

20ª Aula - Biblioteca Standard (III)

Programação Mestrado em Engenharia Física Tecnológica

Samuel M. Eleutério
sme@tecnico.ulisboa.pt

Departamento de Física
Instituto Superior Técnico
Universidade de Lisboa

Biblioteca Standard - 'locale.h'

('Prog37_01.c')

- Em 'locale.h' encontram-se definidas as **propriedades** relativas às **localizações geográficas**.

Biblioteca Standard - 'locale.h' ('Prog37_01.c')

- Em 'locale.h' encontram-se definidas as **propriedades** relativas às **localizações geográficas**.
- As constantes definidas em 'locale' iniciam-se por 'LC_'.

Biblioteca Standard - 'locale.h' ('Prog37_01.c')

- Em 'locale.h' encontram-se definidas as **propriedades** relativas às **localizações geográficas**.
- As constantes definidas em 'locale' iniciam-se por 'LC_'.
- '**struct lconv**' contém as informações sobre a **localização**.

Biblioteca Standard - 'locale.h'

('Prog37_01.c')

- Em 'locale.h' encontram-se definidas as **propriedades** relativas às **localizações geográficas**.
- As constantes definidas em 'locale' iniciam-se por 'LC_'.
- 'struct lconv' contém as informações sobre a **localização**.
- Com a função **setlocale** de protótipo:

Biblioteca Standard - 'locale.h'

('Prog37_01.c')

- Em 'locale.h' encontram-se definidas as **propriedades** relativas às **localizações geográficas**.
- As constantes definidas em 'locale' iniciam-se por 'LC_'.
- 'struct lconv' contém as informações sobre a **localização**.
- Com a função **setlocale** de protótipo:

```
char *setlocale (int category, const char *locale);
```

especifica-se as características do ambiente do programa.
- Para a informação contida na estrutura 'struct lconv' usamos a função 'localeconv' de protótipo:

Biblioteca Standard - 'locale.h'

('Prog37_01.c')

- Em 'locale.h' encontram-se definidas as **propriedades** relativas às **localizações geográficas**.
- As constantes definidas em 'locale' iniciam-se por 'LC_'.
- '**struct lconv**' contém as informações sobre a **localização**.
- Com a função **setlocale** de protótipo:

```
char *setlocale (int category, const char *locale);
```

especifica-se as características do ambiente do programa.

- Para a informação contida na estrutura '**struct lconv**' usamos a função '**localeconv**' de protótipo:

```
struct lconv *localeconv (void);
```

Biblioteca Standard - 'time.h' ('Prog38_01e02.c')

- Em 'time.h' são definidas:

Biblioteca Standard - 'time.h' ('Prog38_01e02.c')

■ Em 'time.h' são definidas:

- 1 A constante '**CLOCKS_PER_SEC**' que está relacionada com o número de **batimentos do CPU** por segundo.

Biblioteca Standard - 'time.h' ('Prog38_01e02.c')

- Em 'time.h' são definidas:
 - 1 A constante '**CLOCKS_PER_SEC**' que está relacionada com o número de **batimentos do CPU** por segundo.
 - 2 Dois tipos correspondentes a **long**: '**clock_t**' e '**time_t**'.

Biblioteca Standard - 'time.h' ('Prog38_01e02.c')

- Em '**time.h**' são definidas:
 - 1 A constante '**CLOCKS_PER_SEC**' que está relacionada com o número de **batimentos do CPU** por segundo.
 - 2 Dois tipos correspondentes a **long**: '**clock_t**' e '**time_t**'.
 - 3 Uma estrutura '**struct tm**' que contém informação sobre o tempo.

Biblioteca Standard - 'time.h' ('Prog38_01e02.c')

- Em '**time.h**' são definidas:
 - 1 A constante '**CLOCKS_PER_SEC**' que está relacionada com o número de **batimentos do CPU** por segundo.
 - 2 Dois tipos correspondentes a **long**: '**clock_t**' e '**time_t**'.
 - 3 Uma estrutura '**struct tm**' que contém informação sobre o tempo.
- A função '**clock**' é a única que acede ao **relógio interno** e permite-nos calcular o **tempo de CPU**. A sua conversão para segundos é feita com a constante '**CLOCKS_PER_SEC**'.

Biblioteca Standard - 'time.h' ('Prog38_01e02.c')

- Em '**time.h**' são definidas:
 - 1 A constante '**CLOCKS_PER_SEC**' que está relacionada com o número de **batimentos do CPU** por segundo.
 - 2 Dois tipos correspondentes a **long**: '**clock_t**' e '**time_t**'.
 - 3 Uma estrutura '**struct tm**' que contém informação sobre o tempo.
- A função '**clock**' é a única que acede ao **relógio interno** e permite-nos calcular o **tempo de CPU**. A sua conversão para segundos é feita com a constante '**CLOCKS_PER_SEC**'.
- As restantes **funções** dizem respeito ao **tempo** (date e hora).

Biblioteca Standard - 'time.h' ('Prog38_01e02.c')

- O **tempo** pode ser representados por dois tipos:

Biblioteca Standard - 'time.h' ('Prog38_01e02.c')

- O **tempo** pode ser representados por dois tipos:
 - 1 O **tempo de calendário**, em geral, representado pelo número de segundos desde o dia 1 de Janeiro de 1970 e é codificado de acordo com **UTC** (**Universal Time Coordinated**), a norma que substituiu o **GMT** (**Tempo Médio de Greenwich**).

Biblioteca Standard - 'time.h' ('Prog38_01e02.c')

- O **tempo** pode ser representados por dois tipos:
 - 1 O **tempo de calendário**, em geral, representado pelo número de segundos desde o dia 1 de Janeiro de 1970 e é codificado de acordo com **UTC** (**Universal Time Coordinated**), a norma que substituiu o **GMT** (**Tempo Médio de Greenwich**).
 - 2 A outra representação encontra-se em '**struct tm**'.

Biblioteca Standard - 'time.h' ('Prog38_01e02.c')

- O **tempo** pode ser representados por dois tipos:
 - 1 O **tempo de calendário**, em geral, representado pelo número de segundos desde o dia 1 de Janeiro de 1970 e é codificado de acordo com **UTC** (**Universal Time Coordinated**), a norma que substituiu o **GMT** (**Tempo Médio de Greenwich**).
 - 2 A outra representação encontra-se em '**struct tm**'.
- Diversas **funções** lidam com estas duas representações e permitem a **conversão** de uma na outra. Por exemplo, '**localtime**' converte o **tempo de calendário** para '**struct tm**':

```
struct tm *localtime (const time_t *time)
```

Biblioteca Standard - 'time.h' ('Prog38_01e02.c')

- O **tempo** pode ser representados por dois tipos:
 - 1 O **tempo de calendário**, em geral, representado pelo número de segundos desde o dia 1 de Janeiro de 1970 e é codificado de acordo com **UTC (Universal Time Coordinated)**, a norma que substituiu o **GMT (Tempo Médio de Greenwich)**.
 - 2 A outra representação encontra-se em '**struct tm**'.
- Diversas **funções** lidam com estas duas representações e permitem a **conversão** de uma na outra. Por exemplo, '**localtime**' converte o **tempo de calendário** para '**struct tm**':

```
struct tm *localtime (const time_t *time)
```
- Note-se ainda que as funções do **tempo** (de calendário) estão articuladas com as **características regionais** definidas em '**locale.h**'.

