Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação de Sistemas Computacionais

Teste Global da Época de Recurso, Inverno de 2020/2021

(Moodle)

Pergunta 1 A

Implemente em linguagem C a função **swap_bits** que troca o conteúdo de um conjunto de bits entre duas palavras. O conjunto de bits é definido, em ambas as palavras, pela posição do primeiro bit - **first_bit** e o número de bits seguintes - **n_bits**.

void swap_bits(unsigned *word1, unsigned *word2, int first_bit, int n_bits);

Pergunta 1 B

Implemente em linguagem C a função **swap_bits** que troca o conteúdo de um conjunto de bits entre duas palavras. O conjunto de *bits*, em ambas as palavras, abrange os *bits* entre a posição **first_bit** e a posição **last_bit**.

void swap_bits(unsigned *word1, unsigned *word2, int first_bit, int last_bit);

Pergunta 2 A

Pretende-se modificar linhas de texto com dados na forma de "*comma separated values*" da convenção americana par a portuguesa. No formato original os elementos de informação são separados por vírgula. O respetivo conteúdo pode ser texto ou valor numérico; neste, pode ser usado o ponto decimal.

Implemente, em linguagem C, a função **change_csv** que modifica a linha indicada por **line**, substituindo os separadores de elemento '**,**' por '**;**' e o ponto decimal '**.**' por vírgula '**,**'.

Se houver elementos originalmente iniciados por ponto, além da conversão para vírgula deve ser inserido um '**0**' à esquerda da mesma. Admita que a linha dispõe de espaço para crescer se necessário.

No caso de existirem separadores consecutivos, devem ser todos reproduzidos.

int change_csv(char *line);

Exemplo: a linha "**abc,,123.456,ABC,.123**" deve ser modificada para "**abc;;123,456;ABC;0,123**".

Pergunta 2 A

Pretende-se modificar linhas de texto com dados na forma de "*comma separated values*" da convenção portuguesa para a americana. No formato original os elementos de informação são separados por ponto-e-vírgula. O respetivo conteúdo pode ser texto ou valor numérico; neste, pode ser usada a vírgula decimal.

Implemente, em linguagem C, a função **change_csv** que modifica a linha indicada por **line**, substituindo os separadores de elemento '**;**' por '**,**' e a vírgula decimal '**,**' por ponto '**.**'.

Se houver elementos contendo apenas um algarismo '**0**' à esquerda da vírgula, além da conversão desta para ponto, o '**0**' deve ser suprimido, ficando o elemento iniciado por '**.**' seguido dos restantes algarismos.

No caso de existirem separadores consecutivos, devem ser todos reproduzidos.

Exemplo: "**abc;;123,456;ABC;0,123**" deve ser convertido para "**abc,,123.456,ABC,.123**".

int change_csv(char *line);

Pergunta 3 A

Programe, em assembly x86-64, a função **get_val_ptr** cuja definição em linguagem C se apresenta a seguir. Deve integrar, como início da sua implementação, o código *assembly* apresentado.

```
typedef struct data { short flags:6; short length:10; short vals[]; } Data;

typedef struct info { double ref; Data **data; int valid; } Info;

short *get_val_ptr(Info items[], size_t item_idx, size_t data_idx,
                   size_t val_idx, short mask) {
      return items[item_idx].valid
            && val_idx < items[item_idx].data[data_idx]->length
            && (items[item_idx].data[data_idx]->flags & mask)
                  ? &(items[item_idx].data[data_idx]->vals[val_idx])
                  : NULL;
}
```

```
      .text
      .global  get_val_ptr
get_val_ptr:
      shl   $3, %rsi
      lea   (%rsi, %rsi, 2), %rsi
      lea   (%rdi, %rsi), %r10
      ...
```

Perguntar 3 B

Programe, em assembly x86-64, a função **get_val_ptr** cuja definição em linguagem C se apresenta a seguir. Deve integrar, como início da sua implementação, o código *assembly* apresentado.

```
typedef struct data { short flags:6; short length:10; short vals[7]; } Data;

typedef struct info { double ref; Data data[4]; int valid; } Info;

short *get_val_ptr(Info items[], size_t item_idx, size_t data_idx,
                   size_t val_idx, short mask) {
      return items[item_idx].valid
            && val_idx < items[item_idx].data[data_idx].length
            && (items[item_idx].data[data_idx].flags & mask)
                  ? &(items[item_idx].data[data_idx].vals[val_idx])
                  : NULL;
}
```

```
      .text
      .global  get_val_ptr
get_val_ptr:
      shl   $4, %rsi
      lea   (%rsi, %rsi, 4), %rsi
      lea   (%rdi, %rsi), %r9
      ...
```

