Insper

03 - Representação de dados em RAM

Sistemas Hardware-Software - 2019/2

Igor Montagner

Parte 0 - experimentos

Vamos trabalhar com os arquivos `experimento0-4.c`. Compile e execute cada um deles e coloque suas saídas abaixo. Comente os resultados comparando os valores das constantes no código em *C* e a saída dos programas.

experimento0.c

experimento1.c

experimento2.c

experimento3.c

experimento4.c

Parte 1 - `struct`

A utilização de `struct` junta tudo que já vimos sobre representação de todos os tipos de dados na memória. Não se esqueça de levar em conta as questões de alinhamento mostradas na parte expositiva da aula.

> **!** Você **não** deverá abrir o código de `parte1.c`. Ele está no repositório para você poder conferir suas respostas.

Exercício 1: Execute *parte1* no terminal. Os endereços mostrados na saída do programa pertencem a um só struct declarado como `struct player one`. Quais são seus campos?

Exercício 3: A segunda parte da saída de `parte1` mostra os endereços de cada campo do struct. Note que `&one` e `&one.icon` são iguais. Você consegue explicar por que? Se não, volte na aula expositiva e reveja a parte sobre `struct` e alinhamento de memória.

Exercício 2: Baseado nesses endereços, declare abaixo o `struct`. Escreva-o da mesma maneira que usaria em um programa em *C*.

Parte 2 - executando programas

Podemos examinar um programa durante sua execução usando o *gdb*. Podemos parar em qualquer instrução do programa, examinar conteúdo de registradores e da memória e listar todos os símbolos disponíveis (que podem ser funções ou variáveis globais). Por conveniência os códigos fonte dos programas estão disponíveis, porém isto se tornará cada vez menos comum conforme avançamos no curso.

Exercício 0: Para carregar um programa usando o *gdb*

```
$> gdb exemplo1
```

Isto nos colocará em um prompt esperando comandos. Se digitarmos

```
(gdb) run
```

o programa será executado. Para sair use

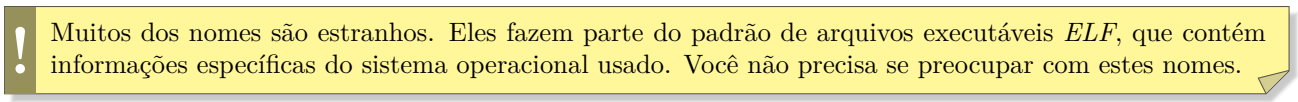
```
(gdb) quit
```

O *gdb* é uma ferramenta poderosa que possui muitas opções. Sua documentação está online e pode ser vista em https://sourceware.org/gdb/current/onlinedocs/gdb/index.html#Top.

Nesta primeira parte iremos abrir o arquivo *parte2.c* e olhar seu conteúdo. Também executaremos o programa compilado *parte2*.

Exercício 1: Abra o código *parte2.c* e liste os nomes das variáveis globais declaradas e seus tamanhos. Anote também as funções declaradas.

Exercício 2: O comando *info* mostra informações que podem ser obtidas a partir de um executável. Pesquise como usar este comando para listar as variáveis globais e as funções deste executável. Você consegue encontrar as informações listadas no exercício anterior? Escreva abaixo os comandos usados e as informações obtidas



Exercício 3: Qual é o significado da primeira coluna do comando *info variables*?

Exercício 4: Podemos usar o comando *print* para mostrar o valor inicial das variáveis globais identificadas. Faça isto para as variáveis identificadas acima. Escreva os comandos abaixo.

Dicas:

- Pode não funcionar de primeira. O quê a mensagem de erro diz?
- A sintaxe de conversão de tipos (*casting*) pode ser útil aqui.

Exercício 5: Podemos usar o *gdb* também para **examinar a memória** de um executável. A seção *10.6* da documentação explica como usar o comando `explore` (`x`) para examinar a memória. Acesse esta página da documentação, entenda seu conteúdo e escreva abaixo qual o comando usado para mostrar o conteúdo do segundo item do vetor `global_array`.

Dicas:

1. volte no exercício 3 e encontre o endereço do início do vetor
2. calcule o endereço do segundo elemento manualmente e passe para o comando `x`
3. não se esqueça de buscar na memória o tamanho correto do elemento

Exercício 6: Mostre agora o conteúdo de cada um dos bytes do item anterior. Ou seja, você deverá dar dois comandos para examinar a memória. Explique como converter esses dois bytes para o valor mostrado no exercício anterior.

Exercício 6: Use agora o comando `x` para exibir `global_str` como uma *string*. Escreva o comando abaixo.