Laboratório 02

Prof Bruno Silvestre

1. A função abaixo faz um “dump” de uma região de memória:

```
#include <stdio.h>

void dump (void *p, int n) {
  unsigned char *p1 = p;
  while (n--) {
    printf("%p - %02x\n", p1, *p1);
    p1++;
  }
}
```

Esse *dump* mostra uma sequência de posições da memória, exibindo, para cada posição, o seu endereço e o seu conteúdo (em hexadecimal). O parâmetro “p” é o endereço inicial a ser mostrado, e “n” é o número de bytes.

Um exemplo de uso da função *dump* é dado abaixo, mostrando como um inteiro é armazenado na memória:

```
int main (void) {
  int i = 10000;
  dump(&i, sizeof(i));
  return 0;
}
```

Faça as seguintes atividades:

(a) Crie um arquivo em C contendo as funções *dump* e *main*, compile-o e execute o programa. Explique detalhadamente o resultado desse programa. Experimente, se necessário, outros valores para “i”.

(b) Aplique a mesma técnica para ver como um *long* é armazenado.

(c) Aplique a mesma técnica para ver como um *short* é armazenado.

(d) Aplique a mesma técnica para ver como um *char* é armazenado. Lembre-se que uma variável do tipo *char* armazena um valor inteiro. Experimente atribuir a essa variável tanto um caractere como o valor do código ASCII correspondente (por exemplo, 'a' e 97).

(e) A mesma técnica pode ser usada para vermos como uma *string* é armazenada. Modifique o exemplo abaixo para descobrir os códigos ASCII dos caracteres 'A', ' ', '\n', e '$':

```
int main (void) {
  char p[] = "7509";
  dump(p, sizeof(p));
  return 0;
}
```

2. Compile e execute o seguinte programa:

```
#include <ctype.h>
#include <stdio.h>

int string2num (char *s) {
  int a = 0;
  for (; *s; s++)
    a = a*10 + (*s - '0');
  return a;
}

int main (void) {
  printf("==> %d\n", string2num("1234"));
  printf("==> %d\n", string2num("1234") + 1);
  printf("==> %d\n", string2num("1234") + string2num("1"));
  return 0;
}
```

(a) Procure entender como funciona a função *string2num*. Mande imprimir o valor de "*s" e de "a" a cada iteração do *for*, ambos com %d. O que está acontecendo? O que será o valor de "(*s – '0')" a cada iteração?

(b) Modifique a função *string2num* do exercício anterior para converter *strings* contendo numerais escritos em qualquer base entre 2 e 10. A base em que o numeral está escrito deverá ser dada como um segundo parâmetro (do tipo *int*) para a função.

Teste sua função com as chamadas abaixo:

```
  printf("%d\n", string2num("777", 8));
  printf("%d\n", string2num("777", 10));
```

(c) Modifique novamente a função *string2num* para aceitar bases maiores que 10. Nesse caso, precisamos de "dígitos" para os valores a partir de 10. Use as letras a-z para isso, com "a" representando o "dígito" 10, "b" representando 11, e assim por diante, até "z" representando o "dígito" 35. (Qual a maior base que podemos usar com esse esquema?)

Dica: talvez as funções *isdigit*() ou *isalpha*() possam ser úteis.

Teste sua função com as chamadas abaixo:

```
printf("%d\n", string2num("1a", 16));
printf("%d\n", string2num("a09b", 16));
printf("%d\n", string2num("z09b", 36));
```