Insper

Sistemas Hardware-Software

Aula 12 – Alocação de memória

2020 – Engenharia

Maciel Calebe, Igor Montagner <igorsm1@insper.edu.br>, Fábio Ayres

Alocação de memória

```c
#include <stdio.h>
#include <stdlib.h>

#define MAXW 512
#define MAXH 512

int main(int argc, char *argv[]) {
    int mat[MAXH][MAXW];

    /*
        trabalhar com arquivo PGM
    */

    return 0;
}
```

Alocação de memória

- Qual o consumo de memória do programa anterior?
- Ele varia conforme o tamanho da matriz?
- E se precisarmos de matrizes maiores?

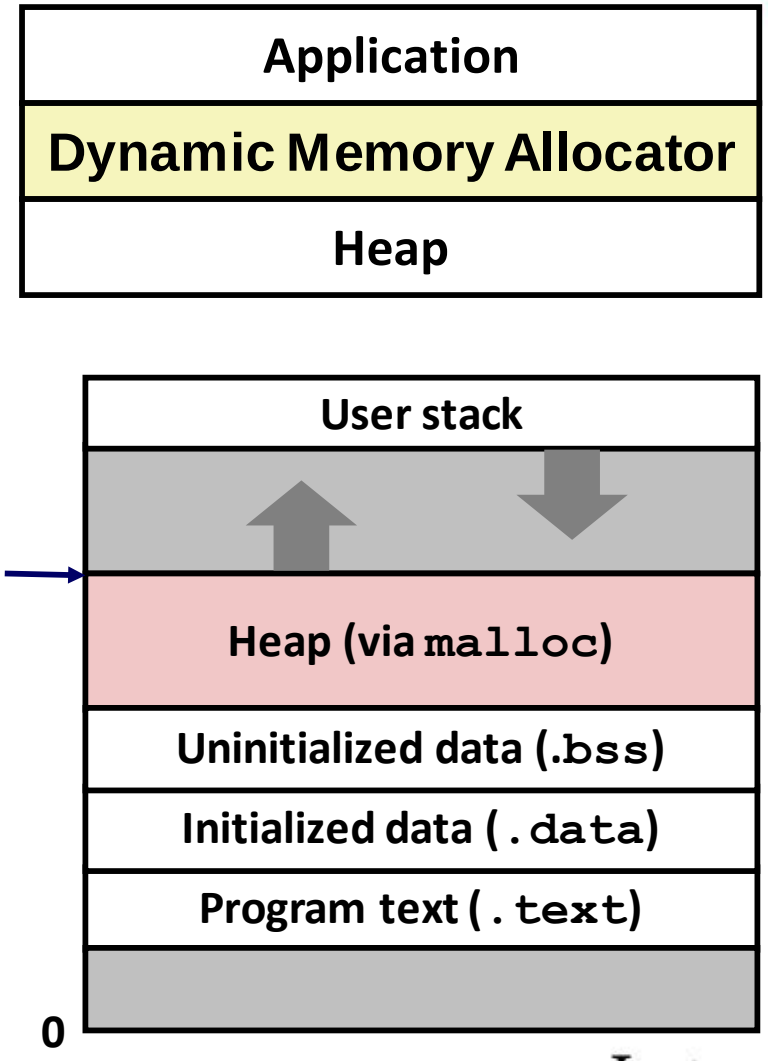
Alocação estática

```c
#include <stdio.h>
#include <stdlib.h>

#define MAXW 512
#define MAXH 512

int main(int argc
    int mat[MAXH]

    /*
        trabalhar com arquivo PGM
    */

    return 0;
}
```

- Tamanho definido no momento da compilação
- Armazenado na pilha (se for variável local) ou no arquivo executável diretamente (se for global)

Alocação dinâmica de memória

- Programas usam **alocadores de memória dinâmica** para criar e gerenciar novos espaços de memória virtual
  - C: malloc, free
  - C++: new, delete
- A área do espaço de memória virtual gerenciada por estes alocadores é chamada de **heap**



Alocação dinâmica de memória

- Alocadores organizam o heap como uma coleção de blocos de memória que estão alocados ou disponíveis

- Tipos de alocadores
  - Explícitos: usuário é responsável por alocar e dealocar (ou liberar) a memória.
    Exemplo: malloc, new
  - Implícitos: usuário não precisa se preocupar com a liberação da memória.
    Exemplo: garbage collector em Java

malloc

```
#include <stdlib.h>
void *malloc(size_t size)
```

Se bem sucedido: retorna ponteiro para bloco de memória com pelo menos **`size`** bytes reservados, e com alinhamento de 16 bytes (em x86-64). Se **`size`** for zero, retorna **`NULL`**.

Se falhou: retorna **`NULL`** e preenche **`errno`**

free

```
#include <stdlib.h>
void free(void *p)
```

Devolve o bloco apontado por **`p`** para o *pool* de memória disponível

Alocação dinâmica

```c
int main() {
    int w = 512;
    int h = 512;
    int *mat = malloc(sizeof(int) * w * h);
    // iremos considerar uma linha colocada atrás da outra
    mat[i * w + j] // acesso ao elemento [i][j].
```

- Tamanho definido em tempo de execução pelas variáveis `w` e `h`
- Representadas por um apontador para o começo do bloco alocado
- Permanece alocado até ser liberado com `free`

Alocação dinâmica

- Vantagens
  - Controle feito em tempo de execução
  - Economia de memória
  - Expandir / diminuir / liberar conforme necessário

- Desvantagens
  - Riscos da gerência
    - Liberar espaços não mais necessários
    - Não acessar espaços já liberados
    - Acessar apenas a quantidade requisitada
    - Etc.

Exemplo

```c
#include <stdio.h>
#include <stdlib.h>

void foo(int n) {
  int i, *p;

  /* Allocate a block of n ints */
  p = (int *) malloc(n * sizeof(int));
  if (p == NULL) {
    perror("malloc");
    exit(0);
  }

  /* Initialize allocated block */
  for (i = 0; i < n; i++) {
    p[i] = i;
  }

  /* Return allocated block to the heap */
  free(p);
}
```

Atividade

Exercícios 1, 2 e 3 da atividade

30 minutos

Ler memória não-inicializada

Bug clássico: assumir que dados no heap são pré-inicializados com zero

```
/* return y = Ax */
int *matvec(int **A, int *x) {
   int *y = malloc(N*sizeof(int));
   int i, j;

   for (i=0; i<N; i++)
      for (j=0; j<N; j++)
         y[i] += A[i][j]*x[j];
   return y;
}
```

Sobrescrever memória

Bug clássico: off-by-one!

```
int **p;

p = malloc(N*sizeof(int *));

for (i=0; i<=N; i++) {
   p[i] = malloc(M*sizeof(int));
}
```

Memory leaks

```
foo() {
    int *x = malloc(N*sizeof(int));
    ...
    return;
}
```

C++ tem uma boa solução para esse problema: smart pointers!

Atividade

Exercícios 4, 5 e 6

Outras funções

`calloc`: Versão de malloc que inicializa bloco alocado com zeros.

`realloc`: "Re-aloca" um bloco – muda o tamanho do bloco garantindo a integridade dos dados. Note que o bloco realocado pode mudar de lugar na memória!

`sbrk`: usado internamente pelos alocadores para aumentar ou diminuir o heap

Insper

www.insper.edu.br