Biblioteca Standard - 'time.h' ('Prog38_01e02.c')

- O **tempo** pode ser representados por dois tipos:
 - 1 O **tempo de calendário**, em geral, representado pelo número de segundos desde o dia 1 de Janeiro de 1970 e é codificado de acordo com **UTC (Universal Time Coordinated)**, a norma que substituiu o **GMT (Tempo Médio de Greenwich)**.
 - 2 A outra representação encontra-se em '**struct tm**'.
- Diversas **funções** lidam com estas duas representações e permitem a **conversão** de uma na outra. Por exemplo, '**localtime**' converte o **tempo de calendário** para '**struct tm**':

```
struct tm *localtime (const time_t *time)
```
- Note-se ainda que as funções do **tempo** (de calendário) estão articuladas com as **características regionais** definidas em '**locale.h**'.
- Para se imprimirem os **valores do tempo** a partir de '**struct tm**' usa-se a função '**strftime**'.

Biblioteca Standard - 'string.h' (I) ('Prog11_01.c')

Já anteriormente foram referidas algumas das **funções** incluídas em 'string.h'. Assim, far-se-á simplesmente a sistematização das funções de manipulação de 'strings'. É necessário garantir que **dest** tem **tamanho suficiente** para a operação. As funções que se seguem retornam um ponteiro 'char *' para 'dest'.

Biblioteca Standard - 'string.h' (I) ('Prog11_01.c')

Já anteriormente foram referidas algumas das **funções** incluídas em 'string.h'. Assim, far-se-á simplesmente a sistematização das funções de manipulação de 'strings'. É necessário garantir que **dest** tem **tamanho suficiente** para a operação. As funções que se seguem retornam um ponteiro 'char *' para 'dest'.

- `char *strcat (char *dest, const char *orig);`
Acrescenta 'orig' a seguir a 'dest'.

Biblioteca Standard - 'string.h' (I) ('Prog11_01.c')

Já anteriormente foram referidas algumas das **funções** incluídas em 'string.h'. Assim, far-se-á simplesmente a sistematização das funções de manipulação de 'strings'. É necessário garantir que **dest** tem **tamanho suficiente** para a operação. As funções que se seguem retornam um ponteiro 'char *' para 'dest'.

- **char *strcat** (**char *dest**, **const char *orig**);

Acrescenta 'orig' a seguir a 'dest'.

- **char *strncat** (**char *dest**, **const char *orig**, **size_t len**);

Idêntica a 'strcat', mas acrescenta **no máximo 'len'** caracteres.

Biblioteca Standard - 'string.h' (I) ('Prog11_01.c')

Já anteriormente foram referidas algumas das **funções** incluídas em 'string.h'. Assim, far-se-á simplesmente a sistematização das funções de manipulação de 'strings'. É necessário garantir que **dest** tem **tamanho suficiente** para a operação. As funções que se seguem retornam um ponteiro 'char *' para 'dest'.

- **char *strcat** (**char *dest**, **const char *orig**);
Acrescenta 'orig' a seguir a 'dest'.
- **char *strncat** (**char *dest**, **const char *orig**, **size_t len**);
Idêntica a 'strcat', mas acrescenta **no máximo 'len'** caracteres.
- **char *strcpy** (**char *dest**, **const char *orig**);
Copia a string 'orig' para 'dest'.

Biblioteca Standard - 'string.h' (I) ('Prog11_01.c')

Já anteriormente foram referidas algumas das **funções** incluídas em 'string.h'. Assim, far-se-á simplesmente a sistematização das funções de manipulação de 'strings'. É necessário garantir que **dest** tem **tamanho suficiente** para a operação. As funções que se seguem retornam um ponteiro 'char *' para 'dest'.

- **char *strcat** (**char *dest**, **const char *orig**);
Acrescenta 'orig' a seguir a 'dest'.
- **char *strncat** (**char *dest**, **const char *orig**, **size_t len**);
Idêntica a 'strcat', mas acrescenta **no máximo 'len'** caracteres.
- **char *strcpy** (**char *dest**, **const char *orig**);
Copia a string 'orig' para 'dest'.
- **char *strncpy** (**char *dest**, **const char *orig**, **size_t len**);
Idêntica a 'strcpy', mas copia **no máximo 'len'** caracteres.

Biblioteca Standard - 'string.h' (I) ('Prog11_01.c')

Já anteriormente foram referidas algumas das **funções** incluídas em '**string.h**'. Assim, far-se-á simplesmente a sistematização das funções de manipulação de '**strings**'. É necessário garantir que **dest** tem **tamanho suficiente** para a operação. As funções que se seguem retornam um ponteiro '**char ***' para '**dest**'.

- **char *strcat** (**char *dest**, **const char *orig**);
Acrescenta '**orig**' a seguir a '**dest**'.
- **char *strncat** (**char *dest**, **const char *orig**, **size_t len**);
Idêntica a '**strcat**', mas acrescenta **no máximo 'len'** caracteres.
- **char *strcpy** (**char *dest**, **const char *orig**);
Copia a string '**orig**' para '**dest**'.
- **char *strncpy** (**char *dest**, **const char *orig**, **size_t len**);
Idêntica a '**strcpy**', mas copia **no máximo 'len'** caracteres.
- **size_t strlen** (**const char *str**);
Retorna o número de caracteres da string '**str**' (**comprimento**).

Biblioteca Standard - 'string.h' (II) ('Prog42_01.c')

Para comparação e pesquisa em '**strings**', têm-se:

Biblioteca Standard - 'string.h' (II) ('Prog42_01.c')

Para comparação e pesquisa em 'strings', têm-se:

- `int strcmp (const char *str1, const char *str2);`
Retorna '0' se forem **lexograficamente iguais**, **positivo** se 'str1' for **maior** que 'str2' e **negativo** no caso contrário.

Biblioteca Standard - 'string.h' (II) ('Prog42_01.c')

Para comparação e pesquisa em 'strings', têm-se:

- **int strcmp** (**const char *str1**, **const char *str2**);
Retorna '0' se forem **lexograficamente iguais**, **positivo** se 'str1' for **maior** que 'str2' e **negativo** no caso contrário.
- **int strncmp** (**const char *str1**, **const char *str2**, **size_t len**);
Idêntica a 'strcmp', mas compara **no máximo 'len'** caracteres.

Biblioteca Standard - 'string.h' (II) ('Prog42_01.c')

Para comparação e pesquisa em 'strings', têm-se:

- **int strcmp** (**const char *str1**, **const char *str2**);
Retorna '0' se forem **lexograficamente iguais**, **positivo** se 'str1' for **maior** que 'str2' e **negativo** no caso contrário.
- **int strncmp** (**const char *str1**, **const char *str2**, **size_t len**);
Idêntica a 'strcmp', mas compara **no máximo 'len'** caracteres.
- **char *strchr** (**const char *str**, **int c**);
Procura a primeira ocorrência de 'c' em 'str' e retorna o **ponteiro** para essa posição, ou '**NULL**' no caso contrário.

