Universidade de São Paulo
Brasil

# Aula 14
# Introdução à Linguagem C (Parte 2)

## MAC0216 - Técnicas de Programação I

Professores: Alfredo, Daniel, Fabio e Kelly

**Departamento de Ciência da Computação**
**Instituto de Matemática e Estatística**

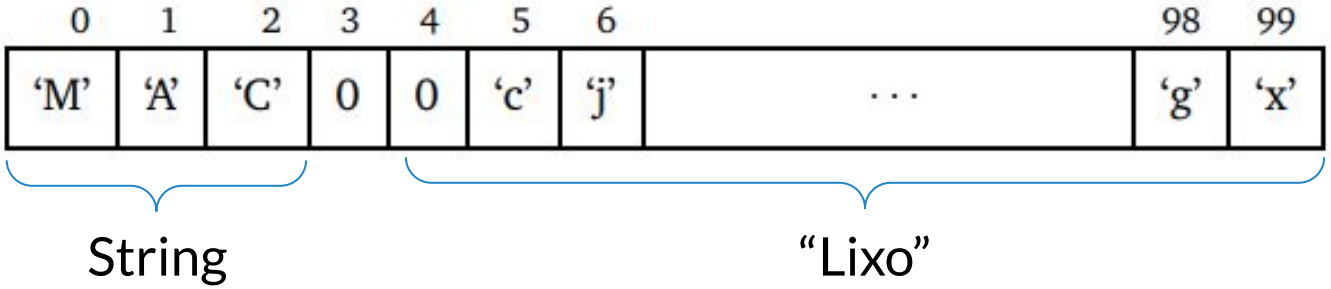
# Strings

## Biblioteca string.h

# Strings

- Uma string é um vetor de char terminado com o '\0' (caracter que tem código 0)

```
char my_str[100];
```



# Caracteres e Strings

- Tipo char ocupa 1 byte
- Um caractere pode ocupar mais de um byte
  - Depende do esquema de codificação usado
  - ASCII: códigos de 1 byte
  - UTF-8: códigos de 1 a 4 bytes
- Ex.: char *s = “Computação”

  strlen(s) != qtde de caracteres da string s

  12 != 10

# Funções de Manipulação

- A biblioteca string.h contém funções de manipulação de strings em C
- A página do professor Paulo Feofiloff mostra algumas dessas funções

  http://www.ime.usp.br/~pf/algoritmos/apend/string.h.html
- A Wikipédia tem mais alguns detalhes
  http://en.wikipedia.org/wiki/C_string_handling

# Biblioteca string (<string.h>)

- Devolve o comprimento da string x

```
unsigned int strlen (char *x);
```

- Copia a string y para a string x

```
char *strcpy (char *x, char *y);
```

- Compara duas strings

```
int strcmp (char *x, char *y);
```

- Concatena duas strings (acrescenta y em x)

```
char *strcat (char *x, char *y);
```

# Arquivos

## Funções da stdio.h

# Arquivos

- Arquivo - sequência de bytes armazenada em memória secundária
- Acesso aos dados é sequencial
  - Bytes do arquivo não podem ser endereçados individualmente
- Abertura - associação do arquivo a uma variável do tipo FILE (stdio.h)
  - FILE é uma estrutura que encapsula o que é necessário para a manipulação do arquivo como um stream, como o descritor do arquivo,  a posição atual e o indicador de erro (entre outros)

# Abertura e Fechamento de Arquivos

**FILE * fopen (char const *path, char const *mode);**

- *path* - nome do arquivo com caminho
- *mode* - modo de abertura do arquivo
- Retorna NULL quando não foi possível abrir o arquivo

**int fclose (FILE *stream);**

# Modos de Abertura para fopen e freopen

| | |
|---|---|
| `"a"` | Creates an empty text file if necessary; open for writing at end-of-file |
| `"w"` | Creates an empty text file or wipes out content; open for writing |
| `"r"` | Opens an existing text file for reading |
| `"a+"` | Creates an empty text file if necessary; open for reading and writing at end-of-file |
| `"w+"` | Creates an empty text file or wipes out content; open for reading and writing |
| `"r+"` | Opens an existing text file for reading and writing at beginning of file |
| `"ab"` `"rb"` `"wb"` `"a+b"` `"ab+"` `"r+b"` `"rb+"` `"w+b"` `"wb+"` | Same as above, but for a binary file instead of a text file |
| `"wx"` `"w+x"` `"wbx"` `"w+bx"` `"wb+x"` | Same as above, but error if the file exists prior to the call |

# Escrita em Arquivos Texto

- Para escrever um único caracter:

**int fputc (int c , FILE * stream) ;**

- Para escrever uma string:

**int fputs (char const *s, FILE * stream) ;**

- Para escrever um valor formatado:

