Insper

12 - Alocação Dinâmica de Memória

Sistemas Hardware-Software - 2020/1

Igor Montagner

Parte 1 - Aquecimento

Exercício 1: Abra o arquivo *ex1.c* em um editor e compile-o usando as flags da disciplina (`-Og -Wall -std=c99`). **Sem rodar o programa**, responda as questões abaixo.

a) Analisando seu código-fonte, o que este programa faz?

b) Na execução deste programa, existe alguma possibilidade da alocação dinâmica com `malloc` falhar? Caso sim, indique as situações onde isto poderia acontecer.

c) O seu programa libera toda memória que aloca? Se não, aponte onde ele deveria fazer isto.

Exercício 2: Execute o programa acima e responda as questões abaixo.

a) Ocorreu algum problema durante a execução?

b) O comportamento de seu programa muda conforme N? Vá incrementando de um em um e veja o que acontece. Você consegue explicar por que? Discuta com seu grupo e valide sua resposta com o professor.

b) Existem três problemas no código. O primeiro (`vetor` não é desalocado) já indetificamos no exercíco anterior. Você consegue identificar os outros dois? Corrija-os e salve o programa em um arquivo *ex1-certo.c*.

Parte 2 - Ferramentas de verificação de memória

Para poder identificar mais facilmente problemas relativos a memória, iremos utilizar uma ferramenta chamada Valgrind.

O Valgrind é um detector de má gestão de memória. Ele roda seu programa em cima de um ambiente modificado e aponta os seguintes erros:

1. memória alocada e não liberada
2. acessos (leituras e escritas) a posições de memória não alocada ou inválidas

Instale o Valgrind com os seguintes comandos:

```
sudo apt-get update
```

`sudo apt-get install valgrind`

Para que os problemas encontrados pelo Valgrind sejam mais facilmente identificados, iremos passar a compilar utilizando a flag `-g`.

Exercício 3: Vamos rodar agora o programa acima usando o valgrind. Estamos supondos que tanto `ex1` quanto `ex1-certo` foram compilados com `-g` em adição às flags já usadas.

a) Rode o Valgrind com `valgrind --leak-check=yes ./ex1`. Quais foram os problemas encontrados e em quais linhas do código?

b) O quê significa o primeiro erro? Como corrigí-lo?

c) O quê significa o segundo erro? Como corrigí-lo?

d) A seção *HEAP SUMMARY* faz um resumo dos dados alocados/desalocados no seu programa. Ela mostra algum problema? Se sim, qual linha de código é apontada? Qual é o problema diagnosticado por este aviso?

e) Verifique que seu programa corrigido *ex1-certo.c* roda sem erros no valgrind. Se não, corrija os problema e rode novamente até que rode sem erros.

Parte 3 - implementações de funções (Entrega)

Exercício 4: Abra o arquivo *ex4.c* e implemente a função *mystrcpy*. Esta função recebe uma string, e devolve uma cópia da string original, alocando apenas o espaço realmente necessário.

a) Efetue alguns testes no terminal e confira se está ok.

b) Confira com o Valgrind se a sua implementação produz algum erro em relação aos acessos de memória.

Exercício 5: Abra o arquivo *ex5.c* e implemente a função *mystrcat*. Esta função recebe duas string, e devolve uma terceira que é a concatenação das duas primeiras, alocando apenas o espaço realmente necessário.

a) Efetue alguns testes no terminal e confira se está ok.

b) Confira com o Valgrind se a sua implementação produz algum erro em relação aos acessos de memória.

Exercício 6: Você percebeu que, no código base dos dois exercícios anteriores a memória alocada dinamicamente não foi devolvida ao sistema? Nestes casos, explique qual parte do código deve ser responsável pela liberação e por que?

Parte 4 - implementação de programas completos

Ao finalizar os exercícios abaixo cheque se sua solução não produz erros de memória usando o *Valgrind*.

Exercício 7: Crie um programa para ler o preço de fechamento do dia de uma determinada ação para n dias. Após a leitura, mostre no terminal a média e o desvio padrão do preço de fechamento.

1. Você pode supor que o primeiro valor passado para seu programa será um inteiro n referente a quantidade de dias, seguido por n valores reais referentes ao preço de fechamento da ação em cada um dos n dias. Você precisará alocar um vetor de tamanho n para armazenar as leituras.
2. Neste item o número de ações não será passado no início do programa. O programa deverá parar quando receber o valor `0`. Inicialmente, aloque um vetor de cinco posições. Cada vez que o vetor estiver cheio e for necessária uma nova posição, aloque mais cinco utilizando a função `realloc`. A saída de seu programa deverá ser a mesma do ite anterior.