Universidade Federal do Piauí
Campus SHNB – Picos -PI
Curso de Sistemas de Informação

Algoritmos e Programação II

Glauber Dias Gonçalves
ggoncalves@ufpi.edu.br

Conteúdo

- Fundamentos de processamento de arquivo
- Abertura, escrita e leitura de arquivos em C
- Acessos sequencial e aleatório em arquivos em C

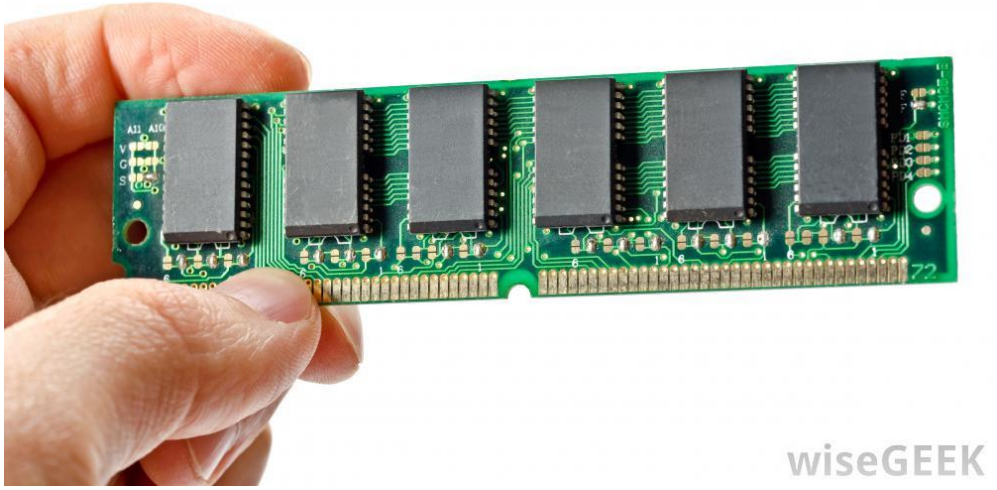
Bibliografia

- Deitel, Paul & Deitel, Harvey. C Como Programar. 6 . Ediçao. Editora Pearson, 2011 (**Cap 11**)

Fundamentos

Memória Primária

- Dados na memória primária (RAM) são temporários
  - Variáveis de um programa: vetores, estruturas e etc.
  - Armazenamento não persistente (volátil)
    - Enquanto o computador está ligado