Biblioteca Standard - 'string.h' (II) ('Prog42_01.c')

Para comparação e pesquisa em 'strings', têm-se:

- **int strcmp** (**const char *str1**, **const char *str2**);
Retorna '0' se forem **lexograficamente iguais**, **positivo** se 'str1' for **maior** que 'str2' e **negativo** no caso contrário.
- **int strncmp** (**const char *str1**, **const char *str2**, **size_t len**);
Idêntica a 'strcmp', mas compara **no máximo 'len'** caracteres.
- **char *strchr** (**const char *str**, **int c**);
Procura a primeira ocorrência de 'c' em 'str' e retorna o **ponteiro** para essa posição, ou 'NULL' no caso contrário.
- **char *strrchr** (**const char *str**, **int c**);
Idêntica a 'strchr' mas procura **do fim para o princípio**.

Biblioteca Standard - 'string.h' (II) ('Prog42_01.c')

Para comparação e pesquisa em 'strings', têm-se:

- **int strcmp** (**const char *str1**, **const char *str2**);
Retorna '0' se forem **lexograficamente iguais**, **positivo** se 'str1' for **maior** que 'str2' e **negativo** no caso contrário.
- **int strncmp** (**const char *str1**, **const char *str2**, **size_t len**);
Idêntica a 'strcmp', mas compara **no máximo 'len'** caracteres.
- **char *strchr** (**const char *str**, **int c**);
Procura a primeira ocorrência de 'c' em 'str' e retorna o **ponteiro** para essa posição, ou 'NULL' no caso contrário.
- **char *strrchr** (**const char *str**, **int c**);
Idêntica a 'strchr' mas procura **do fim para o princípio**.
- **size_t strspn** (**const char *str**, **const char *skipset**);
Esta função retorna o **comprimento da substring** inicial que contém apenas caracteres de 'skipset'.

Biblioteca Standard - 'string.h' (III)

('Prog42_01e2.c')

- **size_t strcspn** (**const char *str**, **const char *stopset**);
É a inversa de '**strspn**'. Retorna o comprimento da *substring* inicial que **não contém** caracteres de '**stopset**'.

Biblioteca Standard - 'string.h' (III)

('Prog42_01e2.c')

- **size_t strcspn** (**const char *str**, **const char *stopset**);
É a inversa de '**strspn**'. Retorna o comprimento da *substring* inicial que **não contém** caracteres de '**stopset**'.
- **char *strpbrk** (**const char *str**, **const char *stopset**);
Está relacionada com '**strcspn**' mas retorna um **ponteiro** para o **primeiro character** da string que é membro de '**stopset**'.

Biblioteca Standard - 'string.h' (III)

('Prog42_01e2.c')

- **size_t strcspn** (**const char *str**, **const char *stopset**);
É a inversa de '**strspn**'. Retorna o comprimento da *substring* inicial que **não contém** caracteres de '**stopset**'.
- **char *strpbrk** (**const char *str**, **const char *stopset**);
Está relacionada com '**strcspn**' mas retorna um **ponteiro** para o **primeiro caracter** da string que é membro de '**stopset**'.
- **char *strstr** (**const char *str**, **const char *needle**);
Procura a **primeira** ocorrência de '**needle**' em '**str**' e retorna o **ponteiro** para essa posição.

Biblioteca Standard - 'string.h' (III)

('Prog42_01e2.c')

- **size_t strcspn** (**const char *str**, **const char *stopset**);
É a inversa de '**strspn**'. Retorna o comprimento da *substring* inicial que **não contém** caracteres de '**stopset**'.
- **char *strpbrk** (**const char *str**, **const char *stopset**);
Está relacionada com '**strcspn**' mas retorna um **ponteiro** para o **primeiro caracter** da string que é membro de '**stopset**'.
- **char *strstr** (**const char *str**, **const char *needle**);
Procura a **primeira** ocorrência de '**needle**' em '**str**' e retorna o **ponteiro** para essa posição.
- **char *strtok** (**const char *str**, **const char *delim**);
Busca conjuntos de caracteres de '**str**' que não contêm os caracteres incluídos em '**delim**'. Na primeira chamada recebe '**str**' como primeiro argumento, nas restantes deve receber um '**NULL**'. A cada chamada retorna a **palavra seguinte**. Acaba quando retorna um '**NULL**'.

Biblioteca Standard - 'string.h' (V) ('Prog42_03.c')

As **funções** que se seguem destinam-se à manipulação de **blocos de memória**. Estas funções actuam sobre **posições de memória** e obviamente **não são sensíveis** ao carácter de **fim de string '0'**.

Biblioteca Standard - 'string.h' (V) ('Prog42_03.c')

As **funções** que se seguem destinam-se à manipulação de **blocos de memória**. Estas funções actuam sobre **posições de memória** e obviamente **não são sensíveis** ao carácter de **fim de string '0'**.

- **void *memcpy (void *dest, const void *orig, size_t len);**
Copia '**len**' bytes de '**orig**' para '**dest**'. Os dois blocos não devem estar sobrepostos, pois, o resultado é indeterminado.

Biblioteca Standard - 'string.h' (V) ('Prog42_03.c')

As **funções** que se seguem destinam-se à manipulação de **blocos de memória**. Estas funções actuam sobre **posições de memória** e obviamente **não são sensíveis** ao carácter de **fim de string '0'**.

- **void *memcpy** (**void *dest**, **const void *orig**, **size_t len**);
Copia '**len**' bytes de '**orig**' para '**dest**'. Os dois blocos não devem estar sobrepostos, pois, o resultado é indeterminado.
- **void *memmove** (**void *dest**, **const void *orig**, **size_t len**);
Idêntica a '**memcpy**' mas, se houver sobreposição garante uma cópia correcta para as novas posições.

Biblioteca Standard - 'string.h' (V) ('Prog42_03.c')

As **funções** que se seguem destinam-se à manipulação de **blocos de memória**. Estas funções actuam sobre **posições de memória** e obviamente **não são sensíveis** ao carácter de **fim de string '0'**.

- **void *memcpy (void *dest, const void *orig, size_t len);**
Copia '**len**' bytes de '**orig**' para '**dest**'. Os dois blocos não devem estar sobrepostos, pois, o resultado é indeterminado.
- **void *memmove (void *dest, const void *orig, size_t len);**
Idêntica a '**memcpy**' mas, se houver sobreposição garante uma cópia correcta para as novas posições.
- **void *memchr (void *str, int c, size_t len);**
Procura em '**len**' bytes o valor '**c**' e retorna um ponteiro para ele.

Biblioteca Standard - 'string.h' (V) ('Prog42_03.c')

As **funções** que se seguem destinam-se à manipulação de **blocos de memória**. Estas funções actuam sobre **posições de memória** e obviamente **não são sensíveis** ao carácter de **fim de string '0'**.

- **void *memcpy** (**void *dest**, **const void *orig**, **size_t len**);
Copia '**len**' bytes de '**orig**' para '**dest**'. Os dois blocos não devem estar sobrepostos, pois, o resultado é indeterminado.
- **void *memmove** (**void *dest**, **const void *orig**, **size_t len**);
Idêntica a '**memcpy**' mas, se houver sobreposição garante uma cópia correcta para as novas posições.
- **void *memchr** (**void *str**, **int c**, **size_t len**);
Procura em '**len**' bytes o valor '**c**' e retorna um ponteiro para ele.
- **int memcmp** (**void *str1**, **void *str2**, **size_t len**);
Compara nos primeiros '**len**' bytes os valores de '**str1**' e '**str2**' e retorna o resultado usando a mesma regra que '**strcmp**'.

Biblioteca Standard - 'string.h' (V) ('Prog42_03.c')

As **funções** que se seguem destinam-se à manipulação de **blocos de memória**. Estas funções actuam sobre **posições de memória** e obviamente **não são sensíveis** ao carácter de **fim de string '0'**.

