Programação de Computadores

Aula #08

Ponteiros para Funções

O que são ponteiros para função? Como chamar uma função por meio de um ponteiro? Como trabalhar com parâmetros neste contexto?

Ciência da Computação - BCC e IBM - 2024/01
Prof. Vinícius Fülber Garcia

Mergulho Profundo: Funções

Você já parou para pensar **como o próprio código-fonte é compreendido** durante o processo de execução?

Existem múltiplos conceitos importantes para compreender isso, alguns são:

- Pilha de memória
- Ponteiro de pilha
- Registradores
- ...

MEMÓRIA

Mergulho Profundo: Funções

Na prática, acessamos a memória não apenas para acessar dados, mas também para acessar instruções; códigos que devem ser executados:

```
int plus_one(int a) {
      return (a + 1);
}
```

```
func:
      push    ebp
      mov     ebp, esp
      mov     edx, [ebp+8]
      mov     eax, [ebp+12]
      add     eax, edx
      pop     ebp
      ret
```

Mergulho Profundo: Funções

Naturalmente, podemos deduzir que é possível utilizar um ponteiro para apontar uma posição de memória relacionada às instruções presentes nos binários de um programa.

Se esse ponteiro indicar uma posição de memória que define uma função, o que temos é um **ponteiro para função!**

Nota de rodapé: essas noções são intuitivas – na prática, é um pouco mais complexo.

Declarando um Ponteiro para Função

O protótipo para a declaração de um ponteiro de função é o seguinte:

```
tipo_de_retorno (*nome_do_ponteiro) (tipo_do_argumento_1, ...)
```

- **tipo_de_retorno**: o ponteiro de função deve ter o mesmo tipo de retorno da função para a qual ele aponta
- **nome_do_ponteiro**: nome do ponteiro, segue as mesmas regras de nomeação de variáveis
- **tipo_do_argumento**: na mesma ordem em que aparecem na função, deve-se declarar cada um dos tipos de argumentos a serem recebidos

Declarando um Ponteiro para Função

Função #1:

```
void funcao1 (){
  ...
}
```

Declaração de um ponteiro que suporta a função #1:

```
void (*ponteiro1) ();
```

Função #2:

```
float funcao2 (int x, float y){
  ...
}
```

Declaração de um ponteiro que suporta a função #2:

```
float (*ponteiro2) (int, float);
```

Declarando um Ponteiro para Função

Após a declaração do ponteiro, devemos então **atribuir ao mesmo uma função** que esteja conforme as suas especificações.

Função #1:

```
void funcao1 (){
  ...
}
...
void (*ponteiro1) ();
ponteiro1 = funcao1;
```

Função #2:

```
float funcao2 (int x, float y){
  ...
}
...
float (*ponteiro2) (int, float);
ponteiro2 = funcao2;
```

Usando um Ponteiro para Função

A utilização de um ponteiro para função é bastante simples: basta utilizar o seu nome como se fosse o próprio nome da função para fazer uma chamada.

Função #1:

```
void funcao1 (){
  ...
}
...
void (*ponteiro1) ();
ponteiro1 = funcao1;
ponteiro1();
```

Função #2:

```
float funcao2 (int x, float y){
  ...
}
...
float (*ponteiro2) (int, float);
ponteiro2 = funcao2;
float res = ponteiro2(10, 5.5);
```

Ponteiro para Função como Argumento

Outra possibilidade interessante alcançada via ponteiros para funções é a passagem dos mesmos como argumentos para outras funções.

Para fazermos isso, basta **declararmos o ponteiro para função como um dos argumentos da função** que o utilizará.

```
void falar(char *personagem, void (*mensagem)());
```

Ponteiro para Função como Argumento

```c
void funcao1 (){
  printf("Hello World\n");
}

void funcao2(){
 printf("Olá Mundo!\n");
}

void falar(char *personagem, void (*mensagem)()){
 printf("%s: ", personagem);
 mensagem();
}

int main(void) {
 char language[10] = "en";
 if (!strcmp("pt-br", language)) falar("Robert", funcao2);
 else if (!strcmp("en", language)) falar("Robert", funcao1);
 else printf("????\n");
}
```

No programa ao lado, **a função de fala do personagem é alterada** conforme a linguagem selecionada.

A função falar é genérica, sendo as palavras a serem faladas determinadas pela função enviada como argumento.

Exercício #8

Escreva uma função que **receba um vetor de números reais e um operador (no formato de ponteiro para função) e aplique o operador para cada célula** no vetor. Considere os operadores: sqrt, cbrt, ceil, floor.

Por exemplo, suponha que o vetor seja {1.1,2.3,3.6,4.8,5.9} e o operador seja *floor*, então o vetor resultante será {1.0,2.0,3.0,4.0,5.0}.

Obrigado!

Vinícius Fülber Garcia
inf.ufpr.br/vinicius
vinicius@inf.ufpr.br