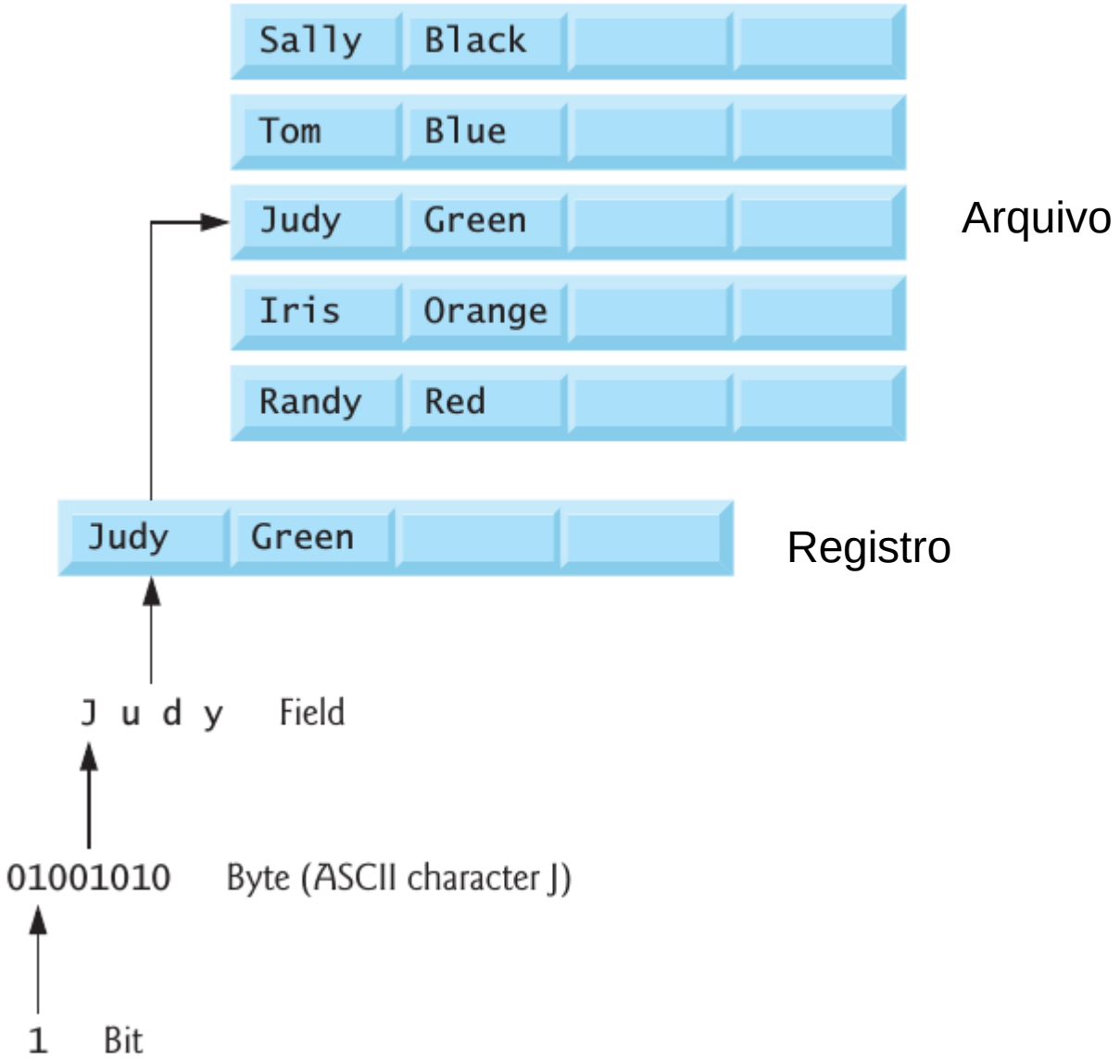
Memória Secundária

- Arquivos: dados armazenados de forma persistente
  - Podem ser acessados ao desligar/ligar o computador
- Dispositivos de armazenamento secundários
  - Podem armazenar grande volume de dados

Dados e Informação!

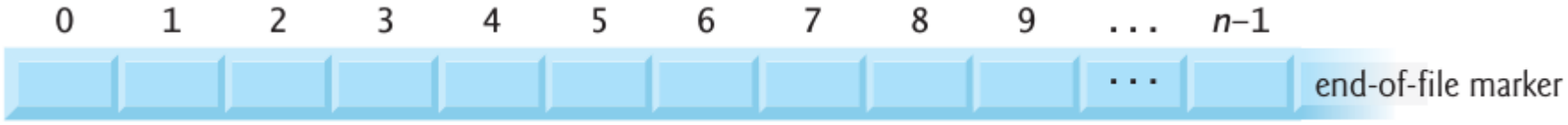
- Organização de dados no computador
  - Todo dado no computador é uma combinação de 0-1 (bits)
  - Os bits formam bytes (8 bits)
  - Bytes representam números ou caracteres
    - Dado → informação
  - Conjunto de caracteres (vetores) podem formar nomes
  - Registros: conjunto de nomes e/ou números

Dados e Informação!