- **void *memcpy** (**void *dest**, **const void *orig**, **size_t len**);
Copia '**len**' bytes de '**orig**' para '**dest**'. Os dois blocos não devem estar sobrepostos, pois, o resultado é indeterminado.
- **void *memmove** (**void *dest**, **const void *orig**, **size_t len**);
Idêntica a '**memcpy**' mas, se houver sobreposição garante uma cópia correcta para as novas posições.
- **void *memchr** (**void *str**, **int c**, **size_t len**);
Procura em '**len**' bytes o valor '**c**' e retorna um ponteiro para ele.
- **int memcmp** (**void *str1**, **void *str2**, **size_t len**);
Compara nos primeiros '**len**' bytes os valores de '**str1**' e '**str2**' e retorna o resultado usando a mesma regra que '**strcmp**'.
- **void *memset** (**void *str**, **int c**, **size_t len**);
Preenche os primeiros '**len**' bytes de '**str**' com o valor '**c**'.

Biblioteca Standard - 'stdlib.h' ('Prog41_01.c')

Esta '**header file**' inclui um conjunto de **definições** de âmbito geral:

Biblioteca Standard - 'stdlib.h' ('Prog41_01.c')

Esta 'header file' inclui um conjunto de definições de âmbito geral:

- **EXIT_SUCCESS**: Com o valor '0', usada pela função 'exit'.

Biblioteca Standard - 'stdlib.h' ('Prog41_01.c')

Esta 'header file' inclui um conjunto de definições de âmbito geral:

- **EXIT_SUCCESS**: Com o valor '0', usada pela função 'exit'.
- **EXIT_FAILURE**: Com o valor '1', usada pela função 'exit'.

Biblioteca Standard - 'stdlib.h' ('Prog41_01.c')

Esta 'header file' inclui um conjunto de definições de âmbito geral:

- **EXIT_SUCCESS**: Com o valor '0', usada pela função 'exit'.
- **EXIT_FAILURE**: Com o valor '1', usada pela função 'exit'.
- **RAND_MAX**: O maior inteiro fornecido pela função 'rand'. O seu valor é, pelo menos '32767' ('SHRT_MAX') mas normalmente é 'INT_MAX'. Depende da implementação.

Biblioteca Standard - 'stdlib.h' ('Prog41_01.c')

Esta 'header file' inclui um conjunto de definições de âmbito geral:

- **EXIT_SUCCESS**: Com o valor '0', usada pela função 'exit'.
- **EXIT_FAILURE**: Com o valor '1', usada pela função 'exit'.
- **RAND_MAX**: O maior inteiro fornecido pela função 'rand'. O seu valor é, pelo menos '32767' ('SHRT_MAX') mas normalmente é 'INT_MAX'. Depende da implementação.
- **MB_CUR_MAX**: Representa o número máximo de bytes num character multibyte segundo a definição local.

Biblioteca Standard - 'stdlib.h' ('Prog41_01.c')

Esta 'header file' inclui um conjunto de definições de âmbito geral:

- **EXIT_SUCCESS**: Com o valor '0', usada pela função 'exit'.
- **EXIT_FAILURE**: Com o valor '1', usada pela função 'exit'.
- **RAND_MAX**: O maior inteiro fornecido pela função 'rand'. O seu valor é, pelo menos '32767' ('SHRT_MAX') mas normalmente é 'INT_MAX'. Depende da implementação.
- **MB_CUR_MAX**: Representa o número máximo de bytes num character multibyte segundo a definição local.
- Os tipo 'div_t' e 'ldiv_t' usados pelas funções 'div' e 'ldiv':

```
typedef struct {  
    int quot ;  
    int rem ;  
} div_t ;
```

```
typedef struct {  
    long int quot ;  
    long int rem ;  
} ldiv_t ;
```

Biblioteca Standard - 'stdlib.h' ('Prog41_01.c')

Esta 'header file' inclui um conjunto de definições de âmbito geral:

- **EXIT_SUCCESS**: Com o valor '0', usada pela função 'exit'.
- **EXIT_FAILURE**: Com o valor '1', usada pela função 'exit'.
- **RAND_MAX**: O maior inteiro fornecido pela função 'rand'. O seu valor é, pelo menos '32767' ('SHRT_MAX') mas normalmente é 'INT_MAX'. Depende da implementação.
- **MB_CUR_MAX**: Representa o número máximo de bytes num character multibyte segundo a definição local.
- Os tipos 'div_t' e 'ldiv_t' usados pelas funções 'div' e 'ldiv':

```
typedef struct {
    int quot ;
    int rem ;
} div_t ;

typedef struct {
    long int quot ;
    long int rem ;
} ldiv_t ;
```
- Os tipos 'size_t' e 'wchar_t', iguais aos definidos em 'stddef.h'

Biblioteca Standard - 'stdlib.h'

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

Biblioteca Standard - 'stdlib.h'

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

- 1 **Gestão dinâmica** de memória;

Biblioteca Standard - 'stdlib.h'

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

- 1 **Gestão dinâmica** de memória;
- 2 **Aritmética** inteira;

Biblioteca Standard - 'stdlib.h'

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

- 1 **Gestão dinâmica** de memória;
- 2 **Aritmética** inteira;
- 3 Gerador de números **pseudo-aleatórios**;

Biblioteca Standard - 'stdlib.h'

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

- 1 **Gestão dinâmica** de memória;
- 2 **Aritmética** inteira;
- 3 Gerador de números **pseudo-aleatórios**;
- 4 **Ordenação** e **busca**;

Biblioteca Standard - 'stdlib.h'

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

- 1 **Gestão dinâmica** de memória;
- 2 **Aritmética** inteira;
- 3 Gerador de números **pseudo-aleatórios**;
- 4 **Ordenação** e **busca**;
- 5 **Conversão** de cadeias de **caracteres**;

Biblioteca Standard - 'stdlib.h'

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

- 1 **Gestão dinâmica** de memória;
- 2 **Aritmética** inteira;
- 3 Gerador de números **pseudo-aleatórios**;
- 4 **Ordenação** e **busca**;
- 5 **Conversão** de cadeias de **caracteres**;
- 6 Controle de **saída** do programa.

Biblioteca Standard - 'stdlib.h'

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

- 1 **Gestão dinâmica** de memória;
- 2 **Aritmética** inteira;
- 3 Gerador de números **pseudo-aleatórios**;
- 4 **Ordenação** e **busca**;
- 5 **Conversão** de cadeias de **caracteres**;
- 6 Controle de **saída** do programa.
- 7 Comunicação com o **ambiente**;

Biblioteca Standard - 'stdlib.h'

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

- 1 **Gestão dinâmica** de memória;
- 2 **Aritmética** inteira;
- 3 Gerador de números **pseudo-aleatórios**;
- 4 **Ordenação** e **busca**;
- 5 **Conversão** de cadeias de **caracteres**;
- 6 Controle de **saída** do programa.
- 7 Comunicação com o **ambiente**;
- 8 Funções envolvendo **caracteres** e **strings multibyte**;

Biblioteca Standard - 'stdlib.h' ('Prog12_02.c')

Gestão Dinâmica de Memória

As **funções** que se seguem destinam-se à reserva de **memória contígua**. No caso de **sucesso** retornam o **endereço de memória** do início do espaço reservado; no caso de erro retornam '**NULL**'.

