Programação de Computadores

Profs. Marilton Sanchotene de Aguiar e Giovani Parente Farias
{marilton, giovani.pfarias}@inf.ufpel.edu.br

Cursos de Ciência e Engenharia de Computação
Centro de Desenvolvimento Tecnológico
Universidade Federal de Pelotas

2020/2

Semana IX

Alocação Dinâmica

Alocação de memória diz respeito a como a memória (necessária para o armazenamento dos dados) é reservada em um programa

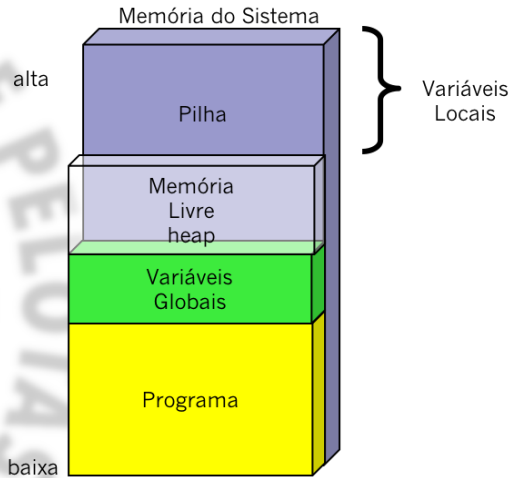
Existem 2 formas de um programa em C alocar memória:

- Estática
- Dinâmica

Alocação Estática

- Ocorre na declaração de variáveis globais e variáveis locais;
- No caso de variáveis globais e variáveis locais estáticas, o armazenamento é fixo durante todo o tempo de execução do programa;
- Variáveis globais são alocadas em tempo de compilação;
- No caso de variáveis locais, o armazenamento dura o tempo de vida da variável.

Mapa da Memória

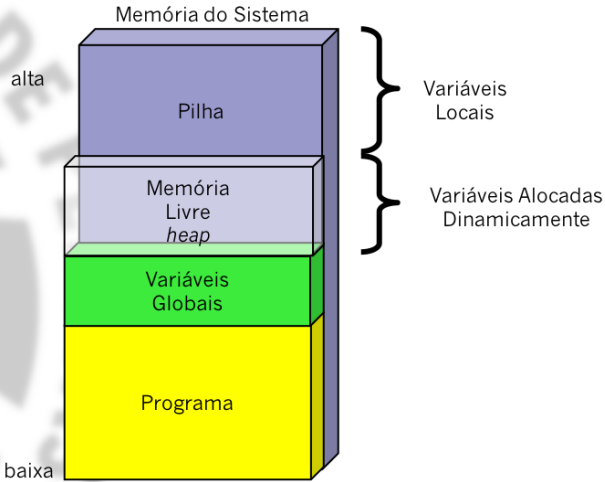


- Para a alocação estática, é necessário que se saiba de antemão (antes do início do programa) a quantidade de memória que será necessária;
- Estimativas podem ser feitas, porém há o risco de sub ou superestimar;
- Assim, muitas vezes é necessário que um programa possa ir ajustando a memória a ser usada durante sua execução;
- “Ajustar” = alocar ou desalocar

Alocação Dinâmica

- Alocação dinâmica é o método pelo qual um programa ajusta dinamicamente a quantidade de memória a ser usada durante sua execução
- Permite otimizar o uso da memória
- É implementado mediante funções de alocação/liberação da memória, as quais o programador precisa usar de maneira coerente

Mapa da Memória



Alocação Dinâmica

- Principais funções C para alocação dinâmica de memória: `malloc` e `free`;
- Implementadas na biblioteca `stdlib.h`
- Há diversas outras funções mais específicas, as quais normalmente estão implementadas em `stdlib.h`

- Serve para alocar memória;
- Devolve um ponteiro para o início da quantidade de memória alocada;

Exemplo

```
char *p;
p=(char *)malloc(1000);
```

O trecho acima aloca 1000 bytes de memória para o armazenamento de caracteres.

- Porém:
  - A memória ocupada por um determinado tipo pode variar de máquina para máquina!
  - As implementações devem ser independentes da máquina!
- Solução: usar o operador `sizeof`.

Exemplo

```
int *p;
p=(int *)malloc(50*sizeof(int));
```

Aloca memória suficiente para armazenar 50 inteiros (de maneira contígua na memória)

- Porém, pode ser que o programa tenha alocado muita memória, a ponto de não restar espaço na área de *heap* (efetivamente, isto pode ocorrer);
- Se `malloc` não conseguir alocar memória, ele retornará um ponteiro nulo;
- Logo, é preciso testar o resultado de `malloc`.

Exemplo

```
char *p;
if( (p=(char *)malloc( 50*sizeof(char) ) ) == NULL ) {
  printf("Não foi possivel alocar memoria\n");
  exit(1);
}
```

- Libera memória previamente alocada de maneira dinâmica, por meio de uma das funções de alocação dinâmica (devolve memória ao *heap*)
- Recebe como argumento um ponteiro para a porção de memória que se deseja liberar
- Jamais use `free` com um argumento inválido, pois isto destrói a lista de memória livre!!

```
free(p);
```

Veja

- `exemplo-57.c`
- `exemplo-58.c`
- Assista os vídeos da playlist disponível em `http://bit.ly/pc_aloc`

Exercício

1. Modifique o programa `exemplo-48.c`, de modo a substituir a alocação estática das estruturas envolvidas pela alocação dinâmica.