PUC-Rio – Software Básico – INF1018
Prova 2 – 23/11/2010 – Turma 3WB

Atenção: Não esqueça de colocar o seu nome e número de matrícula na(s) folha(s) de respostas. Seja claro e objetivo. **Boa sorte!**

1. (2,5 pontos) No segundo trabalho foi pedida a implementação da função `create_func`, que tem o seguinte protótipo:

   ```
   DynCall create_func (void *f, const char *signature)
   ```

   `create_func` cria dinamicamente uma função que recebe como parâmetro um *array* com valores a serem passados para `f`, chama `f`, e armazena na primeira posição do *array* (índice 0) o valor de retorno de `f`.

   A assinatura de `f` é descrita através da *string* `signature`, onde os tipos dos parâmetros e do valor de retorno são indicados pelos caracteres `i, c, d, f` e `p`. O caractere indicando o tipo do valor de retorno (caso haja algum) deve ser precedido pelo caractere '`>`'.

   O *array* passado à função criada por `create_func` tem o seguinte tipo:

   ```
   typedef union {
         int i;
         char c;
         double d;
         float f;
         void* p;
     } Any;
   ```

   Forneça o código **assembly** (atenção: **não o código de máquina!**) das funções geradas por `create_func` para os casos abaixo. **Comente o seu código! (requisito indispensável)**

   (a) `create_func(f1, "pd>c");`

   (b) `create_func(f2, ">f");`

   Observação: no código assembly você pode usar o nome da função (`f1` ou `f2`) na instrução (`call`) que realiza a sua chamada.

2. (2,5 pontos) Escreva uma função `mywrite` em assembly IA-32 que receba um descritor de arquivo e uma *string* (isto é, um *array* de caracteres terminado em '`\0`'), e escreva a string no arquivo:

   ```
   int mywrite (int d, char *s);
   ```

   A função `mywrite` deve chamar diretamente o serviço `write` do Sistema Operacional para fazer a escrita. Esse serviço recebe três parâmetros:

   - o descritor do arquivo (um inteiro)
   - o endereço do buffer a ser escrito
   - o número de bytes a escrever (um inteiro)

   Para construir a chamada ao sistema, lembre-se que o número da chamada é colocado em `%eax` e os parâmetros do serviço nos demais registradores. O número da chamada `write` é 4.

3. (2,5 pontos) Considere o módulo abaixo:

```
#include <stdio.h>
extern int j;
float k = 2.0;
static int l = 1;

int f(float y);

int h(int z, float w) {
  int a;
  a = z * f(w);
  printf("%d\n", a);
}
```

   (a) Liste os símbolos exportados (definidos neste módulo) e importados (referências externas) para este arquivo, ou seja, o que apareceria como D (dados exportados), T (código exportado) ou U (undefined) na listagem do `nm`. Há algum símbolo em uma categoria diferente de D, T e U?

   (b) Considere agora o seguinte módulo :

```
#include <stdio.h>
extern int k;
... /* mais declarações */

int main() {
  if (k == 2)
    printf("valor de k é igual a 2\n");
  else
    if (k < 2)
      printf("valor de k é menor que 2\n");
    else
      printf("valor de k é maior que 2\n");
  ... /* mais código */
}
```

   Se este módulo fosse "ligado" com o módulo anterior para compor um programa executável, qual seria o resultado da chamada a `printf` (ou seja, qual o valor exibido)? **Justifique a sua resposta!** Respostas sem justificativa não serão consideradas.

4. (2,5 pontos) Traduza a função `soma_map` abaixo para assembly IA-32 do gcc/Linux (visto em sala), utilizando as regras básicas de passagem de parâmetros e retorno de resultado em C. **Não esqueça de comentar o seu código!**

```
double map(double d);

double soma_map(float *a, int n) {
  double d = 0.0;
  float *p;
  int i;

  for (i = n, p = a; i > 0; i--, p++)
    d = d + map(*p);

  return d;
}
```