Fluxo de bytes (*stream*)

- Arquivos são processados como um fluxo de bytes



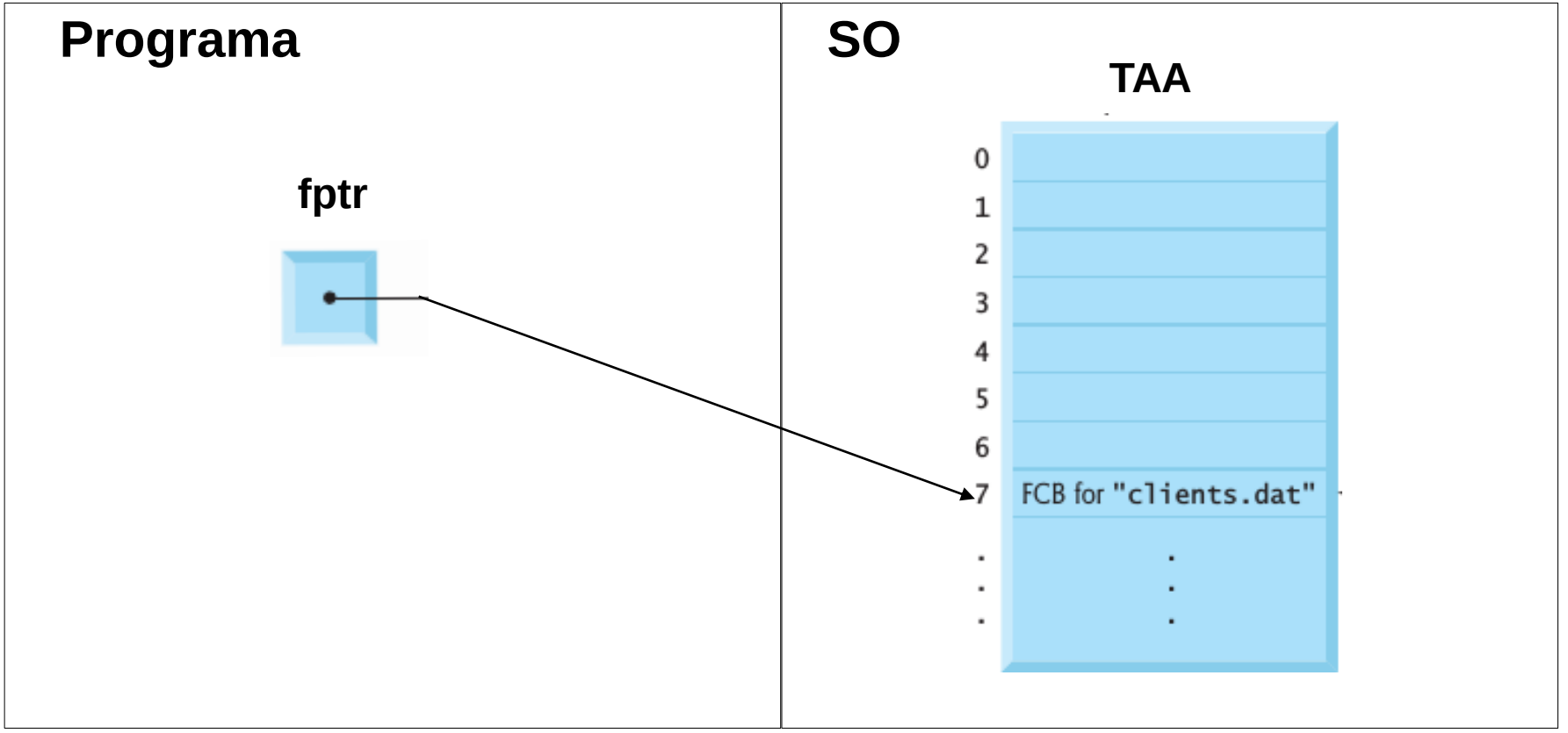
- O arquivo finaliza com um byte especial (EOF)
- Abrir um arquivo significa:
  - Associar o arquivo (end. memória) a um fluxo de bytes

Controle do Sistema Operacional

- Tabela de Arquivos Abertos (TAA)
  - O sistema operacional controla a TAA
    - Mais de um programa pode acessar o mesmo arquivo!
  - Cada entrada da TAA é chamada FCB (*File Control Block*)
- Abertura de arquivo em um programa:
  - Instanciar uma variável do tipo arquivo para uma posição na TAA
  - Parâmetros: endereço e modo de abertura do arquivo (r/w)