Biblioteca Standard - 'stdlib.h' ('Prog12_02.c')

Gestão Dinâmica de Memória

As **funções** que se seguem destinam-se à reserva de **memória contígua**. No caso de **sucesso** retornam o **endereço de memória** do início do espaço reservado; no caso de erro retornam '**NULL**'.

- **void *malloc (size_t size);**

Tem como argumento o **número de bytes** a reservar. O espaço reservado **não é inicializado**.

Biblioteca Standard - 'stdlib.h' ('Prog12_02.c')

Gestão Dinâmica de Memória

As **funções** que se seguem destinam-se à reserva de **memória contígua**. No caso de **sucesso** retornam o **endereço de memória** do início do espaço reservado; no caso de erro retornam '**NULL**'.

- **void *malloc (size_t size);**

Tem como argumento o **número de bytes** a reservar. O espaço reservado **não é inicializado**.

- **void *calloc (size_t count, size_t size);**

Reserva espaço para '**count**' elementos, cada um deles ocupando '**size**' bytes. O espaço reservado **é inicializado** a '**0**'.

Biblioteca Standard - 'stdlib.h' ('Prog12_02.c')

Gestão Dinâmica de Memória

As **funções** que se seguem destinam-se à reserva de **memória contígua**. No caso de **sucesso** retornam o **endereço de memória** do início do espaço reservado; no caso de erro retornam '**NULL**'.

- **void *malloc (size_t size);**

Tem como argumento o **número de bytes** a reservar. O espaço reservado **não é inicializado**.

- **void *calloc (size_t count, size_t size);**

Reserva espaço para '**count**' elementos, cada um deles ocupando '**size**' bytes. O espaço reservado **é inicializado** a '**0**'.

- **void *realloc (void *ptr, size_t size);**

Serve para **alterar** o tamanho da memória reservada ('**size**'). Retorna o **ponteiro** que aponta para o **novo tamanho**.

Biblioteca Standard - 'stdlib.h' ('Prog12_02.c')

Gestão Dinâmica de Memória

As **funções** que se seguem destinam-se à reserva de **memória contígua**. No caso de **sucesso** retornam o **endereço de memória** do início do espaço reservado; no caso de erro retornam '**NULL**'.

- **void *malloc (size_t size);**

Tem como argumento o **número de bytes** a reservar. O espaço reservado **não é inicializado**.

- **void *calloc (size_t count, size_t size);**

Reserva espaço para '**count**' elementos, cada um deles ocupando '**size**' bytes. O espaço reservado **é inicializado** a '**0**'.

- **void *realloc (void *ptr, size_t size);**

Serve para **alterar** o tamanho da memória reservada ('**size**'). Retorna o **ponteiro** que aponta para o **novo tamanho**.

Para **libertar** a memória reservada pelas funções anterior em '**ptr**':

Biblioteca Standard - 'stdlib.h' ('Prog12_02.c')

Gestão Dinâmica de Memória

As **funções** que se seguem destinam-se à reserva de **memória contígua**. No caso de **sucesso** retornam o **endereço de memória** do início do espaço reservado; no caso de erro retornam '**NULL**'.

- **void *malloc (size_t size);**

Tem como argumento o **número de bytes** a reservar. O espaço reservado **não é inicializado**.

- **void *calloc (size_t count, size_t size);**

Reserva espaço para '**count**' elementos, cada um deles ocupando '**size**' bytes. O espaço reservado **é inicializado** a '**0**'.

- **void *realloc (void *ptr, size_t size);**

Serve para **alterar** o tamanho da memória reservada ('**size**'). Retorna o **ponteiro** que aponta para o **novo tamanho**.

Para **libertar** a memória reservada pelas funções anterior em '**ptr**':

- **void free (void *ptr);**

Biblioteca Standard - 'stdlib.h'

Aritmética Inteira ('Prog41_05.c')

As **funções** aqui definidas para a **aritmética inteira** são:

Biblioteca Standard - 'stdlib.h'

Aritmética Inteira ('Prog41_05.c')

As **funções** aqui definidas para a **aritmética inteira** são:

- **int abs (int j);**

- **long int labs (long int j);**

Retornam o valor absoluto de 'j' com o tipo respectivo.

Biblioteca Standard - 'stdlib.h'

Aritmética Inteira ('Prog41_05.c')

As **funções** aqui definidas para a **aritmética inteira** são:

- **int abs (int j);**

- **long int labs (long int j);**

Retornam o valor absoluto de 'j' com o tipo respectivo.

- **div_t div (int numerador, int denominador);**

- **ldiv_t ldiv (long int numerador, long int denominador);**

Retornam respectivamente as estruturas 'div_t' e 'ldiv_t', já definidas, que têm o **quociente** e **resto da divisão**.

Biblioteca Standard - 'stdlib.h' ('Prog08_01 e 02.c')

Gerador de Números Pseudo-Aleatórios

Como já se viu, as **funções aleatórias** definidas em **C** são:

Biblioteca Standard - 'stdlib.h' ('Prog08_01 e 02.c')

Gerador de Números Pseudo-Aleatórios

Como já se viu, as **funções aleatórias** definidas em **C** são:

- **void srand (unsigned int seed);**

Serve para definir **internamente** em que ponto se **inicia** a sequência de **números aleatórios** (usa-se, em geral, uma vez);

Biblioteca Standard - 'stdlib.h' ('Prog08_01 e 02.c')

Gerador de Números Pseudo-Aleatórios

Como já se viu, as **funções aleatórias** definidas em **C** são:

- **void srand (unsigned int seed);**

Serve para definir **internamente** em que ponto se **inicia** a sequência de **números aleatórios** (usa-se, em geral, uma vez); Para se ter **valores diferentes**, cada vez que se corre o programa, é usual dar-lhe como argumento o retorno da função **'time'** (o instante actual) que se encontra definida em **'time.h'**:

srand (time (NULL));

Biblioteca Standard - 'stdlib.h' ('Prog08_01 e 02.c')

Gerador de Números Pseudo-Aleatórios

Como já se viu, as **funções aleatórias** definidas em **C** são:

- **void srand (unsigned int seed);**

Serve para definir **internamente** em que ponto se **inicia** a sequência de **números aleatórios** (usa-se, em geral, uma vez); Para se ter **valores diferentes**, cada vez que se corre o programa, é usual dar-lhe como argumento o retorno da função **'time'** (o instante actual) que se encontra definida em **'time.h'**:

srand (time (NULL));

- **int rand (void);**

Não tem argumentos e retorna um número inteiro entre **0** e **RAND_MAX**.

Biblioteca Standard - 'stdlib.h'

Ordenação e Busca

Para a **busca** encontra-se definida a **função**:

```
void *bsearch (const void *key, const void *array,  
              size_t count, size_t size,  
              int (*comp) (const void *, const void *));
```

Biblioteca Standard - 'stdlib.h'

Ordenação e Busca

Para a **busca** encontra-se definida a **função**:

```
void *bsearch (const void *key, const void *array,  
              size_t count, size_t size,  
              int (*comp) (const void *, const void *));
```

- Esta função procura em '**array**' o objecto identificado pelo ponteiro '**key**'. O '**array**' é composto de '**count**' elementos de tamanho '**size**' e tem de estar ordenado por ordem ascendente segundo o critério especificado na função '**comp**' a ser fornecida pelo programador.