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

# Buffer de escrita

- O SO pode postergar a escrita física num stream, por questões de eficiência
  - Dados são acumulados em um *buffer*
- O *buffer* é "descarregado" quando o stream é fechado
- A função **fflush** pode ser usada para forçar que o *buffer* seja escrito imediatamente
- Para arquivos texto, a forma mais comum de *buffering* é por linha
  - Saída é fisicamente escrita quando o fim da linha é encontrado

# Leitura de Arquivos Texto

- Para leitura de um único caracter

**int fgetc (FILE *stream);**

  - devolve o cast do valor char lido para int

- Para a leitura de uma linha inteira (string)

**char *fgets (char *s, int size, FILE *stream);**

  - devolve s em caso de sucesso, NULL em caso de falha
  - size: máximo de caracteres lidos
  - o “\n” (caso seja lido), é incluído em s

- Para leitura de valores com formato especificado

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

  - Devolve a qtde de objetos lidos

# Escrita na Saída Padrão

int putchar (int c);

Ex.: putchar (c); fputc (c, stdout);

int puts (const char *s);

Ex.: puts (s); fputs (s, stdout); fputc ('\n', stdout);

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

Ex.: printf ("%d", x); fprintf (stdout, "%d", x);

# Leitura da Entrada Padrão

**int getchar (void);**

Ex.: char c = getchar (c); fgetc (c, stdin);

**char *gets (char *s);**

Obsoleta (removida no C11)

Ex.: if (gets (s)) if (fgets (s, tam_limite, stdin))

**int scanf (const char *format, ...);**

Ex.: scanf (“%d”, x) fscanf (stdin, “%d”, x)

# Leitura e Escrita em Arquivos Binários

**size_t fread(void *ptr, size_t size, size_t nmemb,**
**FILE *stream);**
**size_t fwrite(const void *ptr, size_t size, size_t nmemb,**
**FILE *stream);**

- ptr - ponteiro para os dados a serem escritos
- size - tamanho do tipo de dado
- nmemb - quantidade de itens do tipo de dado
- stream - ponteiro para o arquivo

# Reposicionamento no arquivo

- Para reposicionar o indicador de posição:

**int fseek(FILE *stream, long offset, int whence);**

- *whence* - posição de referência; opções:
  - SEEK_SET - início do arquivo
  - SEEK_CUR - posição atual
  - SEEK_END - fim do arquivo
- *offset* - deslocamento em bytes (relativo à posição de referência) para o reposicionamento

A nova posição é dada pela adição de offset à posição especificada por whence.

# Atenção!

- Cuidado com a abertura de arquivos para leitura e escrita ao mesmo tempo
  - O gerenciamento é complicado
- Para garantir que as leituras sejam feitas sempre sobre os dados mais recentes, recomenda-se a chamada de uma função de reposicionamento entre uma escrita e uma leitura
  - Ex.: fseek(stream, 0, SEEK_END)

# Miscelânea

# O tipo de dado bool (_Bool)

- Foi introduzido na versão C99
- Está definido em <stdbool.h>
- Valores: true e false
  - Tecnicamente, são apenas apelidos para os valores 1 e 0
  - Podem ser usados para enfatizar que o valor deve ser interpretado como uma condição
- O resultado dos operadores relacionais e lógicos é sempre um valor do tipo bool
  - ==, != >, >=, <, <= , !, &&, ||

# Seleção de fluxo com switch

```
switch (<expressão>)
{
   case <constante_1>:
     <comandos_1>;
   case <constante_2>:
     <comandos_2>;
   ...
   default
     <comandos_n>;
}
```

- A expressão e as constantes precisam ser um valor integer ou um caractere
- Uma vez que a execução entrou no bloco, ela vai continuar até encontrar um comando *break* ou o fim do bloco

```c
#include <stdio.h>
int main() {
 int number = 2;
 switch (number)  {
 case 1:
 case 2:
 case 3:
   printf("Um, doi ou três.\n");
   break;
 case 4:
 case 5:
   printf("Quatro e cinco.\n");
   break;
 default:
   printf("Maior que cinco.\n");
 }
}
```

# Referências Bibliográficas


- Carlos Hitoshi Morimoto, Ronaldo Fumio Hashimoto, ***Introdução a Ciência da Computação em C*** (Apostila)

  https://www.ime.usp.br/~hitoshi/introducao/index.html
- Jens Gustedt, ***Modern C***, Manning, 2019.

  https://modernc.gforge.inria.fr/
- E. Roberts, ***The Art and Science of C***, Addison-Wesley, 1995.
- Paulo Feoffilof, ***Projeto de Algoritmos em C***, ver links na seção "Recursos da linguagem C:" https://www.ime.usp.br/~pf/algoritmos/
- Exercícios

  https://www.ime.usp.br/~macmulti/