Controle do Sistema Operacional

arquivo *fptr = abrir(endereço_arquivo, modo_abertura)



Abertura, Escrita e Leitura de Arquivos em C (Processamento Básico de Arquivos)

Fluxo de bytes (*stream*)

- Programas em C usam 3 fluxos de bytes padrões:
  - **stdin**: entrada padrão de dados via teclado
  - **stdout**: saída padrão de dados via tela
  - **stderr**: saída padrão de dados via tela – para erros!
- Automáticos em muitas funções em C (stdio.h)
  - *printf, scanf, getchar, putchar, gets, puts* e outras ...

Fluxo de bytes (*stream*)

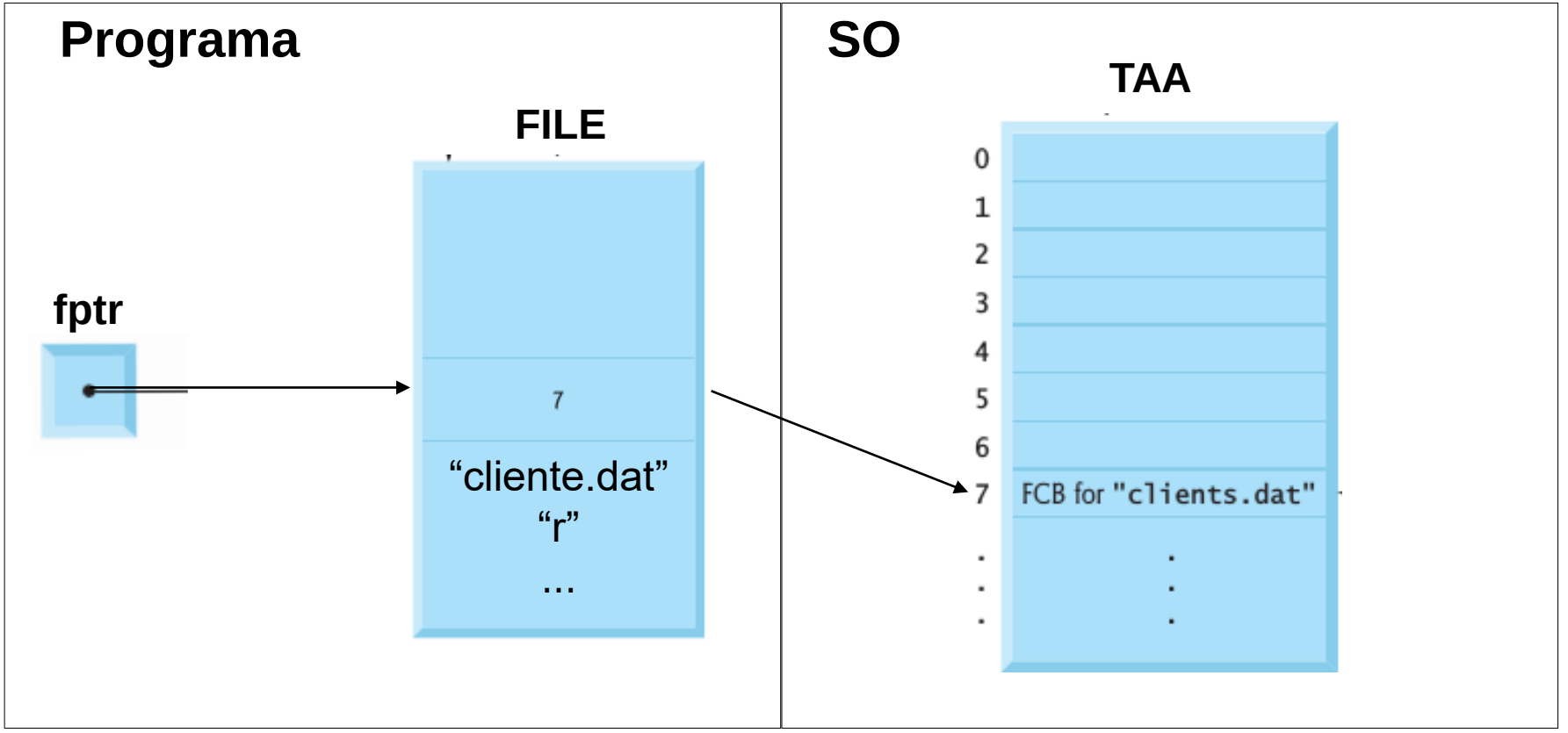
- Ver uso de getchar, gets, putchar e puts
  - Código: std_io_getchar.c e std_io_gets.c

