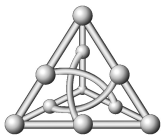


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Trabalho 03: Ponteiros e *Strings* em C

Redes de Computadores

1 Descrição

Este trabalho pode ser feito em dupla e apenas um dos integrantes precisa entregar no Moodle.

2 *String* e Ponteiros

Seu programa deve ter um único argumento por linha de comando. Ou seja, definindo a `main` como:

```
int main (int argc, char **argv)
```

seu programa deve:

- se `argc != 2`, imprimir uma mensagem de erro e sair;
- se `argc == 2`, usar o valor armazenado em `argv[1]` como argumento para o resto do programa.

O argumento será uma *string* separada por '/', por exemplo:

```
brivaldo/junior/edoardo/biagioni/jon/moroney/foo/bar/baz
```

Seu programa deverá criar um ponteiro `p` e apontar para o primeiro caractere do argumento. Para cada *string* ele deve:

- encontrar o tamannho `L` da *string* que se refere a `p` (sem incluir qualquer caractere '/');
- use `malloc` para alocar o *buffer* `B` com tamanho `L` +1;
- verifique se o valor de retorno do `malloc` é não NULL. Se o valor for NULL, imprima a mensagem e saia;
- copie os caracteres da *string* do argumento passado por linha de comando para o *buffer*;
- adicione o terminador de caracteres nulo ('\0') ao *buffer*;
- imprima o *buffer* e o número de caracteres que o *buffer* utilizou:
  `printf ("%s *(%d)\n", buffer, numero_de_chars_no_buffer);` [1]

[1] `numero_de_chars_no_buffer` deve ser substituido pela expressão apropriada

- libere (`free`) o *buffer*;
- atualize o ponteiro `p` para apontar para a próxima *string* no argumento.

Por exemplo, fornecendo a *string* de exemplo anterior, seu programa deverá imprimir:

```
brivaldo (9)
junior (7)
edoardo (8)
biagioni (9)
jon (4)
moroney (8)
foo (4)
bar (4)
baz (4)
```

O nome do seu arquivo como parte do trabalho deve ser `strings.c`.

3 Estruturas, Ponteiros e *Buffers*

Inclua o seguinte no seu código:

```
#include <stdio.h>
#include <netinet/in.h>  /* gives htons and friends */

struct hw3
  char x;
  char y;
  short z;
  int w;
};

char buffer [] = { 0x12, 0x34, 0x56, 0x78, 0x9a, 0xbc, 0xde, 0xf0 };

struct hw3 * sp = (struct hw3 *) buffer;
```

Escreta uma função com nome `print_buffer` para imprimir todos os *bytes* de um *buffer* em hexadecimal. Para o exemplo acima, seu código deve imprimir:

```
12 34 56 78 9a bc de f0
```

A função `print_buffer` recebe dois parâmetros, um ponteiro para o *buffer* e o número de *bytes* a serem impressos.

Agora escreva outra função chamada `print_struct` que imprime todos os valores em uma estrutura do tipo `struct hw3`. Se `x` é $0x12$, `y` é $0x34$, `z` é $0x5678$ e `w` é $0x9abcdef0$, sua função deve imprimir:

```
sp->x is 0x12, sp->y is 0x34, sp->z is 0x5678, sp->w is 0x9abcdef0
```

Os valores na estrutura estarão na ordem de *bytes* da rede (*big-endian*), mas você DEVE imprimir na ordem de *bytes* do computador (*little-endian*). A função `print_buffer` recebe apenas um parâmetro: um ponteiro para `struct hw3`.

Os protótipos dessas duas funções são:

```
void print\_buffer (char * buffer, int num\_bytes);
void print\_struct (struct hw3 * sp);
```

Na sua `main`, você deve invocar `print_buffer` e `print_struct` no `buffer` e `sp`. Sua saída deve ser como no exemplo acima.

Em seguida, use `sp` para definir o valor do `buffer` com as regras a seguir (cada valor de `sp` deve estar no formato *big-endian*, que é a como os *bytes* são ordenados na rede):

- `sp->x` deve ter o valor $0x77$
- `sp->y` deve ter o valor $0x92$
- `sp->z` deve ter o valor $0x4389$
- `sp->w` deve ter o valor $0xabc89032$

Se você fez tudo corretamente, quando imprimir ser *buffer*, deverá ver:

```
77 92 43 89 ab c8 90 32
```

Esteja atendo que ao imprimir um campo caractere, ele captura automaticamente o sinal extendido. Por exemplo:

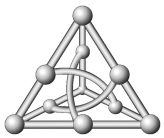
```
char foo [1] = { 0x89 };
print ("%02x\n", foo[0]);
```

irá imprimir `ffffff89`. Para retornar o valor desejado, você deve usar uma máscara:

```
char foo [1] = { 0x89 };
print ("%02x\n", foo[0] & 0xff);
```

Algo similar deve ser feito para variáveis (mas com $16bits$) `short`.

O nome do arquivo para esta parte do trabalho deve ser `sp.c`.



4 Manipulação de *Bits*

Escreva uma função:

```
unsigned int set_bits (unsigned int value);
```

que altere para 1 os dois *bits* mais significativos do valor de um único *byte*. `set_bits` deve imprimir o resultado em hexadecimal e retorná-lo.

Por exemplo, `set_bits (0x5)` deve imprimir e retornar $0xc5$ e `set_bits (0x72)` deve imprimir e retornar $0xf2$.

Se o valor for maior que 255 ($0xff$, o valor máximo de um *byte* único), `set_bits` deve imprimir somente e retornar os 8*bits* de ordem mais baixa. Use uma máscara para fazer isso (o operador &) e não uma expressão condicional (`if`).

Agora escreva outra função:

```
unsigned int combine_bits (int top, int middle, int low);
```

para computar resultados de 16-*bit* onde:

- o `top` (mais significantes) 2 *bits* são o primeiro parâmetro;
- os próximos 4 *bits* são o segundo parâmetro;
- o final (menos significantes) 10 *bits* são o terceiro parâmetro.

Sua função deve imprimir o resultado em hexadecimal e retorná-lo.

Por exemplo, `combine_bits (0x2, 0x5, 0x123)` deve imprimir e retornar $0x9523$, `combine_bits (0x1, 0x9, 0x321)` deve imprimir e retornar $0x6721$.

Se você ficar confuso como como fazer isso, converta o valor hexadecimal em binário, escreva o binário em um papel e converta (após as operações) o binário novamente em hexadecimal. Olhe a diferença, o valor $0x123$ é 291 em decimal e o valor $0x321$ é 801 em decimal.

Note que o seu programa não deve fazer nenhuma conversão para binário (números em C já estão em binário, o que o `printf` faz é converter inteiros binários para decimal ou hexadecimal), mas apenas usar operações de deslocamento (*shift* $<<$ e $>>$) e bit a bit (`|` & e ~).

Novamente, se o valor for maior que 65.535 ($0xffff$, o máximo valor para dois *bytes*), `combine_bits` deve imprimir e retornar somente os 16 *bits* de ordem mais baixa. Use uma máscara para fazer isso (o operador &) e não uma expressão condicional (`if`).

Essa função deve estar em um arquivo em C chamado `bits.c`.