

# 18ª Aula - Biblioteca Standard (I)

## Programação Mestrado em Engenharia Física Tecnológica

Samuel M. Eleutério  
sme@tecnico.ulisboa.pt

Departamento de Física  
Instituto Superior Técnico  
Universidade de Lisboa

# Biblioteca Standard

- As funções de **C** encontram-se agrupadas de acordo com as características e as suas descrições encontram em ficheiros **“.h”**.
- O **C standard** contém na sua biblioteca um conjunto de funções que iremos descrever sucintamente.
- Para além deste **conjunto standard**, cada implementação tem mais **funções** definidas que podem ser encontradas nos respectivos manuais. Aqui, vamos limitarmo-nos ao **conjunto standard**.
- Associado a essa biblioteca existe um conjunto de **“header files”** (ficheiros **‘.h’**) que, para além dos **protótipos das funções**, contém ainda a definição de **macros** e outros elementos de programação.
- A seguinte encontra-se a lista completa das **“header files”**.
- Estes ficheiros **‘.h’** podem ser incluídos por **qualquer** ordem e até mesmo serem **repetidos**.

# Biblioteca Standard

- Não é nossa intenção fazer aqui uma **descrição completa** de **todas** as funções definidas na **biblioteca standard**. Nem tal seria muito interessante.
- Na **bibliografia** apresentada ela está descrita em certo detalhe.
- Para além dessa bibliografia, o **manual da GNU** sobre a biblioteca de **C** (**'libc'**) tem uma descrição bastante detalhada, sendo um importante elemento de consulta.
- Ver: '<http://www.gnu.org/software/libc/manual/>'. Aí podem ser encontradas versões em '**pdf**' e em '**html**' de elevada qualidade.
- Iremos também usar **programas** em que se **exemplifica** como utilizar algumas funções.

# Biblioteca Standard (I)

Header file	Descrição
<a href="#">assert.h</a>	<b>Diagnóstico</b>
<a href="#">ctype.h</a>	Tratamento de <b>caracteres</b>
<a href="#">errno.h</a>	Tratamento de <b>erros</b>
<a href="#">float.h</a>	Características das variáveis de <b>tipo real</b>
<a href="#">limits.h</a>	Características das variáveis de <b>tipo inteiro</b>
<a href="#">locale.h</a>	Definições relacionadas com <b>localização</b>
<a href="#">math.h</a>	<b>Funções matemáticas</b>
<a href="#">setjmp.h</a>	Tratamento de <b>exceções</b>
<a href="#">signal.h</a>	<b>Saltos</b> não locais
<a href="#">stdarg.h</a>	Funções com um <b>número variado de argumentos</b>
<a href="#">stddef.h</a>	Definição de <b>tipos e macros standard</b>
<a href="#">stdio.h</a>	Funções de <b>entrada/saída (input/output)</b>
<a href="#">stdlib.h</a>	Ferramentas gerais; <b>gestão dinâmica</b> de memória
<a href="#">string.h</a>	Manipulação de <b>memória</b> e <b>strings</b>
<a href="#">time.h</a>	Gestão de <b>data</b> e <b>hora</b>

# Biblioteca Standard - 'assert.h' ('Prog34\_01.c')

- Neste ficheiro encontra-se basicamente a **macro** 'assert' para verificar a **existência de erros** durante a execução:

**assert** (**expr**);

- Esta **macro** é definida em **C** de forma a **parar o programa** caso a expressão '**expr**' corresponda a um valor falso.
- O seu uso torna mais fácil a **análise de erros** num programa.
- O programa ('Prog34\_01.c') exemplifica o seu uso no caso de termos um divisão por zero.

