Memória Stack x Memória Heap

Pedro Benício Januário Oliveira

BCC/ICMC - n°USP 12543843

Principais diferenças

Stack x Heap	
Variáveis locais aos seus escopos	Variáveis globais
Alocação automática do compilador	Alocação manual do algoritmo
Alocação em endereços de memória sequenciais	Alocação em células de memória não sequenciadas
Abandonados após o uso da memória	Necessidade de desalocação manual por parte do desenvolvedor

Organização e comportamento

A Stack tem sua organização e comportamento de alocação análogos à dinâmica da estrutura de dados que leva o mesmo nome: uma forma otimizada de organizar os dados na memória alocando-os em sequência direta e os abandonando na sequência inversa à de alocação, assumindo assim a identidade de pilha ("abandonados" pelo fato de normalmente não haver desalocação automática por parte do compilador).
A Heap, com tradução livre pra monte ou amontoado, possui uma organização de memória de natureza mais flexível, uma vez que não tem a necessidade de alocação sequencial - podendo assim alocar em quaisquer células de memória disponíveis.

Capacidade de armazenamento

Uma vez que a capacidade de armazenamento da Stack se delimita ao espaço de memória destinado ao uso do programa que alocará e acessará as variáveis dessa mesma Stack, tem-se que sua memória é bem limitada em tamanho - gerando inclusive um fatal assim que haja a tentativa de acessar uma célula de memória alocada fora do espaço reservado para uso da Stack (denominado assim *Stack Overflow*).
Já a Heap, que não possui necessidade de alocamento sequencial, tem a possibilidade de alocar memória fora da Stack destinada ao programa, podendo assim alocar até os últimos bytes disponíveis na memória RAM.

Por que alocar na Heap?

Em geral, para alocações de tamanhos indeterminados ou simplesmente grandes demais, o recomendado é fazer uso da Heap, tendo assim a possibilidade de mudar a quantidade de memória reservada para as variáveis ao longo do algoritmo além de otimizar a distribuição das células de memória utilizadas pela estrutura de dados alocada.
Também podem ser motivos para se optar pela Heap pontos como a necessidade de globalidade da variável alocada, usando a alocação na Heap como um modo de evitar que a variável morra ao fim do escopo em que foi criada; ou então, por fim, a necessidade de liberação do endereço de memória fora do padrão sequencial imposto pela lógica de pilha da Stack.

Por que alocar na Stack?

Apesar de na maior parte dos pontos levantados a Heap proporcionar mais vantagens, a Stack ainda se mantêm uma ótima opção. Não por possuir numerosas vantagens, mas por ter as suas em campos extremamente interessantes para possíveis otimizações no algoritmo. A Stack não só oferece maior facilidade de manipulação e manejamento de variáveis e endereços de memória, como também é significativamente mais rápida que a Heap, podendo trazer melhoras relevantes para a performance do programa. O abuso da Stack, por mais perigoso que seja, é extremamente benéfico para o programa e apesar dos riscos se faz uma ótima opção caso seja de conhecimento e controle dos desenvolvedores.