

19ª Aula - Biblioteca Standard (II)

Programação Mestrado em Engenharia Física Tecnológica

Samuel M. Eleutério
sme@tecnico.ulisboa.pt

Departamento de Física
Instituto Superior Técnico
Universidade de Lisboa

Biblioteca Standard - 'stdio.h'

Tratamento de Erros

Aqui são apresentadas as **funções** de **tratamento de erros**. Note-se que os erros poderão não ser directamente resultantes de operações sobre ficheiros.

- **int feof** (**FILE** ***stream**);

Retorna um valor **não nulo** se tiver sido fixado o indicador de **fim de ficheiro**.

- **int ferror** (**FILE** ***stream**);

Retorna um valor **não nulo** se tiver sido fixado o indicador de erro para o '**stream**'.

- **void perror** (**const char** ***str**);

Ver função definida em '**errno.h**'

Biblioteca Standard - 'stdio.h'

Leitura e Escrita de Formatação (I)

- Todas as **funções** (com exceção da iniciadas por '**v**'), têm um **número variável de argumentos** e retornam o **número de elementos** lidos ou escritos **com sucesso**.
- A sequência de formatação pode incluir
 - 1 **caracteres especiais** ('\n', '\t', '\r', '\b', '\a', etc.);
 - 2 **sequências especiais** ('\\', '%%', '\"').
- A inserção de **variáveis** é feita por:
 - 1 Começa com o símbolo '%';
 - 2 Se necessário, seguem-se **caracteres de controle** e **identificadores** que alteram as características da conversão;
 - 3 **Caracter de conversão**.
- No slide seguinte é apresentada a **tabela caracteres** de '**printf**'. Uma tabela análoga existe para '**scanf**' com poucas diferenças em relação a esta.



Exemplos de Caracteres de Conversão de 'printf'

| Conversão | Significado |
|---------------------|---|
| <code>%d, %i</code> | Inteiro decimal |
| <code>%o</code> | Inteiro octal (sem '0' inicial) |
| <code>%x, %X</code> | Inteiro hexadecimal (sem '0x' ou '0X') |
| <code>%u</code> | Inteiro sem sinal (unsigned) |
| <code>%c</code> | Caracter (convertido 'unsigned char') |
| <code>%s</code> | String (char *). Imprime até '\0' |
| <code>%f</code> | real na forma ' <code>[-]mmm.ddd</code> ' |
| <code>%e, %E</code> | real na forma ' <code>[-]m.ddde±xx</code> ' ou ' <code>[-]m.dddE±xx</code> ' |
| <code>%g, %G</code> | Basicamente o mais curto de 'f' e 'e' |
| <code>%p</code> | Ponteiro (void *) |
| <code>%n</code> | Não escreve. Retorna o número de caracteres escritos até esse ponto. É lido por referência para um 'int'. |
| <code>%%</code> | Nenhum argumento é convertido; imprime '%' |

Nota: Para mais indicações ver, por exemplo, em '[C Library](#)'.

Biblioteca Standard - 'stdio.h' ('Prog39_03.c')

Leitura e Escrita de Formatação (II)

- **int fprintf** (**FILE** ***stream**, **const char** ***format**, ...);
Escreve em '**stream**' os dados **formatados** em '**format**' e descritos em '...'.

- **int printf** (**const char** ***format**, ...);
Equivalente a **fprintf** mas escreve no canal '**stdout**'.
- **int sprintf** (**char** ***str**, **const char** ***format**, ...);
Equivalente a **fprintf** mas escreve na string '**str**'.
- **int fscanf** (**FILE** ***stream**, **const char** ***format**, ...);
Lê de '**stream**' os dados **formatados** em '**format**' e recebe-os, por referência, em '...'.

- **int scanf** (**const char** ***format**, ...);
Equivalente a **fscanf** mas lê do canal '**stdin**'.
- **int sscanf** (**char** ***str**, **const char** ***format**, ...);
Equivalente a **fscanf** mas lê da string '**str**'.