# Biblioteca Standard - 'errno.h' ('Prog34\_02.c')

- A maioria das funções da **biblioteca standard** gera **mensagens de erro** quando ocorre alguma anomalia. Com o intuito de gerir de forma coerente as mensagens de erro, o ficheiro '**errno.h**' introduz a variável '**errno**'.
- Este ficheiro define também **3 constantes** para diferentes erros:
  - **EDOM**: quando o **argumento** de uma função matemática está **fora do domínio**;
  - **EILSEQ**: quando há uma **sequência ilegal** de bytes;
  - **ERANGE**: quando o resultado é **demasiado grande**.
- Existe ainda a função '**perror**' que recebe como argumento uma '**string**' e imprime-a conjuntamente com a mensagem de erro.
- A mensagem de erro pode ser obtida com '**stderr**'.

# Biblioteca Standard - 'setjmp.h' e 'signal.h' ( 'Prog34\_04.c' ) e ( 'Prog34\_05.c' )

- No ficheiro '**setjmp.h**' são declarados os mecanismos para executarmos 'saltos não locais', isto é, interrupções do fluxo normal do programa.
- A utilização dos processos aqui definidos são basicamente úteis para o tratamento de erros e interrupções.
- Funciona como um espécie de **goto** mas para fora da função em que é executado.
- '**setjmp**' marca o **ponto de retorno** e '**longjmp**' restaura a **execução** para o ponto marcado ( '**Prog34\_04.c**' ).
- No ficheiro '**signal.h**' são declarados os mecanismos para lidar com 'sinais', isto é, ocorrências excepcionais ( '**Prog34\_05.c**' ).

# Biblioteca Standard - 'stddef.h' e 'stdarg.h'

- Em '**stddef.h**' encontram-se definidos algumas definições de caracter genérico. Como sejam:
  - 1 '**size\_t**': inteiro positivo resultante de '**sizeof**';
  - 2 '**NULL**': constante nula para ponteiros (**(void \*) 0**);
  - 3 '**ptrdiff\_t**': tipo inteiro com sinal resultante da subtracção de ponteiros;
  - 4 '**wchar\_t**': tipo inteiro que pode representar '**wide characters**'.
  - 5 '**offsetof (tipo, membro)**': retorna a distância em bytes a que um membro de uma estrutura se encontra do início da mesma. Se a estrutura for um '**bit field**' o resultado é indeterminado. Ver **Prog14\_07.c**.
- Em '**stdarg.h**' são definidos os tipos e funções usados em funções com um número variado de argumentos (de que já falámos anteriormente). Ver **Prog29\_01e2.c**.



# Biblioteca Standard - 'ctype.h' ('Prog35\_01.c')

- As **funções** aqui definidas são **funções de teste** que se aplicam a **'char'** (**caracteres**) e retornam **'0'** se o resultado for **falso**.
- Em certos casos a sua **implementação** pode ser feita por **macros**.
- São exemplo destas **funções 'isalpha'**, que testa se é **letra**, ou **'isdigit'**, que testa se é **número**.
- Existem ainda duas **funções** que fazem **conversões** entre **minúsculas** e **maiúsculas**.
  - **'tolower'**: converte para minúsculas;
  - **'toupper'**: converte para maiúsculas.

# Biblioteca Standard - 'limits.h' e 'float.h' ( 'Prog36\_01.c' ) e ( 'Prog36\_02.c' )

- Em '**limits.h**' estão definidos os **limites** dos vários tipos de '**inteiros**' bem como o **número de bits** contido num '**char**' ( '**Prog36\_01.c**' ).
- Em '**float.h**' encontram-se definidos os **valores limites** para os diferentes tipos de '**floating point**'.
- Igualmente são definidas macros com o **acrécimo** mais pequeno que se pode dar à unidade e o **número de dígitos** significativos;
- Nota, as definições de '**float.h**' poderão estar também no ficheiro '**values.h**'.

