TP03 - Sistemas Operacionais

Aluno: Nathann Zini dos Reis

Matricula: 19.2.4007

Tabela:

	Heap	Stack	Global.bss	Global.data	Por que?
hypercounter				x	A variável não é declarada dentro de esopo local, mas global e fora inicializada com 0
counter			x		A variável não é declarada dentro de esopo local, mas global e não fora inicializada com 0
*idk2	-	-	-	-	não encontrei essa variável
Cook1		x			A variável foi criada dentro do escopo main e foi alocada automaticamente e também será automaticamente desalocada.
clk1		x			A variável foi criada dentro do escopo main e foi alocada automaticamente e também será automaticamente desalocada.
cook2→c		x			A variável foi criada dentro do escopo main e foi alocada automaticamente e também será automaticamente desalocada.
*arr em clk1	x				A variável foi alocada dinamicamente, logo ela permanecerá alocada em memória até que seja desalocada pelo usuário

	Heap	Stack	Global.bss	Global.data	Por que?
*arr em clkk2	x				A variável foi alocada dinamicamente, logo ela permanecerá alocada em memória até que seja desalocada pelo usuário

Similar ao malloc baseado no heap, o stack allocation utiliza o método FILO (first-in last-out) através de uma pilha para alocar, normalmente, variáveis estáticas de tamanhos fixos. A desalocação desse método é feito automaticamente ao se encerrar o programa ou escopo em que está encontrado.

Conceitos:

heap memory

> Também chamada de memória dinâmica, diferentemente da stack memory, a heap memory pode ter uma tamanho muito grande, tão grande quanto permitido pelo SO. Porém, a memória alocada pela heap, além de ser relativamente de uso/acesso lento, ela precisa ser liberada pelo usuário ao completar seu uso, não sendo liberada automaticamente.

stack memory

> É a memória alocada em pilha. É criada dentro do escopo de função, como o main. Uma vantagem é que, quando acabar a execução do escopo em que ela se encontra, ela é liberada automaticamente. Porém a memória tem um limite de tamanho.

Global memory

> Memoria global existe no programa todo, é alocado pelo SO e geralmente há duas diferentes locações: inicializada com 0 e inicializada com algum outro valor. Não é recomendado o uso desse tipo de memória, pois aumenta o tamanho do programa no disco, aumentando tempo de compilação.

Fragmentação Externa

> A fragmentação externa refere-se ao processo em que a memória é alocada e desalocada em diferentes porções de tamanho, levando a grandes quantidades de

memórias não usadas e não alocadas. Acontece quando se tem memória alocada em porções diferentes de tamanho, e depois desaloca algumas. Quando tentar alocar um outro objeto naquele espaço livre, por mais que tenha espaço livre suficiente, ele pode estar fragmentado em porções de tamanho menor do que o necessário.

Eficiência

A alocação dinâmica específica de memória pode impactar a performance do programa, uma vez que o tamanho do caminho necessário para alocar um slot de memória é 52.

Fixed-size blocks

Se basea numa lista de blocos de memórias de tamanhos fixos (geralmente tem os mesmo tamanhos). É recomendado para sistemas embutidos que não se faz necessário alocar grandes quantidades de objetos, porém sofre com a fragmentação, principalmente com endereços com memórias longas.

Buddy blocks

A memória é alocada em várias pools de memórias, ao invés de apenas uma, em que cada pool representa um bloco de memórias de mesmo tamanho. todos os blocos de mesmo tamanho são guardados em uma espécie de árvore e novos blocos de mesmo tamanhos são adicionados em suas respectivas pool.

Slab Allocation

Esse método pré-aloca pedaços de memórias para objetos de tipos ou tamanhos específicos. Esses pedaços são cachamos de cachos e o alocador apenas tem que manter uma lista de slots de cache livre.

Stack Allocation

Similar ao malloc baseado no heap, o stack allocation utiliza o método FILO (first-in last-out) através de uma pilha para alocar, normalmente, variáveis estáticas de tamanhos fixos. A desalocação desse método é feito automaticamente ao se encerrar o programa ou escopo em que está encontrado.

Alinhamento de Memoria

O alinhamento da memória pode ser necessário quando se programa sistema. O alinhamento é referente a uma padronização do endereço de memória gerado pelo alocamento.

- Globalmente: é utilizado o alignas() ao criar uma variável fora de qualquer escopo local, ou seja, de forma global.
- Na Pilha: é utilizado o mesmo alignas(), mas desse vez dentro de um escopo local. Porém, ao alinhar em pilha, indiretamente é alocado memória adicional na pilha inutilizável, que dependendo do tamanho do alinhamento, pode esgotar a memória/espaço disponível para fazer o alocamento em pilha.
- Dinamicamente: pode ser usado com o aligned_alloc() que é similar ao malloc(), que aloca dinamicamente uma porção da memória para a variável, porém adicionalmente é passado por parâmetro o tamanho do alinhamento que será usado para alinhas aquele endereço de memória.