo para abrir ou criar, podendo incluir um *path*

modo_abertura: string que representa como o arquivo será aberto: escrita, leitura ...

Criar ou Abrir Arquivo

arq = fopen(nome_arquivo, modo_abertura)

Modo de Abertura:

w	Cria um arquivo texto para escrita (apaga se ele já existir)
r	Abre um arquivo texto para leitura
a	Abre um arquivo texto para anexar novos dados (no final)
wb	Cria um arquivo binário para escrita (apaga se ele já existir)
rb	Abre um arquivo binário para leitura
ab	Abre um arquivo binário para anexar novos dados (no final)

Criar ou Abrir Arquivo

Outros modos de Abertura:

w+	Cria um arquivo texto para escrita e leitura (apaga se o arquivo já existir)
r+	Abre um arquivo texto para leitura e escrita (o arquivo deve existir)
a+	Anexa novos dados ou cria um arquivo texto para leitura e escrita (se o arquivo não existir, cria o arquivo)
wb+	Cria um arquivo binário para escrita e leitura (apaga se o arquivo já existir)
rb+	Abre um arquivo binário para leitura e escrita (o arquivo deve existir)
ab+	Anexa novos dados ou cria um arquivo binário para leitura e escrita (se o arquivo não existir, cria o arquivo)

Criar ou Abrir Arquivo

- Exemplo:

```
FILE *arq1, *arq2;

arq1 = fopen("arquivo1.txt", "w");

arq2 = fopen("texto.txt", "a+");
```

Criar ou Abrir Arquivo

- Importante verificar se o arquivo foi criado ou aberto com sucesso!

```
FILE  *arq1, *arq2;
arq1 = fopen("arquivo1.txt", "w");
if (arq1==NULL) {
     printf("Erro na criação do arquivo);
   return(0);
}
else { ....}
```

Fechar Arquivo

- Função fclose(): fecha um arquivo. É importante que todo arquivo aberto seja fechado antes de terminar o programa!

```
int fclose(arq);
```

arq: ponteiro para um arquivo obtido pela função fopen()

Arquivo - modo texto

- Gravação e Leitura:
  - função fputs(): escreve uma cadeia de caracteres em um arquivo

```
fputs(cadeia_caracteres, arq)
```

  - função fgets(): lê uma cadeia de caracteres até tam ou até encontrar \n

```
fgets(cadeia_caracteres, tam, arq)
```

EXEMPLO

Funções de Manipulação de Arquivos

- Escrita de saída formatada
  - `int fprintf (FILE *stream, char *format,...);`
- Leitura de entrada formatada
  - `int fscanf(FILE *stream, char *format,...);`
- Insere uma string no arquivo.
  - `int fputs(char *str, FILE *stream);`
- Lê uma string do arquivo.
  - `char *fgets(char *str, int num, FILE *stream);`

Arquivo - modo binário

- **Gravação**:
  - função fwrite(): escreve qualquer tipo de dados em um arquivo

    fwrite(&dados, tamanho, qtde, arq)
- **Leitura**:
  - função fread(): lê qualquer tipo de dados em um arquivo

    fread(&dados, tamanho, qtde, arq)

dados = struct, vetor, matriz ou uma variável

Arquivo - modo binário

Exemplos:

```
fwrite(&agenda, sizeof(agenda), 1, arq)

fwrite(&x, sizeof(int), 1, arq)

fread(&agenda, sizeof(agenda), 1, arq)

fread(&x, sizeof(int), 1, arq)
```

Exemplo: Leitura Arquivo Texto

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

struct aluno{
    char nome[20];
    int idade;
    char sexo;
};

int main(){
    FILE *arq = fopen("arq-texto.txt", "r");
    if (arq == NULL){
        exit(0);
    }

    do{
        struct aluno aluno1;
        fscanf(arq, "%d %c %[a-z A-Z]s\n", &aluno1.idade, &aluno1.sexo, aluno1.nome);
        printf("%s\t%d\t%c\n", aluno1.nome, aluno1.idade, aluno1.sexo);
    }while(!feof(arq));

    fclose(arq);
    return 0;
}
```

Exemplo: Escrita Arquivo Texto

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

struct aluno{
    char nome[20];
    int idade;
    char sexo;
};
```

Exemplo: Escrita Arquivo Texto

```
11  int main(){
12      FILE *arq = fopen("arq-texto.txt", "w");
13      if (arq == NULL){
14          exit(0);
15      }
16      struct aluno aluno1;
17      strcpy(aluno1.nome, "jesimar");
18      aluno1.idade = 28;
19      aluno1.sexo = 'm';
20      struct aluno aluno2;
21      strcpy(aluno2.nome, "tiago");
22      aluno2.idade = 32;
23      aluno2.sexo = 'm';
24      struct aluno aluno3;
25      strcpy(aluno3.nome, "suzana");
26      aluno3.idade = 20;
27      aluno3.sexo = 'f';
28      fprintf(arq, "%s %d %c\n", aluno1.nome, aluno1.idade, aluno1.sexo);
29      fprintf(arq, "%s %d %c\n", aluno2.nome, aluno2.idade, aluno2.sexo);
30      fprintf(arq, "%s %d %c\n", aluno3.nome, aluno3.idade, aluno3.sexo);
31
32      fclose(arq);
33
34      return 0;
35  }
```

Exemplo: Leitura Arquivo Binário

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

struct aluno{
    int matricula;
    char nome[40];
    char sexo;
    int idade;
};

int main(){
    FILE *arq = fopen("conteudo-bin-aluno.bin", "rb");
    if (arq == NULL){
        exit(0);
    }
    typedef struct aluno tAluno;
    tAluno a1;
    while(fread(&a1, sizeof(tAluno), 1, arq)){
        printf("%s %d %c %d\n", a1.nome, a1.idade, a1.sexo, a1.matricula);
    }
    fclose(arq);
    return 0;
}
```

Exemplo: Escrita Arquivo Binário

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct aluno{
    char nome[40];
    int idade;
    char sexo;
    int matricula;
};

int main(){
    FILE *arq = fopen("conteudo-bin-aluno.bin", "wb");
    if (arq == NULL){
        exit(0);
    }
    typedef struct aluno tAluno;
    tAluno aluno1;
    strcpy(aluno1.nome, "jesimar da silva arantes");
    aluno1.idade = 28;
    aluno1.sexo = 'm';
    aluno1.matricula = 12345;
```

Exemplo: Escrita Arquivo Binário

```
22
23      tAluno aluno2;
24      strcpy(aluno2.nome, "marcio da silva arantes");
25      aluno2.idade = 30;
26      aluno2.sexo = 'm';
27      aluno2.matricula = 45522;
28
29      tAluno aluno3;
30      strcpy(aluno3.nome, "suzana oliveira");
31      aluno3.idade = 23;
32      aluno3.sexo = 'f';
33      aluno3.matricula = 63452;
34
35      fwrite(&aluno1, sizeof(tAluno), 1, arq);
36      fwrite(&aluno2, sizeof(tAluno), 1, arq);
37      fwrite(&aluno3, sizeof(tAluno), 1, arq);
38      fclose(arq);
39      return 0;
40  }
```

Função fseek

- Modifica a posição do ponteiro no arquivo

fseek(arq, qtde, origem)

qtde: número de bytes a partir da origem que deve ser feito o deslocamento do ponteiro

origem: uma das macros definidas para stdio.h:

- SEEK_SET = 0 (início do arquivo)
- SEEK_CUR = 1 (posição atual)
- SEEK_END = 2 (final do arquivo)

Exemplo: Função fseek

exclusao-arquivo.c