Programação em Sistemas Computacionais

Linguagem C

Alocação de Memória

Centro de Cálculo
Instituto Superior de Engenharia de Lisboa

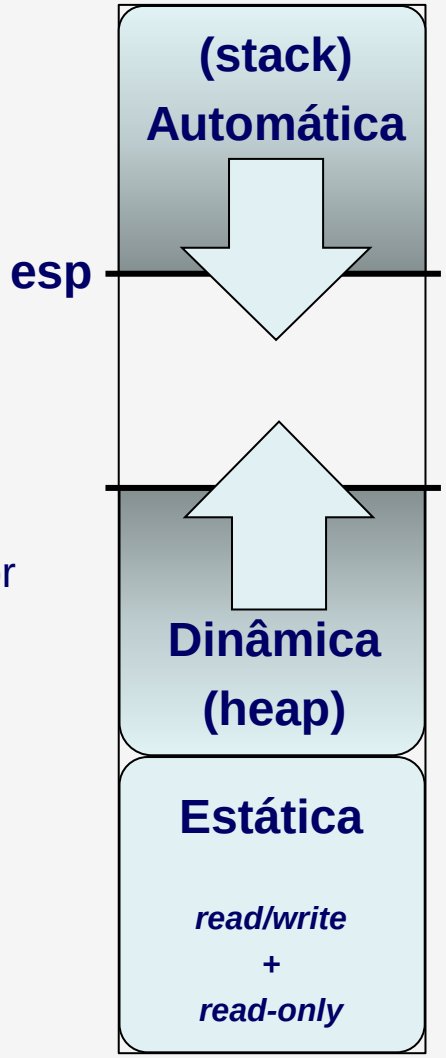
Pedro Pereira palex@cc.isel.ipl.pt

Tipos de Alocação de memória

- **Alocação estática**
  - Os dados a armazenar têm dimensão fixa ou limitada
  - A memória necessária pode ser determinada pelo compilador (*compile-time*)
  - Variáveis globais
  - Variáveis locais e parâmetros
- **Alocação dinâmica**
  - Os dados não têm dimensão fixa
  - A memória necessária só pode ser determinada durante a execução do programa (*run-time*)
  - Memória é alocada e libertada pelo programa (programador) dependendo da necessidade.

Tipos de memória

- **Estática (global)**
  - Reservada antes do programa iniciar
  - Libertada no final do programa
  - A sua dimensão total é determinada pelo compilador
  - Implicitamente iniciada
  - Usada para variáveis globais, constantes (strings), …
- **Automática (stack)**
  - Reservada em cada chamada à função
  - Libertada automaticamente quando a função retorna
  - A dimensão para cada função é determinada pelo compilador
  - Não é implicitamente iniciada
  - Usada para variáveis locais, parâmetros das funções, …
- **Dinâmica (heap)**
  - Reservada explicitamente pelo programador
  - Libertada explicitamente pelo programador
  - A dimensão é indicada no momento da reserva
  - Não é, normalmente, iniciada
  - Usada quando só se sabe a dimensão em *runtime*



Funções para alocação dinâmica

```
void * malloc(size_t size);
```

- **Aloca um bloco de memória não iniciada.**
  - Retorna o ponteiro para `size` bytes de memória não iniciada.
  - Retorna `NULL` se não for possível alocar.

```
void free(void *ptr);
```

- **Liberta um bloco a memória.**
  - Liberta a memória apontada por `ptr`.
  - A memória libertada foi alocada dinamicamente e ainda não foi libertada.

```
void *calloc(size_t n, size_t size);
```

- **Aloca um bloco de memória para um *array* iniciado com zeros.**
  - Retorna o ponteiro para `n` elementos de `size` bytes ou `NULL` se não for possível.

```
void *realloc(void *ptr, size_t size);
```

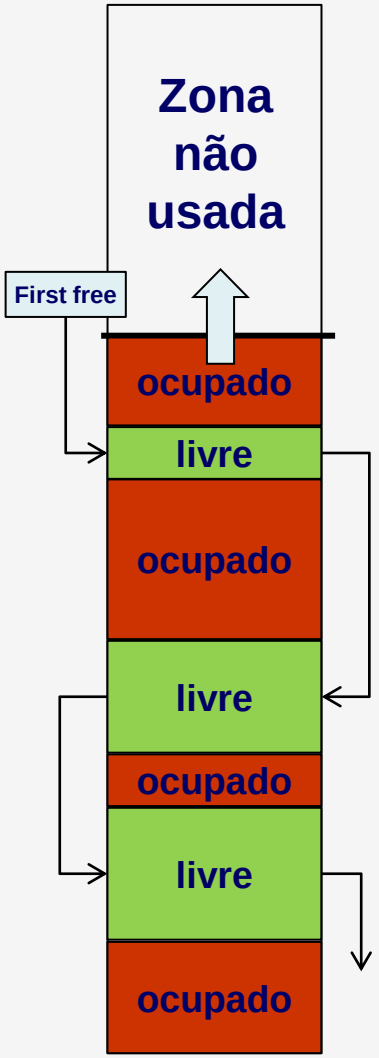
- **Altera a dimensão do bloco já alocado.**
  - Retorna o ponteiro `ptr` se o bloco não foi movido para outra posição.
  - O conteúdo do bloco antigo permanece.

Gestão do *Heap*

- **Os blocos libertados ficam ligados em lista na zona ocupada**

- **free(p)** de um bloco ocupado
  - Se está contíguo à zona não usada
    → A zona aumenta.
  - Senão é adicionado à lista de blocos livres.
    - Se contíguo a outro livre → Junta os blocos livres.

- **malloc(size)**
  - Se existe na lista dos blocos livres um >=size
    → Retorna esse e o resto fica livre
  - Senão retira à zona não usada.



Blocos alocados

- **Os blocos alocados têm:**
  - Espaço adicional para guardar a dimensão do bloco
    → Para a chamada a free() não ter um parâmetro adicional
    → Porque a dimensão dada pode ser maior que a pedida
  - Dimensão múltipla de 4, 8, 16, …
    → Para garantir alinhamento
  - Dimensão mínima (16 bytes aproximadamente)
    → Para armazenar na lista de blocos livres

```
char * ptr;

ptr = malloc( 1 );
*ptr = 'A';
...
```