Biblioteca Standard - 'stdio.h'

Leitura e Escrita de Formatada (III)

- **int** **vfprintf** (**FILE** ***stream**, **const char** ***format**, **va_list** **vlist**);
Equivalente **fprintf** mas é usada uma **va_list**. Após o seu uso deve ser feita uma chamada explícita de '**va_end** (**vlist**)'.
- **int** **vprintf** (**const char** ***format**, **va_list** **vlist**);
Equivalente a **vfprintf** mas escreve no canal '**stdout**'.
- **int** **vsprintf** (**char** ***str**, **const char** ***format**, **va_list** **vlist**);
Equivalente a **vfprintf** mas escreve na string '**str**'.
- **int** **vfscanf** (**FILE** ***stream**, **const char** ***format**, **va_list** **vlist**);
Equivalente **fscanf** mas é usada uma **va_list** como em **vfprintf**.
- **int** **vscanf** (**const char** ***format**, **va_list** **vlist**);
Equivalente a **vfscanf** mas lê do canal '**stdin**'.
- **int** **vsscanf** (**char** ***str**, **const char** ***format**, **va_list** **vlist**);
Equivalente a **vfscanf** mas lê da string '**str**'.

Biblioteca Standard - 'stdio.h' ('Prog39_04' e '05')

Leitura e Escrita Binária

- As **funções** aqui abordadas permitem **ler** e **escrever** em **ficheiros binários**, não havendo lugar a formatação. Em muitos casos, o seu uso torna os **programas** muito mais **eficientes**.
- O **posicionamento** nesses ficheiros pode ser feito com as **funções de posicionamento** já vistas (por exemplo, **fseek**).
- O **espaço ocupado** pelas variáveis nestes ficheiros corresponde ao espaço que ocupam em **memória** (por exemplo, um **'float'** ocupa **4 bytes**, um **'double'** **8 bytes**, etc.).
- Há duas **funções** para a **leitura** e **escrita**:

1 **size_t fread** (**void *ptr**, **size_t size**, **size_t qt**, **FILE *stream**);

2 **size_t fwrite** (**void *ptr**, **size_t size**, **size_t qt**, **FILE *stream**);

Em que **ptr** é o **ponteiro** para o início dos elementos a escrever (ler), **'size'** é **tamanho** em bytes de cada elemento a escrever (ler), **'qt'** o **número** de elementos a escrever (ler) e **'stream'** é o **canal** de escrita (leitura).

Biblioteca Standard - 'stdio.h'

'rename' e 'remove'

Finalmente, existem ainda duas funções para lidar com ficheiros:

- **int rename** (**const char** ***fnome_inicial**, **const char** ***fnome_final**);

Altera o nome do ficheiro de '**fnome_inicial**' para '**fnome_final**'. Dependendo do **sistema operativo**, o nome também pode conter a **directaria**. Se '**fnome_final**' **já existe** o resultado depende também do **sistema operativo**.

Se a **operação** tiver **sucesso**, retorna '**0**', senão retorna '**-1**'.

- **int remove** (**const char** ***fnome**);

Elimina o **ficheiro** (ou o **directório**, no caso da implementação o permitir).

Se a **operação** tiver **sucesso**, retorna '**0**', senão retorna '**-1**'.

Biblioteca Standard - 'math.h' ('Prog40_01.c')

- Em '**math.h**' encontram-se definidas as **funções matemáticas**.
- Algumas **constantes matemáticas**:
 - **M_PI**
 - **M_E**
 - etc.
- E ainda as constantes:
 - '**HUGE_VAL**'
 - '**INFINITY**'
 - '**NAN**'.
- As **funções**, aqui definidas, podem ser agrupadas em:
 - 1 Funções **trigonométricas** (e inversas);
 - 2 Funções **hiperbólicas** (e inversas);
 - 3 Funções **exponencial** e **logaritmo**;
 - 4 Outras funções: '**pow**', '**sqr**', '**floor**', '**ceil**', ...

Biblioteca Standard - '**complex.h**' ('Prog40_02.c')

- Em '**complex.h**' (acrescentada na revisão C99 do '**C**') são definidas as **funções** que manipulam **números complexos**.
- O tipo associado é '**_Complex**' (ou '**complex**'). Exemplo:
double _Complexo z1, z2, z3 ;
- A representação de '**i**' é feita com a macro '**_Complex_I**' (ou '**I**'). Dado que '**I**' pode ser inadvertidamente posto como variável, é, muitas vezes, aconselhável a sua remoção '**#undef I**'.
- Pode obter-se a **parte real** ou a **parte imaginária** do complexo, respectivamente, com as funções '**creal**' ou '**cimag**'.
- São ainda definidas as funções '**conj**' (**complexo conjugado**), '**cabs**' (**módulo**) e '**carg**' (**ângulo no plano complexo**).
- De um modo geral as **funções reais** têm a sua extensão para **complexos**. Ela é feita usando o prefixo '**c**'.
- Para os tipos **float** e **long double** deve acrescentar-se o sufixo '**f**' ou '**l**', respectivamente.