Abertura de Arquivo

- Abertura de arquivo em um programa:
  - instanciar um ponteiro para o tipo **FILE**
- FILE é um TAD definido na biblioteca stdio.h
  - Contem informações de um arquivo
  - Exemplo: nome, caminho, posição (índice) na TAA

Abertura de Arquivo

File *fptr = fopen(“clients.dat”,”r”)



Abertura de Arquivo

- Função para abertura de um arquivo (2 parâmetros)

  FILE *fopen(const char *filename, const char *mode)

  - filename: caminho e nome do arquivo
  - modo: forma de abertura do arquivo (leitura/escrita)
  - o retorno é um ponteiro para estrutura FILE ou NULL (erro)

- Função para fechamento de um arquivo (1 parâmetro)

  fclose(FILE *fptr)

  - ponteiro para estrutura FILE

Abertura de Arquivo

Tipos de arquivo

- Arquivo texto: caracteres que podem ser vistos diretamente na tela ou modificados por editores de texto.
  - Exemplos: código C, texto simples, páginas HTML
- Arquivo binário: Sequência de bits sujeita às convenções dos programas que o gerou, não legíveis diretamente
  - Exemplos: arquivos de imagem (jpeg, png), arquivos compactados (zip, rar)

Abertura de Arquivo

- Principais modos:

“r”	Abre um arquivo texto para leitura. O arquivo deve existir antes de ser aberto.
“w”	Abrir um arquivo texto para gravação. Se o arquivo não existir, ele será criado. Se já existir, o conteúdo anterior será destruído.
“a”	Abrir um arquivo texto para gravação. Os dados serão adicionados no fim do arquivo (“append”), se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
“rb”	Abre um arquivo binário para leitura. Igual ao modo “r” anterior, só que o arquivo é binário.
“wb”	Cria um arquivo binário para escrita, como no modo “w” anterior, só que o arquivo é binário.
“ab”	Acrescenta dados binários no fim do arquivo, como no modo “a” anterior, só que o arquivo é binário.
“r+”	Abre um arquivo texto para leitura e gravação. O arquivo deve existir e pode ser modificado.

Abertura de Arquivo

- Ver código abertura e fechamento de arquivos
  - open_close.c

Escrita de Arquivo

- Funções para escrita

int fputc(char chr, FILE *fptr)

- Retorna o caracter escrito ou -1 (erro)

int fputs(char *chr, FILE *fptr)

- Retorna um valor positivo ou -1 (erro)

int fprintf(FILE *stream, const char *format, ...)

- Retorna o número de caracteres escritos ou -1 (erro)

Escrita de Arquivo

- Ver código escrita de arquivos
  - escreve.c

Leitura de Arquivo

- Funções para leitura

int fgetc(char chr, FILE *fptr)

- Retorna o caracter lido ou EOF (erro)

char * fgets(char *chr, int tam, FILE *fptr)

- Retorna a *string* lida ou EOF (erro)

int fscanf(FILE *stream, const char *format, ...)

