# ESTRUTURAS DE DADOS E ALGORITMOS

Arquivos

# File

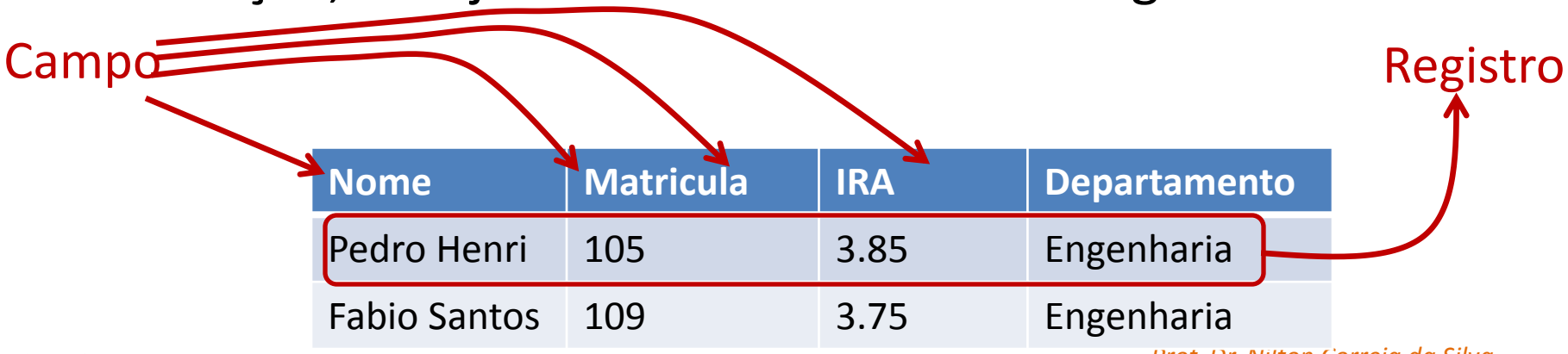
- Até agora trabalhamos somente com registros guardados em memória volátil. (RAM)
  - Ex. Leitura de contas bancárias, trabalho 03 de figuras geométricas (struct)
- Arquivo pode ser usado para armazenar dados permanentemente (Hard Disk)
- Os dados em arquivo podem ser recuperados mesmo que o computador seja desligado
- O programa C é capaz de ler, escrever, modificar e apagar conteúdo de arquivo

# Cont.

- Uma hierarquia comum de arquivo pode ser subdividida em cinco categorias
  - Bit Binary : digit, 0 or 1
  - Byte : oito bits
  - Field : Agrupamento de bytes
  - Record : Agrupamento de campos
  - File : Agrupamento de registros

# Campos, Registros

- Campos podem ser um nome, o IRA, o departamento, o endereço, o telefone e por ai vai
- *Registros são agrupamentos lógicos de campos que compreende uma única linha de informação*
- Um registro de aluno pode compreendendo dos campos nome, idade, matricula e o IRA. Cada campo é único na descrição, mas juntos formam um único registro




| Nome | Matricula | IRA | Departamento |
|---|---|---|---|
| Pedro Henri | 105 | 3.85 | Engenharia |
| Fabio Santos | 109 | 3.75 | Engenharia |

# Arquivos de Dados

- *Arquivos de dados* compreendem um ou mais regs
- Os arquivos podem ser usados para armazenar todos os tipos de dados, como os de aluno e de funcionário

| | | | |
|---|---|---|---|
| Fatima Tariq | 102 | 3.45 | Physics |
| Ali Ahmad | 105 | 3.85 | Computer science |
| Fahad Hamid | 109 | 3.75 | Computer science |

- Programadores C usam ponteiros para gerenciar arquivo para leitura e escrita

# Operações com Arquivos

- Criação de um novo arquivo
- Abertura de um arquivo existente
- Leitura de um arquivo
- Escrita em arquivo
- Mover a um local específico no arquivo (seeking)
- Fechamento de arquivo

# Código de Exemplo

```
main( )
{
   FILE *fp ;
   char ch ;
   fp = fopen ( "prog1.c", "r" ) ;
   while ( 1 )
   {
      ch = fgetc ( fp ) ;
      if ( ch == EOF )
        break ;
      printf ( "%c", ch ) ;
   }
   fclose ( fp ) ;
}
```

