Insper

03 - Representação de dados em RAM

Sistemas Hardware-Software - 2018/2

Igor Montagner

Neste handout vamos trabalhar com o *gdb* para analisar o conteúdo de memória de um programa. Usaremos os programas contidos nopacote *src.zip*. 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.

Faça os exercícios abaixo com o executável *exemplo1* carregado.

Exercício 1: Abra o código *exemplo1.c* e liste os nomes das variáveis globais declaradas e seus tamanhos.

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. Escreva abaixo todos os nomes listados que você consegue identificar como pertencendo ao código do *exemplo1*.

> **!** Muitos dos nomes são estranhos. Eles fazem parte do padrão de arquivos executáveis *ELF*, que contém informações específicas do sistema operacional usado. Você não precisa se preocupar com estes nomes.

Exercício 3: 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. **Dica:** Pode não funcionar de primeira.

Exercício 4: Qual é o significado da primeira coluna do comando *info variables*?

Exercício 5: Podemos usar o *gdb* também para **examinar a memória** de um executável. Pesquise como fazê-lo e escreva abaixo qual o comando usado para mostrar o conteúdo do segundo item do vetor `global_array`.

Exercício 5b: 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: Pesquise agora como usar **examinar a memória** para mostrar o conteúdo de `global_str`. Escreva o comando abaixo.

Vamos agora trabalhar com o arquivo *exemplo2*. Neste exercício você **não deve olhar o código** no arquivo *exemplo2.c*.

Exercício 7: Execute *exemplo2* usando *gdb* e anote abaixo os endereços mostrados no terminal.

Exercício 8: Usando seus conhecimentos de estruturas e alinhamento de memória, declare abaixo `struct player` usando somente a saída de *exemplo2*. Então, desenhe o layout de `struct player` na memória. Não deixe de conferir se o tamanho final bate com `sizeof(struct player)`.

Exercício 9: Encontre a variável global `struct player one` no executável e liste os valores iniciais de cada um de seus campos. Você deve usar o layout acima como guia para encontrar os endereços relativos de cada campo.

Exercício 10: Qual o significado do conteúdo da variável `ptr_struct`? Relacione com os itens respondidos anteriormente.

Exercício 11: Modifique `struct player` para que ele ocupe a menor quantidade de espaço possível. Desenhe o novo layout abaixo. Confira que o tamanho de seu desenho bate com o novo `sizeof(struct player)`.