

# Correção da prova 03

Prática de Conceitos e Técnicas de Programação  
Departamento de Informática e Matemática Aplicada  
Centro de Ciências Exatas e da Terra  
Universidade Federal do Rio Grande do Norte  
Autor: David Déharbe

6 de dezembro de 2011

O programa tem um papel simples: imprimir o nome do diretório atual no sistema de arquivos. Sendo assim, o programa não precisa fazer nenhuma leitura, apenas imprimir este nome, que é um texto.

O nome do diretório atual pode ser obtido através da sub-rotina `getcwd`. O enunciado da prova dava algumas informações acerca desta sub-rotina.

Esta rotina é uma chamada ao sistema operacional para saber qual o diretório atual, e pode ser usada através da seguinte diretiva de inclusão de arquivo-cabeçalho:

```
#include <unistd.h>
```

A sub-rotina tem a interface `char * getcwd(char *buf, unsigned size);`, onde:

- O parâmetro `buf` é o endereço de um bloco de memória, e
- o parâmetro `size` deve ser o número de bytes deste bloco de memória.

Se a chamada à sub-rotina for bem sucedida, ela preenche o bloco passado em parâmetro com o nome completo do diretória atual e retorna o endereço deste bloco. Mas se o tamanho do bloco de memória não for grande suficiente para armazenar o nome do caminho completo, então a sub-rotina retorna o ponteiro nulo.

Portanto, o programa deve proceder como segue:

1. Criar os valores que são usados para chamar a sub-rotina `getcwd`:

- (a) O endereço de um bloco de memória onde o comando `getcwd` vai armazenar o nome do diretório atual;
  - (b) A capacidade deste bloco de memória.
2. Chamar a sub-rotina `getcwd` com estes valores
  3. Verificar o resultado da chamada:
    - (a) Se o resultado for diferente do ponteiro nulo, então o resultado é o texto que devemos imprimir.
    - (b) Se o resultado for o ponteiro nulo, então o bloco de memória que passamos não tinha capacidade suficiente. Neste caso é preciso criar um novo bloco de memória, com um tamanho maior, e recomeçar a partir da etapa 2.

Identificamos nestas etapas que precisaremos de dois valores: o endereço do bloco de memória onde o nome do diretório será armazenado, e a capacidade deste bloco. A nossa solução terá duas variáveis, digamos `buf`, para o endereço, e `size` para a capacidade deste bloco. A variável `buf` é o endereço de um bloco de memória onde serão armazenados caracteres: deve ser declarada do tipo `char *`. A variável `size` é uma capacidade inteira: deve ser declarada do tipo `unsigned`.

O algoritmo proposto é o seguinte:

- Definir uma capacidade inicial não nula (atribuir a `size` um valor positivo);
- Alocar um bloco de memória grande suficiente para armazenar `size` caracteres (atribuir a `buf` o endereço deste bloco).
- Repetir os passos seguintes:
  - Chamar a sub-rotina `getcwd` com os valores armazenados em `buf` e `size`.
  - Se o resultado for diferente de `NULL`, terminar a repetição pois o nome do diretório pôde ser armazenado no bloco de memória alocado.
  - Se o resultado for `NULL`, deve se alocar um bloco de tamanho maior:
    - \* liberar o bloco de memória anteriormente alocado, pois não vai servir;
    - \* aumentar o valor guardado em `size`;
    - \* atribuir um novo bloco de memória de capacidade suficiente para armazenar `size` caracteres.
- Imprimir uma linha com o texto guardado no bloco de memória `buf`.

A alocação dinâmica será realizada com as rotinas padrão `malloc` e `free`. A impressão será realizada com a rotina padrão `printf`. Também lançaremos mão de uma terceira variável que será responsável por determinar se a chamada à sub-rotina `getcwd` foi bem-sucedida ou não.

Uma implementação direta deste algoritmo é a seguinte:

```

#include <stdio.h> /* printf */
#include <stdlib.h> /* malloc, free */
#include <unistd.h> /* getcwd */

int main (void)
{
    char *   buf;      /* address of block where name will be stored */
    unsigned size;    /* capacity of block where name will be stored */
    int      success; /* indicates that call to getcwd is successful */

    size = 10;
    buf = malloc(sizeof(char) * size);
    do {
        if (getcwd(buf, size) == NULL) {
            success = 0;
            free(buf);
            size += 10;
            buf = malloc(sizeof(char) * size);
        } else {
            success = 1;
        }
    } while (!success);
    printf("%s\n", buf);
    return 0;
}

```

Este código pode ser simplificado assim:

```

#include <stdio.h> /* printf */
#include <stdlib.h> /* malloc, free */
#include <unistd.h> /* getcwd */

int main (void)
{
    char *   buf;      /* address of block where name will be stored */
    unsigned size;    /* capacity of block where name will be stored */
    size = 10;
    buf = malloc(sizeof(char) * size);
    while (getcwd(buf, size) == NULL) {
        free(buf);
        size += 10;
        buf = malloc(sizeof(char) * size);
    }
    printf("%s\n", buf);
}

```

```
    return 0;  
}
```

Note que o cerne da solução cabe em seis linhas de código fonte.