Trabalho Prático de Software Básico

Caio Rufino (GRR 20224386), Frank Hannemann (GRR 20224758), Henrique Biehl (GRR 20221257)

A implementação da alocação dinâmica memória foi realizada por meio de 2 etapas. A primeira etapa consistiu em criar uma biblioteca na linguagem C para poder estruturar o fluxo de execução do programa. Para isso, foram implementadas 4 funções - setup_brk, dismiss_brk, memory_alloc e memory_free - e 2 variáveis globais - original_brk e current_brk.

Na implementação da alocação de memória foi utilizado o método "first-fit", onde, ao requisitar uma alocação de memória é escolhido o primeiro bloco livre com bytes suficientes. Caso, além dos bytes requeridos, haja espaço para a criação de um novo bloco com no mínimo 17 bytes - 2 quadwords de registro e 1 byte de bloco -, este é criado na heap. Por fim, caso não haja um bloco que atenda as necessidades, é criado um novo bloco com tamanho de 2 quadwords e dos bytes requeridos.

Na liberação de memória, é primeiro verificado se o ponteiro passado não está à frente do que está armazenado em current_brk. Pois, nesse caso, implica que o endereço está na stack. Caso o ponteiro esteja na heap, o registro de uso é indicado como livre - ou seja, 0 - e o retorno da função é 0 (null).

Já a segunda etapa da implementação consiste na tradução do código em C para assembly. Para fins de depuração, foi feito uso constante do GDB para acompanhar os valores de cada registrador e o fluxo de execução do código. Durante o processo de testagem em assembly foi percebido pelo grupo a necessidade da verificação da posição do ponteiro passado como parâmetro na função memory_free, devido ao erro que apontavam para “stack free”.

Os testes feitos consistem na rotina de testes dada no moodle e em uma série de testes no arquivo teste.c que foca em analisar como está a organização dos blocos e se o encadeamento dele está correto.