Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação de Sistemas Computacionais

Inverno de 2023/2024

Série de Exercícios 1

Realize os exercícios seguintes usando a linguagem C. Deve testar devidamente o código desenvolvido, bem como apresentá-lo de forma cuidada, apropriadamente indentado e comentado. Assegure-se de que o compilador não emite qualquer aviso sobre o seu código, com as opções `-Wall -pedantic` activas. Contacte o docente se tiver dúvidas. Não é necessário relatório. Encoraja-se a discussão de problemas e soluções com outros colegas, mas a partilha directa de soluções leva, no mínimo, à anulação das entregas de todos os envolvidos.

1. Considere o critério "complemento para 2" na representação de valores inteiros relativos (com sinal). Escreva a função **`max_value`**, que retorna o maior valor positivo que pode ser representado pelo número de *bits* equivalente ao número de *bits* de um **`char`**, vezes o valor indicado pelo parâmetro **`nchars`**. Se o valor não for representável num **`unsigned long`**, deve retornar zero. Deve basear-se na macro **`CHAR_BIT`**, definida em **`limits.h`**, para obter o número de *bits* de um **`char`**.

   ```
   unsigned long max_value(unsigned nchars);
   ```

2. Considere vetores de *bits* representados sobre *arrays* **`de unsigned long`** em que **`ULONG_BIT`** é o número de *bits* de um **`unsigned long`**. No índice **`n`** de um *array* de **`unsigned long`** o *bit* de menor peso corresponde ao índice **`n * ULONG_BIT`** do vector de *bits*, enquanto que o *bit* de maior peso corresponde ao índice **`(n + 1) * ULONG_BIT - 1`**. A função **`getbits`** retorna o valor dos *bits* entre as posições **`index`** e **`index + length - 1`** do vetor de *bits* representado por **`data`**. A função **`setbits`** escreve os **`len`** *bits* de menor peso de **`val`** nas posições entre **`index`** e **`index + length - 1`** do vetor de *bits* representado por **`data`**. Em ambas as funções, **`length`** nunca é maior do que **`ULONG_BIT`**. Exemplo: para **`data = {0xBFFFFFECABCD1234, 0xC, 2, 3 }`**, a chamada a **`getbits(data, 29, 8)`** retorna **`0x0000000000000065`**. Defina **`ULONG_BIT`** com base em **`CHAR_BIT`** e implemente as funções **`getbits`** e **`setbits`**.

   ```
   unsigned long getbits(unsigned long data[], unsigned index, unsigned length);

   void setbits(unsigned long data[], unsigned index, unsigned length, unsigned long val);
   ```

3. Programe em linguagem C a função **`string_split`**, que separa, em palavras isoladas, o texto recebido no parâmetro **`text`** com formato *string* C. Palavra é uma sequência de caracteres delimitada por sequências de um ou mais caracteres separadores (**`' '`**, **`'\t'`** ou **`'\n'`**). O array de ponteiros **`words`**, cuja dimensão é definida por **`words_size`**, deve ser preenchido com os endereços de início das palavras, pela ordem em que se encontram no texto. O caractere separador, no final de cada palavra, deve ser substituído por **`'\0'`**, para que as palavras fiquem formatadas como *strings* C. A função devolve o número de palavras assinaladas em **`words`**. No caso da função devolver um valor igual a **`words_size`**, pode também significar que o número de palavras no texto é superior à capacidade de **`words`**. Nesse caso, as restantes palavras podem ser obtidas em chamada(s) posterior(es). A forma de indicar que se pretende continuar a obter palavras do texto da chamada anterior é usar o argumento **`NULL`** como primeiro parâmetro.

   ```
   size_t string_split(char *text, char *separators, char *words[], size_t words_size);
   ```

4. Programe a função **`string_to_time`** que converte a informação de data e hora, recebida na forma de *string* C, para uma *struct* do tipo **`struct tm`**[1]. A *string* de entrada tem o formato "**`dd-mm-aaaa hh:mm:ss`**". Os campos **`tm_wday`**, **`tm_yday`** e **`tm_isdst`** devem ser colocados a zero. Se a *string* de entrada estiver mal formatada a função devolve 0, senão devolve 1;

   ```
   struct tm {
     int   tm_sec;
     int   tm_min;
     int   tm_hour;
     int   tm_mday;
     int   tm_mon;
     int   tm_year;
     int   tm_wday; int   tm_yday; int   tm_isdst;
   };

   int string_to_time(const char *string, struct tm *tm);
   ```

[1] http://www.cplusplus.com/reference/ctime/tm/

5. Realize um programa para operar sobre ficheiros de dados em formato CSV[2]. O programa deve selecionar as linhas que contenham o valor **<pattern>** na coluna indicada por **<column>** produzindo um novo ficheiro com as linhas selecionadas.

   $ csv_filter <option> <column> <pattern>

   Descrição das opções:

-o <file>	indica o nome do ficheiro de saída, com as linhas selecionadas; em caso de omissão usar **stdout**;
-i <file>	indica o nome do ficheiro de entrada, com o texto a processar; em caso de omissão usar **stdin**;
-c	indica se a comparação léxica é sensível a maiúsculas e minúsculas; em caso de omissão é insensível;

   Deve utilizar a função getopt no processamento das opções. A utilização desta função permite que as opções possam ser escritas em qualquer posição relativamente aos parâmetros, como no exemplo abaixo. Já os outros parâmetros, como não dispõem de qualquer elemento distintivo, devem ser escritos por ordem.

   O programa deve procurar o ficheiro de entrada na diretoria indicada pela variável de ambiente **CSV_FILTER_PATH**. No caso desta variável não estar definida procura na diretoria corrente.

   Em situações de erro, o programa deve devolver um valor diferente de zero. As situações de erro são dificuldades no acesso a ficheiros ou argumentos do programa inválidos.

   Exemplo:

   Considerando o ficheiro **stock.csv**, localizado em **/usr/home/data**, com o seguinte conteúdo:

   ```
   máquina lavar roupa,Indesit,Lisboa,L3
   máquina lavar loiça,Bosch,Porto,D4
   torradeira,Moulinex,LISBOA,A10
   microondas,Samsung,Porto,E4
   ```

   Definir a variável de ambiente:

   ```
   $ export CSV_FILTER_PATH=/usr/home/data
   ```

   O seguinte comando:

   ```
   $ csv_filter 3 Lisboa -o stock_lisboa.csv -i stock.csv -c
   ```

   produz o ficheiro **stock_lisboa.csv** com o seguinte conteúdo:

   ```
   máquina lavar roupa, Indesit, Lisboa, L3
   ```

Data recomendada de entrega: 15 de Outubro de 2020

ISEL, 18 de Setembro de 2023

[2] https://pt.wikipedia.org/wiki/Comma-separated_values