# Biblioteca Standard - 'stdio.h'

- Este ficheiro tem as definições das funções que permitem a um programa **interactuar com o exterior**: **escrever** e **ler** no **ecran** e ou em **ficheiros**, escrita de erros, etc.
- Como já vimos associados a **cada programa** estão **três canais**:
  - 1 **stdin**: canal de **leitura** do **terminal**;
  - 2 **stdout**: canal de **escrita** no **terminal**;
  - 3 **stderr**: canal de **escrita** de erros, apontado para o **terminal**.
- São definidos três tipos:
  - 1 **fpos\_t**: usada por '**fsetpos**' para especificar a posição numa file;
  - 2 **size\_t**: idêntico a '**size\_t**' de '**stddef.h**';
  - 3 **FILE**: estrutura destinada a conter toda a informação referente aos canais de comunicação, '**streams**'.

## Biblioteca Standard - 'stdio.h' ('Prog39\_01.c')

Estão ainda definidas as seguintes **constantes**:

- **BUFSIZ**: tamanho da zona de memória ('**buffer**') para as operações de entrada e saída. Pelo menos de 256 bytes.
- **EOF**: indica o fim de um ficheiro. Toma tipicamente o valor '**-1**'.
- **NULL**: valor nulo para ponteiros: '(void \*)**0**' ('**libio.h**', em **gcc**).
- **FILENAME\_MAX**: número máximo de caracteres de que o nome de um ficheiro pode ser composto.
- **FOPEN\_MAX**: número mínimo de '**streams**' abertos ao mesmo tempo que a implementação garante poder estar abertos simultaneamente. Depende da implementação. Pelo menos, '**8**'.
- **L\_tmpnam**: tamanho máximo em caracteres que o nome de um ficheiro temporário pode conter. Este valor está relacionado com o resultado da função '**tmpnam**'. Depende da implementação.
- **TMP\_MAX**: número máximo de nomes de ficheiros temporários distintos que podem ser gerados pelo sistema. Depende da implementação. Tem de ser superior a '**25**'.

# Biblioteca Standard - 'stdio.h'

- A **passagem de dados** de (ou para) um **ficheiro** pode ser feita das seguintes maneiras:
  - 1 **non buffered**: enviando **caracter a caracter**;
  - 2 **line buffered**: armazenar os dados em memória até uma **mudança de linha**;
  - 3 **fully buffered**: juntando os caracteres em **blocos de tamanho variável** antes de enviar.
- Normalmente, um **ficheiro** recém aberto usa o **último método** (por ser o mais eficiente); para **terminal** usualmente é usado o **segundo método**.
- É possível forçar um **determinado método** com a função '**setvbuf**' que usa como argumento:
  - 1 **\_IOFBF**: indica o método **fully buffered** com o valor '**0**';
  - 2 **\_IOLBF**: indica o método **line buffered** com o valor '**1**';
  - 3 **\_IONBF**: indica o método **non buffered** com o valor '**2**';

# Biblioteca Standard - 'stdio.h'

- Estão ainda definidas as **constantes** relacionadas com o **modo** como o **posicionamento** é feitos em **ficheiros**:

- 1 **SEEK\_SET**: é feito a partir do **início**. Valor '**0**';
- 2 **SEEK\_CUR**: é feito a partir da **posição actual**. Valor '**1**';
- 3 **SEEK\_END**: é feito a partir do **fim**. Valor '**2**';

O **posicionamento** é feito por **funções** como **fseek**.

- As funções declaradas em '**stdio.h**' podem ser agrupadas nas seguintes categorias:

- 1 **Abertura e fecho** de ficheiros;
- 2 **Posicionamento** num ficheiro;
- 3 Tratamento de **erros**;
- 4 Leitura e escrita de **caracteres**;
- 5 Leitura e escrita **formatada**;
- 6 Leitura e escrita **binária**;
- 7 **Eliminação** ou **mudança de nome** de ficheiros.