Go to program

# Abertura de arquivo

- Antes de ler e escrever em arquivo, este precisa esta aberto
- A função fopen() é usada para abrir um arquivo
- O código de exemplo abre “prog1.c” no modo leitura
- Modo leitura significa que o conteúdo do arquivo não pode ser modificado

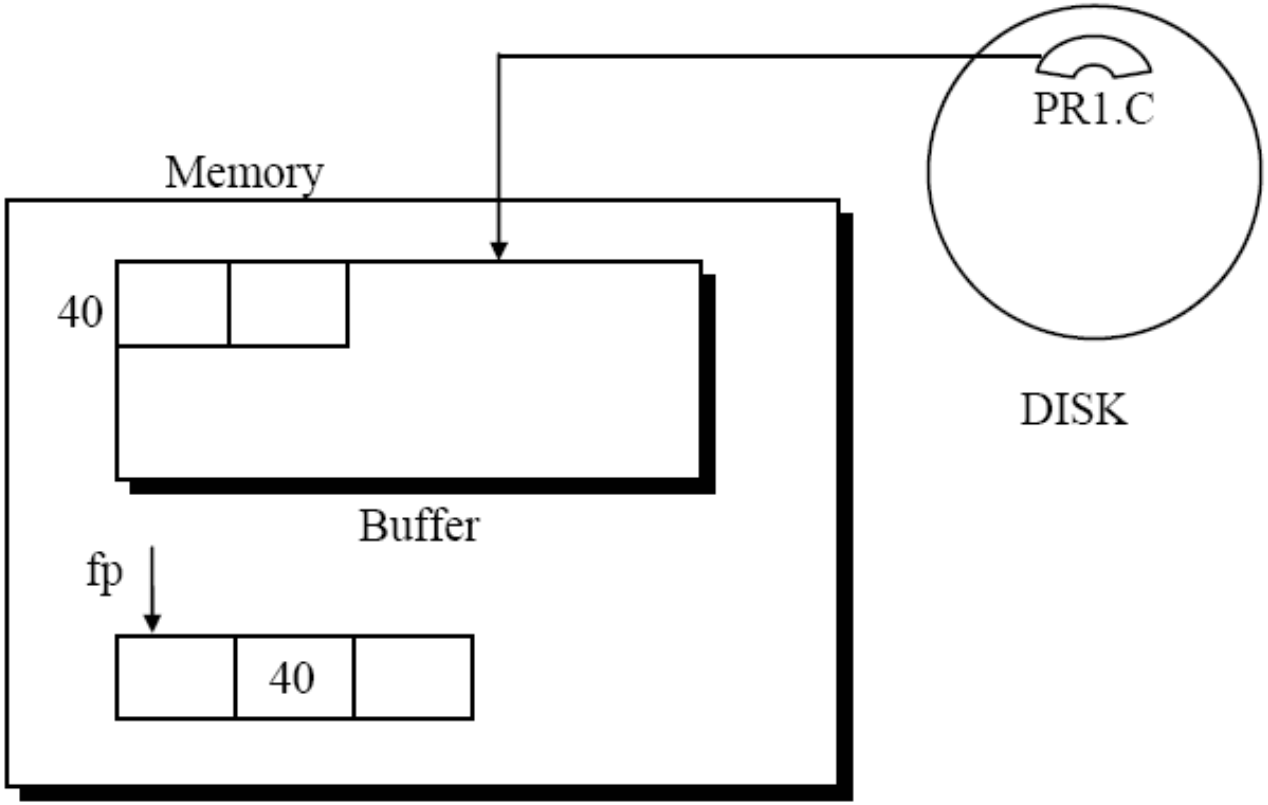
```
fp = fopen ( "prog1.c", "r" ) ;
```

# Tarefas da função fopen( )

1. Busca no disco pelo arquivo a ser aberto
2. O arquivo é carregado do disco em um espaço em memória chamado buffer
3. É definido um ponteiro para char que aponta para o primeiro caractere em buffer

É muito mais rápido ler/escrever conteúdo em memória, quando se compara ao disco.