Pergunta 4 A

Considere a função **sortedarray_insert** cuja definição em linguagem C se apresenta a seguir:

```
void sortedarray_insert(void *array[], unsigned length,
      int (*cmp_func)(const void *, const void *), void *data) {
      void **p;
      for (p = array; p < array + length && cmp_func(*p, data) < 0 ; p++)
            ;
      if (p < array + length) {
            memmove(p + 1, p, sizeof *p * (array + length - p));
            *p = data;
      }
}
```

```
      .text
      .global  sortedarray_insert
sortedarray_insert:
      push  %r15
      push  %r14
      push  %r13
      push  %r12
      mov   %rdx, %r12
      mov   %rcx, %r13
      mov   %rdi, %r14
      mov   %esi, %r15d
      ...
```

```
      .text
      .global  sortedarray_insert
sortedarray_insert:
      mov   %rdi, %r12
      add   $0, %esi
      shl   $3, %rsi
      add   %rdi, %rsi
sai_for:
      cmp   %rsi, %r12
      ...
```

a) [2,5] Programe a função **sortedarray_insert** em *assembly* x86-64, integrando uma das variantes de código *assembly* apresentado, como o início do seu código.

b) [2,5] Considerando o tipo **Car** para representar a informação de um automóvel, escreva, em linguagem C, um programa de teste da função **sortedarray_insert** que insira um novo automóvel, numa lista de automóveis previamente ordenada segundo a chapa de matrícula (campo **registration**). A lista de automóveis é representada por um array de ponteiros para *structs* do tipo Car.

No programa, deve explicitar a definição e inicialização do *array* com pelo menos três posições, a definição da função de comparação, assim como a invocação da função **sortedarray_insert**.

typedef struct car { char *brand, char *registration, float displacement } Car;

Pergunta 4 B

Considere a função **sortedarray_insert** cuja definição em linguagem C se apresenta a seguir:

```
void sortedarray_insert(void *array[], unsigned length,
      void *data, int (*cmp_func)(const void *, const void *)) {
      void **p;
      for (p = array; p < array + length && cmp_func(*p, data) < 0 ; p++)
            ;
      if (p < array + length) {
            memmove(p + 1, p, sizeof *p * (array + length - p));
            *p = data;
      }
}
```

```
      .text
      .global  sortedarray_insert
sortedarray_insert:
      push  %r15
      push  %r14
      push  %r13
      push  %r12
      mov   %rdx, %r12
      mov   %rcx, %r13
      mov   %rdi, %r14
      mov   %esi, %r15d
      ...
```

```
      .text
      .global  sortedarray_insert
sortedarray_insert:
      mov   %rdi, %r12
      mov   %esi, %esi
      lea   (%rdi, %rsi, 8), %rax
sai_for:
      cmp   %r12, %rax
      ...
```

a) [2,5] Programe a função sortedarray_insert em *assembly* x86-64, integrando uma das variantes de código *assembly* apresentado, como o início do seu código.

b) [2,5] Considerando o tipo **Car** para representar a informação de um automóvel, escreva, em linguagem C, um programa de teste da função **sortedarray_insert** que insira um novo automóvel, numa lista de automóveis previamente ordenada segundo a chapa de matrícula (campo **registration**). A lista de automóveis é representada por um array de ponteiros para *structs* do tipo Car.

No programa, deve explicitar a definição e inicialização do *array* com pelo menos três posições, a definição da função de comparação, assim como a invocação da função **sortedarray_insert**.

typedef struct car { char *brand, char *registration, float displacement } Car;

Pergunta 5 A

Considere o conteúdo dos ficheiros fonte f1.c e f2.c.

```
/* f1.c */
#include <stdio.h>

extern int People;

#define DATA_SIZE 34

unsigned char data[DATA_SIZE] =
      { 20, 0, 0, 0, 10, 0, 0, 0, 30 };

int calc(char *, size_t);

int main() {
      printf("%d\n", calc(data, sizeof data));
}
```

```
* f2.c */

#define DATA_SIZE 1

unsigned int data[DATA_SIZE];

static int acc;

int calc(int *array) {
      for (int i = 0; i < sizeof array; ++i)
            acc += array[i] + data[i];
      return acc;
}
```

a) [1,5] Indique o conteúdo das tabelas de símbolos dos ficheiros objecto relocáveis resultantes da compilação de f1.c e de f2.c. Para cada símbolo indique o nome, a secção e o respectivo âmbito (local ou global) (pode usar a mesma notação do utilitário nm).

b) [1,5] Se considerar que os ficheiros objecto f1.o e f2.o podem ser combinados pelo linker com sucesso para gerar um executável, diga o que é mostrado na consola quando se executa o executável produzido. Se, pelo contrário, considera que o linker falhará a combinação, diga qual a razão ou razões porque isso acontece. Justifique a sua resposta.