Biblioteca Standard - 'stdlib.h'

Ordenação e Busca

Para a **busca** encontra-se definida a **função**:

```
void *bsearch (const void *key, const void *array,  
              size_t count, size_t size,  
              int (*comp) (const void *, const void *));
```

- Esta função procura em '**array**' o objecto identificado pelo ponteiro '**key**'. O '**array**' é composto de '**count**' elementos de tamanho '**size**' e tem de estar ordenado por ordem ascendente segundo o critério especificado na função '**comp**' a ser fornecida pelo programador.
- A função '**comp**' tem dois argumentos e retornar um inteiro.

Biblioteca Standard - 'stdlib.h'

Ordenação e Busca

Para a **busca** encontra-se definida a **função**:

```
void *bsearch (const void *key, const void *array,  
              size_t count, size_t size,  
              int (*comp) (const void *, const void *));
```

- Esta função procura em '**array**' o objecto identificado pelo ponteiro '**key**'. O '**array**' é composto de '**count**' elementos de tamanho '**size**' e tem de estar ordenado por ordem ascendente segundo o critério especificado na função '**comp**' a ser fornecida pelo programador.
- A função '**comp**' tem dois argumentos e retornar um inteiro.
- O modo como a função está escrita permite a utilização para vectores de diferentes tipos.

Biblioteca Standard - 'stdlib.h'

Ordenação e Busca

Para a **busca** encontra-se definida a **função**:

```
void *bsearch (const void *key, const void *array,  
              size_t count, size_t size,  
              int (*comp) (const void *, const void *));
```

- Esta função procura em '**array**' o objecto identificado pelo ponteiro '**key**'. O '**array**' é composto de '**count**' elementos de tamanho '**size**' e tem de estar ordenado por ordem ascendente segundo o critério especificado na função '**comp**' a ser fornecida pelo programador.
- A função '**comp**' tem dois argumentos e retornar um inteiro.
- O modo como a função está escrita permite a utilização para vectores de diferentes tipos.
- A função retorna um ponteiro para o elemento cujo conteúdo se identifica com '**key**'. Se não existir, retorna '**NULL**'.

Biblioteca Standard - 'stdlib.h' ('Prog08_08.c')

Ordenação e Busca

Para a **ordenação** encontra-se definida a **função** que implementa o algoritmo '**quicksort**':

Biblioteca Standard - 'stdlib.h' ('Prog08_08.c')

Ordenação e Busca

Para a **ordenação** encontra-se definida a **função** que implementa o algoritmo '**quicksort**':

```
void qsort (const void *array, size_t count, size_t size,  
            int (*comp) (const void *, const void *));
```

Biblioteca Standard - 'stdlib.h' ('Prog08_08.c')

Ordenação e Busca

Para a **ordenação** encontra-se definida a **função** que implementa o algoritmo '**quicksort**':

```
void qsort (const void *array, size_t count, size_t size,  
           int (*comp) (const void *, const void *));
```

- Em que '**array**' é o vector a ordenar, '**count**' o número de elementos, '**size**' o tamanho de cada elemento e '**comp**' a **função de ordenação** fornecida pelo utilizador.

Biblioteca Standard - 'stdlib.h' ('Prog08_08.c')

Ordenação e Busca

Para a **ordenação** encontra-se definida a **função** que implementa o algoritmo '**quicksort**':

```
void qsort (const void *array, size_t count, size_t size,  
           int (*comp) (const void *, const void *));
```

- Em que '**array**' é o vector a ordenar, '**count**' o número de elementos, '**size**' o tamanho de cada elemento e '**comp**' a **função de ordenação** fornecida pelo utilizador.

O algoritmo '**quicksort**' desenvolvido por A.R.Hoare (1962), consiste basicamente em **seccionar** um vector em **duas partes**, deslocando os **elementos menores** do que um dado elemento (**elemento de paricionamento**, por exemplo, o elemento central) para a '**esquerda**' e os **maiores** para a '**direita**'. Depois, aplica-se **recursivamente** o mesmo processo a **cada parte** até termos apenas um elemento.

Biblioteca Standard - 'stdlib.h' ('Prog08_08.c')

Ordenação e Busca

Para a **ordenação** encontra-se definida a **função** que implementa o algoritmo '**quicksort**':

```
void qsort (const void *array, size_t count, size_t size,  
           int (*comp) (const void *, const void *));
```

- Em que '**array**' é o vector a ordenar, '**count**' o número de elementos, '**size**' o tamanho de cada elemento e '**comp**' a **função de ordenação** fornecida pelo utilizador.

O algoritmo '**quicksort**' desenvolvido por A.R.Hoare (1962), consiste basicamente em **seccionar** um vector em **duas partes**, deslocando os **elementos menores** do que um dado elemento (**elemento de paricionamento**, por exemplo, o elemento central) para a '**esquerda**' e os **maiores** para a '**direita**'. Depois, aplica-se **recursivamente** o mesmo processo a **cada parte** até termos apenas um elemento.

- Ver em "<http://www.wikipedia.org/>" a entrada '**Quicksort**'.

Biblioteca Standard - 'stdlib.h' ('Prog41_02e3.c')

Conversão de Cadeias de Caracteres (I)

Este grupo contém **funções** para converter o conteúdo de **cadeias de caracteres** noutros valores. Começando pela conversão em **inteiros**:

Biblioteca Standard - 'stdlib.h' ('Prog41_02e3.c')

Conversão de Cadeias de Caracteres (I)

Este grupo contém **funções** para converter o conteúdo de **cadeias de caracteres** noutros valores. Começando pela conversão em **inteiros**:

- **long int strtol** (**const char *str**, **char **ptrf**, **int base**);
Converte a cadeia de caracteres '**str**' num '**long int**'. Se a **base** tiver um valor entre '**2**' e '**36**' os dígitos e as letras serão interpretados na **base** correspondente a esse valor. '**ptrf**' é o endereço do elemento que parou a conversão. Os espaços ('**isspace**') iniciais são ignorados.

Biblioteca Standard - 'stdlib.h' ('Prog41_02e3.c')

Conversão de Cadeias de Caracteres (I)

Este grupo contém **funções** para converter o conteúdo de **cadeias de caracteres** noutros valores. Começando pela conversão em **inteiros**:

- **long int strtol** (**const char** ***str**, **char** ****ptrf**, **int base**);
Converte a cadeia de caracteres '**str**' num '**long int**'. Se a **base** tiver um valor entre '**2**' e '**36**' os dígitos e as letras serão interpretados na **base** correspondente a esse valor. '**ptrf**' é o endereço do elemento que parou a conversão. Os espaços ('**isspace**') iniciais são ignorados.
- **unsigned long int strtoul** (**const char** ***str**, **char** ****ptrf**, **int base**);
Idêntica a '**strtol**' mas retorna um '**unsigned long int**'.

Biblioteca Standard - 'stdlib.h' ('Prog41_02e3.c')

Conversão de Cadeias de Caracteres (I)

Este grupo contém **funções** para converter o conteúdo de **cadeias de caracteres** noutros valores. Começando pela conversão em **inteiros**:

- **long int strtol** (**const char** ***str**, **char** ****ptrf**, **int base**);
Converte a cadeia de caracteres '**str**' num '**long int**'. Se a **base** tiver um valor entre '**2**' e '**36**' os dígitos e as letras serão interpretados na **base** correspondente a esse valor. '**ptrf**' é o endereço do elemento que parou a conversão. Os espaços ('**isspace**') iniciais são ignorados.
- **unsigned long int strtoul** (**const char** ***str**, **char** ****ptrf**, **int base**);
Idêntica a '**strtol**' mas retorna um '**unsigned long int**'.
- **int atoi** (**const char** ***str**);
Equivalente a '**strtol** (**str**, (**char** **) **NULL**, **10**)';. Retorna '**int**'.