# Cont.



# Estrutura FILE

- Para uma leitura bem sucedida de arquivo, dados como o modo de abertura, tamanho do arquivo, local no arquivo onde a próxima operação de leitura acontecerá, etc. precisam ser mantidos
- fopen( ) armazena tal informação em uma estrutura e retorna o seu endereço
- Esse endereço é armazenado em ponteiro para FILE

```
FILE *fp ;
fp = fopen ( "prog1.c", "r" ) ;
```

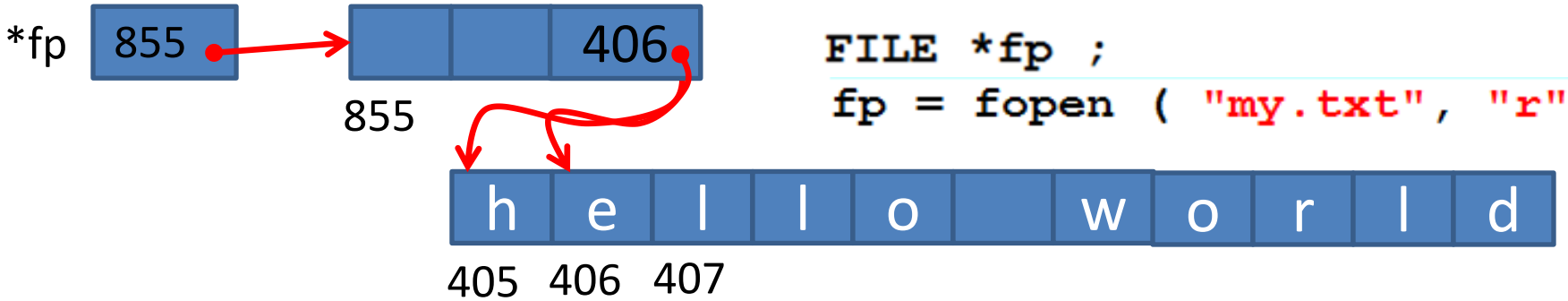
# Leitura de arquivo

- Ao abrir um arquivo via **fopen** ( ), o seu conteúdo é trazido para buffer
- É definido um ponteiro que aponta para o primeiro caractere desse buffer
- Esse ponteiro é um dos elementos da estrutura para a qual **fp** aponta
- Para ler o conteúdo do arquivo da memória use uma função chamada **fgetc( )**

# Cont.

ch = fgetc ( fp ) ;

- **fgetc( )** lê caractere da posição corrente
- avança a posição do ponteiro, de modo que agora ele aponta para o próximo caractere e,
- retorna o caractere lido




```
FILE *fp ;
fp = fopen ( "my.txt", "r");
```

ch = fgetc ( fp ) ;

ch h

# Cont.

- Aberto o arquivo, podemos referencia-lo pelo FILE pointer, ao inves do seu nome
- A função **fgetc( )** dentro de um loop infinito **while**
- O loop será executado até chegar ao fim do arquivo

```
if ( ch == EOF )
  break ;
```

- EOF é um caractere especial de valor ASCII igual a 26
- EOF é inserido após o último caractere no arquivo


- EOF é uma macro definida em stdio.h.

# Problema na abertura de arquivo

- Existe a chance de quando se tentar abrir um arquivo via **fopen( )**, de que o arquivo não seja aberto
- Por exemplo, o arquivo não exista no disco
- Outras razões
  - o espaço em disco pode ser insuficiente para abrir um novo arquivo, por espaço em disco insuficiente para abrir um novo arquivo, ou o disco estar protegido contra escrita, ou esteja danificado …
- Caso a abertura apresenta falha, a função **fopen( )** retorna um valor NULL

# Exemplo

```
FILE *fp ;
fp = fopen ( "prog1.c", "r" );
if(fp == NULL)
{
    printf("Unable to open file\n");
    system("pause");
    exit(0);
}
```

Essa verificação deve ser realizada para cada programa que envolva I/O de arquivo

# Fechamento de arquivo

- Após a leitura/escrita do arquivo, ele deve ser fechado
- Uma vez fechado, não é possível fazer operações no arquivo
- fclose() realiza tres operações
  - Os caracteres no buffer serão escritos no arquivo no disco
  - No final do arquivo um caracteree ASCII 26 será escrito (EOF)
  - O buffer sera eliminado da memória