Pergunta 5 B

Considere o conteúdo dos ficheiros fonte f1.c e f2.c.

```
/* f1.c */
#include <stdio.h>

extern int Person;

#define DATA_SIZE 32

static unsigned char data[DATA_SIZE] =
      { 19, 0, 0, 0, 10, 0, 0, 0, 30 };

int calc(char *, size_t);

int main() {
      printf("%d\n", calc(data, sizeof data));
}
```

```
* f2.c */

#define DATA_SIZE 8

unsigned int data[DATA_SIZE];

int acc;

int calc(int *array) {
      for (int i = 0; i < sizeof array; ++i)
            acc += array[i] + data[i];
      return acc;
}
```

a) [1,5] Indique o conteúdo das tabelas de símbolos dos ficheiros objecto relocáveis resultantes da compilação de f1.c e de f2.c. Para cada símbolo indique o nome, a secção e o respectivo âmbito (local ou global) (pode usar a mesma notação do utilitário nm).

b) [1,5] Se considerar que os ficheiros objecto f1.o e f2.o podem ser combinados pelo linker com sucesso para gerar um executável, diga o que é mostrado na consola quando se executa o executável produzido. Se, pelo contrário, considera que o linker falhará a combinação, diga qual a razão ou razões porque isso acontece. Justifique a sua resposta.

Pergunta 6 A

```
typedef struct book {
      char *title;
      char *collection;
      char *authors;
      char *keywords;
      char *publisher;
      char *isbn;
      int year;
      double price;
} Book;

typedef struct bookList {
      struct bookList *next;
      Book *book;
} BookList;
```

```
typedef struct simpleBook {
      struct simpleBook *next;
      char *title;
      char *collection;
      char *authors;
      char isbn[ISBN_FIELD_SIZE];
      double price;
} SimpleBook;
```

```
SimpleBook *removeAndConvert( BookList *source, char *publ );
```

Considere um catálogo de livros representado por uma lista ligada de nós com o tipo BookList. Cada nó dá acesso ao descritor de um livro, com o tipo Book. Neste, os campos do tipo char * identificam strings alojadas dinamicamente e preenchidas com informações do livro.

A função **removeAndConvert** destina-se a retirar desta lista alguns elementos, identificados pelo campo **publisher** idêntico ao parâmetro **publ**, e colocá-los numa nova lista ligada de nós com o tipo **SimpleBook**. Este tipo tem uma representação diferente, podendo haver conversão de conteúdos e eliminação de alguns campos. Os espaços alojados dinamicamente que sejam úteis na nova representação devem ser aproveitados; os que não forem úteis devem ser reciclados.

Escreva a função **removeAndConvert** , que realiza as modificações especificadas e retorna o endereço do primeiro elemento da nova lista. Se considerar conveniente, pode escrever funções auxiliares.

Pergunta 6 B

```
typedef struct book {
     char *title;
     char *collection;
     char *authors;
     char *keywords;
     char *publisher;
     char *isbn;
     int year;
     double price;
} Book;

typedef struct bookList {
     Book *book;
     struct bookList *link;
} BookList;
```

```
typedef struct simpleBook {
     char *title;
     char *authors;
     char isbn[ISBN_FIELD_SIZE];
     double price;
     struct simpleBook *link;
} SimpleBook;
```

```
SimpleBook *removeAndConvert( BookList *source, char *coll );
```

Considere um catálogo de livros representado por uma lista ligada de nós com o tipo **BookList**. Cada nó dá acesso ao descritor de um livro, com o tipo **Book**. Neste, os campos do tipo **char *** apontam *strings* alojadas dinamicamente e preenchidas com informações do livro.

A função **removeAndConvert** destina-se a retirar desta lista alguns elementos, identificados pelo campo **collection** idêntico ao parâmetro **coll**, e colocá-los numa nova lista ligada de nós com o tipo **SimpleBook**. Este tipo tem uma representação diferente, podendo haver conversão de conteúdos e eliminação de alguns campos. Os espaços alojados dinamicamente que sejam úteis na nova representação devem ser aproveitados; os que não forem úteis devem ser reciclados.

Escreva a função **removeAndConvert**, que realiza as modificações especificadas e retorna o endereço do primeiro elemento da nova lista. Se considerar conveniente, pode escrever funções auxiliares.