Biblioteca Standard - 'stdlib.h' ('Prog41_02.c')

Conversão de Cadeias de Caracteres (II)

No que diz respeito à conversão de **string** para '**double**' o processo é análogo ao que vimos para os inteiros:

Biblioteca Standard - 'stdlib.h' ('Prog41_02.c')

Conversão de Cadeias de Caracteres (II)

No que diz respeito à conversão de **string** para '**double**' o processo é análogo ao que vimos para os inteiros:

- **double strtod** (**const char *str**, **char **ptrf**);

É análoga a '**strtol**' para '**double**'. Converte a cadeia de caracteres '**str**' num '**double**' que retorna.

Biblioteca Standard - 'stdlib.h' ('Prog41_02.c')

Conversão de Cadeias de Caracteres (II)

No que diz respeito à conversão de **string** para '**double**' o processo é análogo ao que vimos para os inteiros:

- **double strtod** (**const char *str**, **char **ptrf**);

É análoga a '**strtol**' para '**double**'. Converte a cadeia de caracteres '**str**' num '**double**' que retorna.

No caso do valor convertido ser **demasiado grande** (situação de '**overflow**') é retornado '**±HUGE_VAL**'. Se é **demasiado pequeno**, é retornado '**0**'.

Biblioteca Standard - 'stdlib.h' ('Prog41_02.c')

Conversão de Cadeias de Caracteres (II)

No que diz respeito à conversão de **string** para **'double'** o processo é análogo ao que vimos para os inteiros:

- **double strtod** (**const char** ***str**, **char** ****ptrf**);

É análoga a **'strtol'** para **'double'**. Converte a cadeia de caracteres **'str'** num **'double'** que retorna.

No caso do valor convertido ser **demasiado grande** (situação de **'overflow'**) é retornado **'±HUGE_VAL'**. Se é **demasiado pequeno**, é retornado **'0'**.

- **double atof** (**const char** ***str**);

Equivalente a **'strtod (str, (char **) NULL);'**.

Biblioteca Standard - 'stdlib.h'

Controle de Saída de Programas

Aqui são definidas **funções** que controlam a **saída forçada** de um programa. Para mais informações sobre estas funções ver manuais.

Biblioteca Standard - 'stdlib.h'

Controle de Saída de Programas

Aqui são definidas **funções** que controlam a **saída forçada** de um programa. Para mais informações sobre estas funções ver manuais.

- **void abort (void);**

Produz a **saída forçada** de um programa e **não executa operações de limpeza**.

Biblioteca Standard - 'stdlib.h'

Controle de Saída de Programas

Aqui são definidas **funções** que controlam a **saída forçada** de um programa. Para mais informações sobre estas funções ver manuais.

- **void abort (void);**

Produz a **saída forçada** de um programa e **não executa operações de limpeza**.

- **int atexit (void (*func) (void));**

Regista a função '**func**' para ser executada numa saída **normal** do programa. Podem ser registadas até **32 funções**. **Não há passagem de argumentos** para estas funções.

Biblioteca Standard - 'stdlib.h'

Controle de Saída de Programas

Aqui são definidas **funções** que controlam a **saída forçada** de um programa. Para mais informações sobre estas funções ver manuais.

- **void abort (void);**

Produz a **saída forçada** de um programa e **não executa operações de limpeza**.

- **int atexit (void (*func) (void));**

Regista a função '**func**' para ser executada numa saída **normal** do programa. Podem ser registadas até **32 funções**. **Não há passagem de argumentos** para estas funções.

- **int exit (int status);**

Esta **função** termina o programa de **forma normal**. As funções registadas com '**atexit**' são executadas, os **stream** abertos são fechados, **os dados são enviados para os destinos**, os ficheiros temporários são **apagados**.

Biblioteca Standard - 'stdlib.h'

Comunicação com o Ambiente ('Prog41_04.c')

São aqui descritas duas **funções** que nos permitir **interactuar** com o sistema:

Biblioteca Standard - 'stdlib.h'

Comunicação com o Ambiente ('Prog41_04.c')

São aqui descritas duas **funções** que nos permitir **interactuar** com o sistema:

- **char *getenv (const char *name);**

Procura na lista das variáveis de ambiente uma com o nome '**name**'. Se tiver sucesso, retorna o **valor associado** a essa variável. Caso contrário retorna '**NULL**'.

Biblioteca Standard - 'stdlib.h'

Comunicação com o Ambiente ('Prog41_04.c')

São aqui descritas duas **funções** que nos permitem **interactuar** com o sistema:

- **char *getenv (const char *name);**

Procura na lista das variáveis de ambiente uma com o nome '**name**'. Se tiver sucesso, retorna o **valor associado** a essa variável. Caso contrário retorna '**NULL**'.

- **int system (const char *command);**

Passa '**command**' como um comando a ser **executado** pelo interpretador de comandos (**shell**) do sistema operativo. Se o **comando** for possível de interpretar pelo sistema, a função retorna o **estado de saída** devolvido pelo comando executado.

Biblioteca Standard - 'stdlib.h'

Comunicação com o Ambiente - 'PATH'

- Quando se quer executar, num **sistema unix**, com uma certa frequência, programas por nós desenvolvidos é cómodo indicar que a pasta em que estamos a trabalhar é uma pasta válida para **executar programas**.

Biblioteca Standard - 'stdlib.h'

Comunicação com o Ambiente - 'PATH'

- Quando se quer executar, num **sistema unix**, com uma certa frequência, programas por nós desenvolvidos é cómodo indicar que a pasta em que estamos a trabalhar é uma pasta válida para **executar programas**.
- A **variável de ambiente** que tem essa informação é '**PATH**'. Para ver o seu conteúdo basta fazer na '**shell**': '**env**'.

Biblioteca Standard - 'stdlib.h'

Comunicação com o Ambiente - 'PATH'

- Quando se quer executar, num **sistema unix**, com uma certa frequência, programas por nós desenvolvidos é cómodo indicar que a pasta em que estamos a trabalhar é uma pasta válida para **executar programas**.
- A **variável de ambiente** que tem essa informação é '**PATH**'. Para ver o seu conteúdo basta fazer na '**shell**': '**env**'.
- Essa **variável** pode ser alterada **modificando** o seu valor no ficheiro '**.bashrc**', que se encontra na pasta principal do utilizador, e onde se encontram as **definições** das **variáveis** e **aliases** que se desejam acrescentar às básicas do sistema.

Biblioteca Standard - 'stdlib.h'

Comunicação com o Ambiente - 'PATH'

- Quando se quer executar, num **sistema unix**, com uma certa frequência, programas por nós desenvolvidos é cómodo indicar que a pasta em que estamos a trabalhar é uma pasta válida para **executar programas**.
- A **variável de ambiente** que tem essa informação é '**PATH**'. Para ver o seu conteúdo basta fazer na '**shell**': '**env**'.
- Essa **variável** pode ser alterada **modificando** o seu valor no ficheiro '**.bashrc**', que se encontra na pasta principal do utilizador, e onde se encontram as **definições** das **variáveis** e **aliases** que se desejam acrescentar às básicas do sistema.
- Para fazer essa alteração deverá **acrescentar-se** a própria pasta, './', à variável '**PATH**':

```
PATH=.:$PATH
```

Biblioteca Standard - 'stdlib.h'

Caracteres e Strings Multibyte ('Prog43_01-03.c')

- Como já se disse, os **256** valores que um byte pode tomar são **insuficientes** para dar conta dos diferentes alfabetos e símbolos.

