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**".

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.

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**.

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.

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**".

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 intensão fazer aqui uma **descrição completa** de **todas** as funções definidas na **biblioteca standard**. Nem tal seria muito interessante.

Biblioteca Standard

- Não é nossa intensão 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.

Biblioteca Standard

- Não é nossa intensão 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.

Biblioteca Standard

- Não é nossa intensão 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.

Biblioteca Standard

- Não é nossa intensão 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
assert.h	**Diagnóstico**
ctype.h	Tratamento de **caracteres**
errno.h	Tratamento de **erros**
float.h	Características das variáveis de **tipo real**
limits.h	Características das variáveis de **tipo inteiro**
locale.h	Definições relacionadas com **localização**
math.h	**Funções matemáticas**
setjmp.h	Tratamento de **excepções**
signal.h	**Saltos** não locais
stdarg.h	Funções com um **número variado de argumentos**
stddef.h	Definição de **tipos e macros standard**
stdio.h	Funções de **entrada/saída** (**input/output**)
stdlib.h	Ferrramentas gerais; **gestão dinâmica** de memória
string.h	Manipulação de **memória** e **strings**
time.h	Gestão de **data** e **hora**

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**);

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.

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.

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**'.

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:

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**.

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.

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 '**stderror**'.

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.

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.

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.

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 executação** para o ponto marcado ('**Prog34_04.c**').

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 executação** 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:

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**';

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**);

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;

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**'.

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**.

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**.

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**.

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**.

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**.

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'**).

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**'.

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 **acréscimo** mais pequeno que se pode dar à unidade e o **número de dígitos** significativos;

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 **acréscimo** 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.

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**:

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**.

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:

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**:

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.

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**'.

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**).

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.

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**'.

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.

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:

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.

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**.

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:

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**:

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**.

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:

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;

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;

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**;

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**;

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**;

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**;

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:

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**.

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**.

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**.

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**.

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.

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:

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**'.

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.

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.

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**').

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:

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**'.

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**.

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);**

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 returna '**NULL**'.

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 returna '**NULL**'.
- **char *gets (char *str);**

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 returna '**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**'.

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**.

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);**

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**.

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);**'

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.