# Biblioteca Standard - 'stdio.h'

## Abertura e Fecho de Ficheiros

- **FILE \*fopen** (**const char \*fname**, **const char \*amode**);  
**abrir** o ficheiro '**fname**', sendo o modo de abertura:
  - 1 '**r**': Modo **leitura**. Posiciona-se no **início**.
  - 2 '**r+**': Modo **leitura** e **escrita**. Posiciona-se no **início**.
  - 3 '**w**': Modo **escrita**. Posiciona-se no **início**.
  - 4 '**w+**': Modo **leitura** e **escrita**. Posiciona-se no **início**.
  - 5 '**a**': Modo **escrita**. Posiciona-se no **final**.
  - 6 '**a+**': Modo **leitura** e **escrita**. Posiciona-se no **final**.
  - 7 '**b**': Dados em modo **binário**.
  - 8 '**t**': Dados em modo **texto**.
- **int fclose** (**FILE \*stream**);  
**fecha** um ficheiro previamente aberto.
- **int fflush** (**FILE \*stream**);  
**envia** para o destino **todos os dados** armazenados na zona de memória associada ao '**stream**'.

# Biblioteca Standard - 'stdio.h' ('Prog39\_02.c')

## Posicionamento num Ficheiro

Quando se abre um **ficheiro** para **escrita** e/ou **leitura**, pode ler-se e escrever-se em qualquer posição. As **função** de posicionamento são:

- **int fseek** (**FILE \*stream**, **long int pos**, **int ponto**);  
coloca o cursor na posição '**pos**' face ao ponto escolhido. Ponto pode ser '**SEEK\_SET**', '**SEEK\_CUR**' e '**SEEK\_END**'.
- **long int ftell** (**FILE \*stream**);  
retorna a **posição actual** no ficheiro.
- **void rewind** (**FILE \*stream**);  
reposiciona o cursor no **início** do ficheiro.
- **int fgetpos** (**FILE \*stream**, **fpos\_t \*pos**);  
retorna a **posição**, por referência, actual no ficheiro ('**pos**').
- **int fsetpos** (**FILE \*stream**, **fpos\_t \*pos**);  
coloca o cursor na posição '**pos**'.



# Biblioteca Standard - 'stdio.h' ('Prog39\_02.c')

## Leitura e Escrita de Caracteres - I

Para ler ou escrever um **caracter** num '**stream**', temos as funções:

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

Lê o **caracter seguinte** do **stream** associado. Retorna um '**unsigned char**' moldado em '**int**'.

- **int getc** (**FILE \*stream**);

Idêntica a '**fgetc**' mas implementada em **macro**.

- **int getchar** (**void**);

Equivalente a '**fgetc (stdin);**'

# Biblioteca Standard - 'stdio.h' ('Prog39\_02.c')

## Leitura e Escrita de Caracteres - I

Para ler ou escrever 'strings' num 'stream', temos as funções:

- **char \*fgets** (**char \*str**, **int str\_len**, **FILE \*stream**);

Lê no máximo '**str\_len - 1**' caracteres e guarda-os na 'string' **str** colocando o terminador de 'string'.

Pára a leitura quando é encontrado o caracter '**nova linha**' ('\n' ou '**EOL**') ou o **final do ficheiro** ('**EOF**').

Quando chega ao fim do ficheiro retorna '**NULL**'.

- **char \*gets** (**char \*str**);

Equivalente a '**fgets**' do '**stdin**' mas **não testa** o tamanho. Por isso é considerada uma **função perigosa** a **não usar**.

# Biblioteca Standard - 'stdio.h' ('Prog39\_02.c')

## Leitura e Escrita de Caracteres - II

- **int fputc** (**int c**, **FILE \*stream**);  
Escreve o caracter 'c' depois do converter para 'unsigned char'.
- **int putc** (**int c**, **FILE \*stream**);  
Idêntica a 'fputc' mas implementada em **macro**.
- **int putchar** (**int c**);  
Equivalente a **fputc (c, stdout)**;
- **int fputs** (**const char \*str**, **FILE \*stream**);  
Copia 'str' para o **stream**.
- **int puts** (**const char \*str**);  
Equivalente a 'fputs (str, stout);'
- **int ungetc** (**int c**, **FILE \*stream**);  
Volta a "introduzir" o caracter **c** no **stream**, excepto se fôr o 'EOF'. Os sistemas devem permitir que, **pelo menos**, um caracter pode ser sujeito a esta operação.