- Retorna o número de caracteres lidos ou EOF (erro)

Leitura de Arquivo

- Ver código escrita de arquivos
  - le.c

Acesso Sequencial e Aleatório em Arquivos em C

Acesso Sequencial

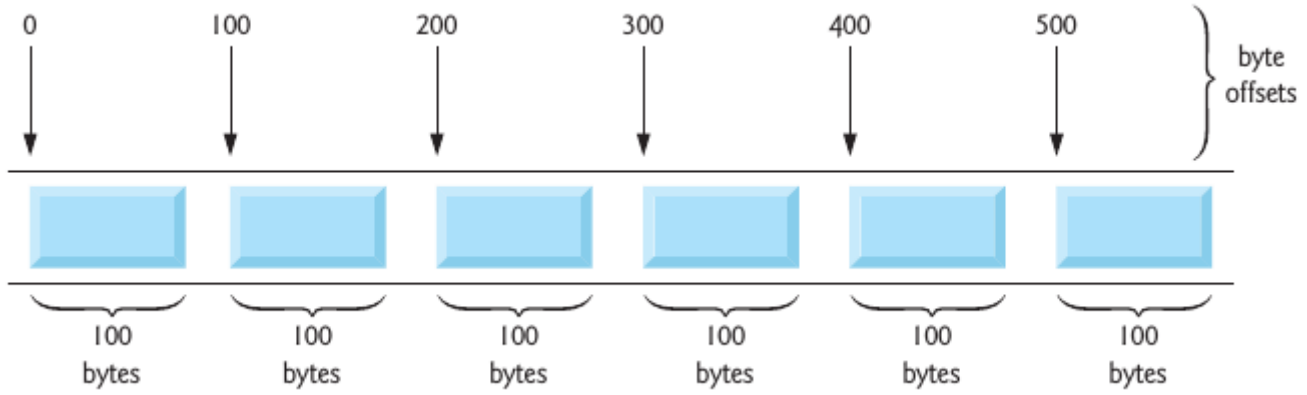
- Arquivos sem estrutura/organização
  - Geralmente arquivos do tipo texto
- A leitura desses arquivos deve ser sequencial
  - Byte por byte: ler cada caractere/palavra sequencialmente
    - Como nos exemplos anteriores

Acesso Sequencial

- Ver exemplos de escrita/leitura em arquivo sequencial
  - Livro Deitel como programar 6 . (Seções 11.4 e 11.5)
  - sequencial_escrita.c
  - sequencial_leitura.c

Acesso Aleatório

- Arquivo com estrutura/organização
  - Geralmente arquivos do tipo binário



- Acesso para escrita/leitura/busca pode ser aleatória
  - Funções *fread, fwrite e seek*

Acesso Aleatório

- Funções fread/fwrite: leitura/escrita em blocos de dados

  size_t fread(void *buffer, size_t nbytes, size_t nitens, FILE *fptr)

  size_t fwrite(void *buffer, size_t nbytes, size_t nitens, FILE *fptr)

  - buffer é um ponteiro para bloco (vetor) de dados a ler/escrever
  - nbytes é o número de bytes a ler/escrever
  - nitens é a qtd. de itens com nbytes a serem lidos/escritos
  - fPtr é o ponteiro para a estrutura FILE
  - Retorno é o número de itens lidos

Acesso Aleatório

- Função fseek: leva a uma posição dentro do arquivo estruturado para posterior leitura/escrita.

  int fseek(FILE *fptr, long int nbytes, int origem)

- nbytes é o número de bytes, a partir de origem, que se deseja avançar.
- origem é a posição de referencia para início da busca
  - SEEK_SET (início do arquivo)
  - SEEK_CUR (posição atual)
  - SEEK_END (final do arquivo)

Acesso Aleatório

- Ver exemplos de escrita/leitura em arquivo aleatório
  - Livro Deitel como programar 6 . (Seções 11.7 e 11.8)
  - aleatoria_criacao.c
  - aleatoria_escrita.c
  - aleatoria_busca.c

Sumário

- Fundamentos de processamento de arquivo:
  - Armazenamento persistente de dados
  - Memória secundária
- Abertura, escrita e leitura de arquivos
  - Arquivos funcionam como fluxos de dados
- Acessos sequencial e aleatório em arquivos em C
  - A escolha da forma de acesso depende da aplicação!
- Lista de Atividades 10 (a última)