Universidade Federal do Ceará
CC - Campus Quixadá
Aluno: Antônio Andson
https://github.com/andsonsilv

25/03/2019

Alocação Dinâmica

A alocação dinâmica é diferente da alocação estática que não é possivel alterar o espaço de memória que as variáveis irão utilizar durante a execução do programa. De maneira informal, podemos dizer que existem três maneiras de reservar espaço de memória para o armazenamento de informações. A primeira é usar variáveis globais (e estáticas). A segunda maneira é usar variáveis locais. A terceira maneira de reservar memória é requisitar ao sistema, em tempo de execução, um espaço de um determinado tamanho.

cada vez que uma determinada função é chamada, o sistema reserva o espaço necessário para as variáveis locais da função. Esse espaço pertence à pilha de execução e, quando a função a função termina, é desempilhado. A parte da memória não ocupada pela pilha de execução pode ser requisita dinamicamente. Se a pilha tentar crescer além do espaço disponível existente, dizemos que ela "estorou", e o programa é abortado com erro. Da mesma forma, se o espaço da memória livre for menor do que o espaço requisitado dinamicamente, a alocação não é feita, e o programa pode prever um tratamento de erro adequado. [rangel2004introduccao]

Existem funções presentes na biblioteca *stdlib*(*malloc*, *calloc*, *realloc* e *free*), que permitem alocar e liberar memória dinamicamente. A função que vamos utlizar é a *malloc*. Vamos exemplicar, iremos alocar dinamicamente um vetor de inteiros com 10 elementos. O *malloc* tem como valor de retorno o valor da área alocada. Como desejamos armazenar valores inteiros nesta área, devemos declarar um ponteiro de inteiro para receber o endereço inicial do espaço alocado.

```
1 int *v;
2 v = malloc(10*4);
```

v armazenará o endereço inicial de uma área contínua de memória suficiente para armazenamento de 10 valores inteiros. No caso, consideramos que cada inteiro ocupa 4 bytes.

Para não ficarmos réfens de compiladores e máquinas específicas, vamos usar o operador *sizeof()*.

```
1 v = malloc(10*sizeof(int));
```

Porto bem interessante agora, é que a função *malloc* é usada para alocar espaço para armazenar valor de qualquer tipo. Por esse motivo, o *malloc* retorna um ponteiro genérico, para um tipo qualquer, representado por *void**, no entanto fazemos um *cast* para o tipo desejado, no nosso exemplo, um inteiro.

```
1 v = (int *) malloc(10*sizeof(int));
```

Com o espaço já determinado, podemos agora usar ele dinamicamente, vejam:

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main(){
5
6     int *v = (int *) malloc(10 * sizeof(int));
```

```
    for(int i = 0; i <= 10; i++){
        v[i] = i;
        printf("%d ", v[i]);
    }
    printf("\n");
    free(v);

    return 0;

}
```

Após o término do laço de repetição, não iremos mais precisar do espaço que foi alocado, então podemos liberamos utilizando o *free()*. Essa função recebe como parâmentro o ponteiro da memória a ser liberado. Abaixo segue a saída do programa acima.

```
$ 0 1 2 3 4 5 6 7 8 9 10
```

Se, por acaso, não houver espaço livre suficiente para realizar a alocação, a função retorna um endereço nulo (*NULL*). Podemos verificar do seguinte modo:

```
...
if( v == NULL){
    printf("Memoria Insuficiente.\n");
    exit(1); /* aborta o programa e retorna 1 para o SO */
}
...
```