# Exemple

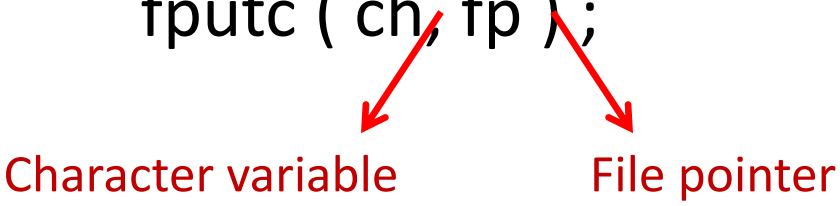
- Escreva um programa que leia um arquivo e conte quantos caracteres, espaços, tabs e newlines estão presentes

write a program

# Writing character in file

- fputc( ) is used to writes character in a file

fputc ( ch, fp ) ;




fputc write the character constant store in character variable in the buffer that is pointer by a pointer fp

# File Opening Modes

- "r" :

  reading from file, if file does not exist it returns null
- "w" :

  writing to the file, if file does not exits a new file is created
- "a" :

  adding new contents at the end of file, if file does not exits a new file is created

# File opening mode

- "r+" :

  Reading, writing, modifying content of file, if file does not exist it returns null

- "w+" :

  Reading, writing, modifying content of file, if file does not exits a new file is created

- "a+" :

  Reading, adding new contents at the end of file, cannot modify existing contents. if file does not exits a new file is created

# A File-copy Program

- Write a program that create a copy of a file. The user input two file names
  - the name of file to be copied
  - Copy file name

write a program

# Writing string in a file

## fputs ( s, fp ) ;

- fputs( ) function writes the contents of the array pointer by pointer s to the file pointed by pointer fp

```
printf ( "\nEnter a few lines of text:\n" ) ;
while ( strlen ( gets ( s ) ) > 0 )
 {
fputs ( s, fp ) ;
fputs ( "\n", fp ) ;
 }
```

Go to program

Enter a few lines of text
Hello world
C programming

Hello world
C programming

# Reading string from a file

fgets ( s, 79, fp ) ;

- This function read 79 bytes (character) from file pointed by pointer fp and write in the memory address s.
- s is base address of a character array
- This function returns NULL if no record is available to read

```
while ( fgets ( s, 79, fp ) != NULL )
  printf ( "%s" , s ) ;
```

Go to program

# New line character

- If we write the following on file using fputs

```
Hello world
programming
```

- Actual number of character in file are 24
- But the size of file will be 26
- Reason : **fputs( )** converts the **\n** to **\r\n** combination

# Cont.

- If we read the same line back using **fgets( )** the reverse conversion happens.
- Thus when we write the first line of the poem and a “\n” using two calls to **fputs( )**, what gets written to the file is

  Baa baa black sheep have you any wool \r\n

Go to program

# Writing integer and float in file

- Until now we have seen following functions
  - fgetc(fp); fputc(ch, fp);
  - fgets(s, 79, fp); fputs(s, fp)
- C provide functions to write integer and float values in file
  - fprintf - use to print (write) on file
  - fscanf - use to scan (read) a file

# fprintf Syntax

```
struct book
{
    char name[20] ;
    float price ;
    int pages ;
};
struct book b1 = { "Let us C", 350.75, 550 } ;
```

- printf("%s %f %d", b1.name, b1.price, b1.pages);

  It prints let us c 350.75 550 on standard output (monitor)

- fprintf(fp, "%s %f %d", b1.name, b1.price, b1.pages);

  it prints lets us c 350.75 550 on file pointed by pointer fp

# fscanf syntax

```
struct book
{
    char name[20] ;
    float price ;
    int pages ;
};
struct book b1;
```

- scanf("%s %f %d", b1.name, &b1.price, &b1.pages);

  It scan/read values (name, price, pages) from standard input (keyboard)

- fscanf(fp, "%s %f %d", b1.name, &b1.price, &b1.pages );

  It scan/read values (name price, pages) from a file pointed by pointer fp

# Example program – fprintf

- Input record for employ
  - Name
  - Age
  - Basic salary
- Input record as much as user required
- Write record in file employee.dat

Go to program

# Example program – fscanf

- Write a program to display the content of file that have been created in the previous example program
  - Name
  - Age
  - Basic salary

Go to program