Biblioteca Standard - 'stdlib.h'

Caracteres e Strings Multibyte ('Prog43_01-03.c')

- Como já se disse, os **256** valores que um byte pode tomar são **insuficientes** para dar conta dos diferentes alfabetos e símbolos.
- A norma **Unicode** permite resolver este problema e implementa duas estratégias:

Biblioteca Standard - 'stdlib.h'

Caracteres e Strings Multibyte ('Prog43_01-03.c')

- Como já se disse, os **256** valores que um byte pode tomar são **insuficientes** para dar conta dos diferentes alfabetos e símbolos.
- A norma **Unicode** permite resolver este problema e implementa duas estratégias:
 - 1 Número de caracteres **fixo** (2 ou 4 bytes, **wide char**, ...);

Biblioteca Standard - 'stdlib.h'

Caracteres e Strings Multibyte ('Prog43_01-03.c')

- Como já se disse, os **256** valores que um byte pode tomar são **insuficientes** para dar conta dos diferentes alfabetos e símbolos.
- A norma **Unicode** permite resolver este problema e implementa duas estratégias:
 - 1 Número de caracteres **fixo** (2 ou 4 bytes, **wide char**, ...);
 - 2 Número de caracteres **variável** (**UTF-8**, etc.);

Caracteres e Strings Multibyte ('Prog43_01-03.c')

- Como já se disse, os **256** valores que um byte pode tomar são **insuficientes** para dar conta dos diferentes alfabetos e símbolos.
- A norma **Unicode** permite resolver este problema e implementa duas estratégias:
 - 1 Número de caracteres **fixo** (2 ou 4 bytes, **wide char**, ...);
 - 2 Número de caracteres **variável** (**UTF-8**, etc.);
- Em **C**, os programas **herdam** as **variáveis de ambiente**.

Caracteres e Strings Multibyte ('Prog43_01-03.c')

- Como já se disse, os **256** valores que um byte pode tomar são **insuficientes** para dar conta dos diferentes alfabetos e símbolos.
- A norma **Unicode** permite resolver este problema e implementa duas estratégias:
 - 1 Número de caracteres **fixo** (2 ou 4 bytes, **wide char**, ...);
 - 2 Número de caracteres **variável** (**UTF-8**, etc.);
- Em **C**, os programas **herdam** as **variáveis de ambiente**.
- Tal **não acontece** com a parte referente aos **idiomas**, porque a norma de **C** diz que um programa deve iniciar-se com o **'locale'** padrão **'C'**.

Biblioteca Standard - 'stdlib.h'

Caracteres e Strings Multibyte ('Prog43_01-03.c')

- Como já se disse, os **256** valores que um byte pode tomar são **insuficientes** para dar conta dos diferentes alfabetos e símbolos.
- A norma **Unicode** permite resolver este problema e implementa duas estratégias:
 - 1 Número de caracteres **fixo** (2 ou 4 bytes, **wide char**, ...);
 - 2 Número de caracteres **variável** (**UTF-8**, etc.);
- Em **C**, os programas **herdam** as **variáveis de ambiente**.
- Tal **não acontece** com a parte referente aos **idiomas**, porque a norma de **C** diz que um programa deve iniciar-se com o **'locale'** padrão **'C'**.
- Para usar os **'locais'** específicos do **ambiente** deve chamar-se a função **'setlocale'** da seguinte forma:

```
setlocale (LC_ALL, "");
```

Biblioteca Standard - 'stdlib.h'

Caracteres e Strings Multibyte ('Prog43_01-03')

Aqui são descritas as **funções** que tratam a **codificação de caracteres** que não podem ser tratados por **um único byte** ('char'):

Biblioteca Standard - 'stdlib.h'

Caracteres e Strings Multibyte ('Prog43_01-03')

Aqui são descritas as **funções** que tratam a **codificação de caracteres** que não podem ser tratados por **um único byte** ('char'):

- **int mblen** (**const char *string**, **size_t nmax**);

Se a '**string**' for nula retorna '**0**'. No caso contrário, a função devolve o **número de bytes necessários** à extracção do **próximo caracter**. Se não for um caracter válido retorna '**-1**'.

Biblioteca Standard - 'stdlib.h'

Caracteres e Strings Multibyte ('Prog43_01-03')

Aqui são descritas as **funções** que tratam a **codificação de caracteres** que não podem ser tratados por **um único byte** ('char'):

- **int mblen** (**const char *string**, **size_t nmax**);
Se a '**string**' for nula retorna '**0**'. No caso contrário, a função devolve o **número de bytes necessários** à extracção do **próximo caracter**. Se não for um caracter válido retorna '**-1**'.
- **int mbtowc** (**wchar_t *pwc**, **const char *string**, **size_t nmax**);
Serve para converter um caracter **multibyte** ('**string**') para **wide char** ('pwc'). O seu retorno é igual ao de '**mblen**'.

Caracteres e Strings Multibyte ('Prog43_01-03')

Aqui são descritas as **funções** que tratam a **codificação de caracteres** que não podem ser tratados por **um único byte** ('char'):

- **int mblen** (**const char *string**, **size_t nmax**);
Se a '**string**' for nula retorna '**0**'. No caso contrário, a função devolve o **número de bytes necessários** à extracção do **próximo caracter**. Se não for um caracter válido retorna '**-1**'.
- **int mbtowc** (**wchar_t *pwc**, **const char *string**, **size_t nmax**);
Serve para converter um caracter **multibyte** ('**string**') para **wide char** ('**pwc**'). O seu retorno é igual ao de '**mblen**'.
- **int wctomb** (**const char *string**, **wchar_t *wc**);
É a função inversa da anterior. Serve para converter um caracter '**wide char**' ('**wc**') num '**multibyte**' ('**string**'). O seu retorno é igual ao número de caracteres escritos em '**string**'.

Caracteres e Strings Multibyte ('Prog43_01-03')

- `size_t mbstowcs (wchar_t *wstring, const char *string, size_t num);`

Converte a string de caracteres 'multibyte' 'string' num vector de 'wide char' 'wstring', sendo 'num' o número máximo de 'wide char' que podem ser escritos. Se tiver sucesso, retorna o número de 'wide char' escritos (não contando o caracter '0').

Caracteres e Strings Multibyte ('Prog43_01-03')

- **size_t mbstowcs** (**wchar_t** ***wstring**, **const char** ***string**, **size_t num**);

Converte a string de caracteres '**multibyte**' '**string**' num vector de '**wide char**' '**wstring**', sendo '**num**' o número máximo de '**wide char**' que podem ser escritos. Se tiver sucesso, retorna o número de '**wide char**' escritos (não contando o caracter '**0**').

- **size_t wcstombs** (**char** ***string**, **const wchar_t** ***wstring**, **size_t num**);

Idêntica à função anterior mas no sentido inverso. Converte a string de caracteres '**wide char**' '**wstring**' numa string de caracteres '**multibyte**' e retorna o número de caracteres escritos sem contar com o caracter '**0**'.